Diseño e implementación de una base de datos relacional para la gestión sanitaria

Proyecto Final de Carrera

Autor:
Francisco Serrano Peris
Ingeniería en Informática

Consultor:
Juan Martínez Bolaños

12 de Junio de 2013

A mi mujer Lidón,
por su apoyo en todo momento.

Resumen

El proyecto final de carrera es una síntesis de los conocimientos adquiridos a lo largo de la carrera, para aplicarlos en un proyecto del mundo real. Éste se enmarca en el área de Bases de datos.

El ejercicio que se nos plantea es desarrollar una base de datos para la gestión integral del sistema sanitario. Además, también tendremos que crear un almacén de datos para facilitar el análisis de la situación del sistema. Para la consecución del proyecto deberemos planificar el trabajo, analizar los requisitos y realizar la implementación.

El Ministerio de Sanidad nos plantea un escenario en el que cada centro hospitalario y cada farmacia funcionan de forma autónoma y sin estar en contacto con el resto de centros. La base de datos a desarrollar deberá soportar la gestión de los pacientes, los médicos; las consultas externas y de urgencias; el consumo de medicamentos, las pruebas médicas, las enfermedades contraídas, las bajas laborales, etc.; de una forma centralizada.

Esta base de datos no podrá ser accedida directamente por ningún sistema externo, el intercambio de información se realizará siempre a través de procedimientos almacenados.

En el desarrollo del proyecto utilizaremos el sistema de gestión de bases de datos Oracle 11*g* Express Edition y la herramienta OLAP Pentaho para explotar el almacén de datos.

Índice de contenido

Índice de figuras

1. Introducción

En este primer apartado de la memoria del Proyecto Final de Carrera (PFC) describiremos el proyecto que hemos realizado. Indicaremos tanto los objetivos que se persiguen con la realización del proyecto, como el alcance del mismo. También expondremos el método de trabajo y la planificación que hemos seguido para su consecución. Por último, mostraremos los entregables del proyecto; y un breve comentario de cada uno de los apartados que forman este documento.

1.1. Justificación del PFC y contexto en el que se desarrolla

El objetivo de este PFC es realizar un ejercicio de síntesis de los conocimientos adquiridos a lo largo de la carrera. Al ser un proyecto del área de Base de Datos, aplicaremos lo estudiado en asignaturas como Sistemas de Gestión de Bases de Datos y Modelos Multidimensionales de Datos; ya que el proyecto incluye el diseño e implementación de un almacén de datos. A partir de ahora lo denominaremos indistintamente como almacén de datos o *data warehouse* (DWH).

Como en todo proyecto, será necesaria una planificación del mismo; lo que nos obliga a poner en práctica lo aprendido en asignaturas como Metodología y Gestión de Proyectos Informáticos. Deberemos afrontar la realización de todas las tareas que intervienen: toma de requisitos, análisis, diseño, implementación, pruebas y documentación.

El propósito del proyecto es diseñar e implementar una base de datos (BD) para informatizar todo el sistema sanitario, con el objetivo de unificar la información necesaria para gestionar la actuación con los pacientes tanto en los hospitales como en las farmacias.

El sistema de gestión de bases datos que utilizaremos será Oracle 11*g* Express Edition. La explotación del almacén de datos la realizaremos con la herramienta OLAP (on-line analytical processing, o procesamiento analítico en línea) Pentaho.

1.2. Objetivos del PFC

Partimos del supuesto de que somos una empresa que es contratada por el ministerio de Sanidad, para analizar los requisitos de una nueva BD para informatizar todo el sistema sanitario; y para realizar posteriormente su implementación.

Actualmente la información de toda la gestión del sistema sanitario no se encuentra centralizada, por lo que la comunicación entre los distintos hospitales y las farmacias es de forma manual. Con la integración de esta información, se podrán registrar todas las acciones del sistema sanitario con cada uno de sus pacientes, incluyendo tanto la atención sanitaria por parte de los hospitales como los medicamentos expedidos por las farmacias.

Cuando un médico esté realizando una consulta a un paciente, la BD facilitará la información necesaria para realizar un mejor diagnóstico. Esta información será todo su historial médico; incluyendo las enfermedades que ha padecido, los médicos que le han visitado, los medicamentos que le han recetado y expedido con anterioridad; las pruebas médicas que le han realizado y las bajas laborales que le han prescrito.

La BD también servirá para analizar la situación actual de la gestión sanitaria. Para ello facilitará estadísticas como: información sobre las épocas del año y hospitales que tienen más consultas externas y de carácter urgente; edad en la que se consumen más medicamentos; o duración de las bajas y enfermedades que las producen.

Un factor crítico del proyecto es diseñar una BD lo más estándar posible para que pueda ser accedida por distintos sistemas externos, y que sea escalable ante nuevos requisitos en un futuro. Estos sistemas externos habrán sido desarrollados por distintas empresas e implementados en diferentes plataformas.

1.3. Enfoque y método seguido

La metodología de trabajo que se ha utilizado es la del ciclo de vida en cascada. Es la mejor forma de trabajar teniendo en cuenta que todas las tareas del proyecto las va a realizar la misma persona. De esta forma también se pueden planificar las tareas de una forma más sencilla y ajustando mejor el tiempo que vamos a invertir en cada una de ellas.

Con este método el proyecto se divide en varias fases y cada una de ellas empieza cuando finaliza la anterior. En cada etapa se obtienen unos entregables que son el punto de partida de la etapa siguiente. En este proyecto no paralelizamos ninguna tarea ya que, como hemos comentado, el equipo lo forma una única persona. Estas son las fases que hemos seguido:

-**Definición de objetivos.** Análisis previo en el que se definen los objetivos. A partir de este análisis, se planifica el proyecto.

-**Análisis de requisitos y viabilidad.** Recopilar y especificar los requisitos del cliente que deberá satisfacer la base de datos. Definición de los casos de uso.

-**Diseño de la base de datos.** Creación del modelo conceptual y del modelo lógico de la base de datos, a partir de los requisitos del cliente.

-**Implementación**. Creación del código necesario para crear la base de datos, a partir del diseño establecido en la etapa anterior. Empezamos la creación de las tablas y seguimos con la creación de los procedimientos almacenados para facilitar el acceso y modificación de la información.

-**Pruebas**. Se debe garantizar el correcto funcionamiento del sistema ante el mayor número de situaciones posibles. Para ello es muy importante diseñar y ejecutar un buen plan de pruebas.

-**Documentación**. Antes de la implantación y posterior mantenimiento, es importante documentar todo lo realizado, para que en un futuro sea más sencillo realizar correcciones o ampliaciones en el sistema creado.

Después de finalizar las pruebas de la base de datos, empezamos por el análisis de requisitos del almacén de datos; para seguir con las fases de diseño, implementación y pruebas del mismo. Una vez probado el DWH, creamos los informes con la herramienta Pentaho para su explotación.

Aunque la base de datos ha sido instalada y probada en varios sistemas antes de ser entregada, no hemos realizado las últimas fases del ciclo de vida en cascada: implantación y mantenimiento.

1.4. Planificación del trabajo

El trabajo se ha repartido a lo largo de los siete días de la semana. De Lunes a Viernes se dedicaron de 3 a 4 horas diarias, el Sábado 8 y el Domingo 4. Esta dedicación varió dependiendo de las tareas propias de la jornada laboral, o de las tareas a realizar en otra asignatura que he cursado. Cada parte del proyecto fue entregada en las fechas establecidas. Al finalizar la PAC3 hubo una ampliación del alcance del proyecto, se añadió la creación de análisis con la suite Pentaho para la explotación del almacén de datos.

En la planificación del trabajo no se incluyen reuniones de seguimiento y control entre el jefe de proyecto y el resto del equipo; ya que todas las tareas las realiza la misma persona; esto se sustituye por la tarea “Revisión planificación”. Como hemos comentando en el apartado anterior, tampoco hemos incluido intencionadamente las tareas de puesta en marcha y mantenimiento del producto.

1.4.1. Hitos significativos

Estas son las fechas más importantes del proyecto.

- Inicio del proyecto.	28 de Febrero de 2013
- Entrega PAC1. Plan de Proyecto.	17 de Marzo de 2013
- Entrega PAC2.	21 de Abril de 2013
- Entrega PAC3. BD finalizada.	19 de Mayo de 2013
- Entrega final. Memoria y presentación virtual.	12 de Junio de 2013

1.4.2. Estructuración del proyecto y distribución del trabajo.

Como hemos comentado en el apartado 1.3 "Enfoque y método seguido", la metodología utilizada ha sido la del ciclo de vida normal o en cascada, y esta ha sido la distribución de las tareas a lo largo del proyecto.

1. PAC 1. Análisis preliminar de los requisitos y creación del Plan de trabajo.

- 1.1 Lectura de la documentación.
- 1.2 Preparación del entorno de trabajo.
  - 1.2.1 Creación de la máquina virtual.
  - 1.2.2 Instalación de Oracle y SQL Developer.
- 1.3 Análisis inicial de requisitos.
- 1.4 Diseño conceptual preliminar.
- 1.5 Elaboración del plan de trabajo.
- 1.6 Entrega del plan de trabajo.

2. PAC 2. Creación de la BD y de los procedimientos de acceso a la información.

- 2.1 Revisión planificación.
- 2.2 Análisis de requisitos.
- 2.3 Diseño de la BD.
  - 2.3.1 Diseño conceptual.
  - 2.3.2 Diseño lógico.
- 2.4 Creación de la BD.
  - 2.4.1 Scripts de creación de tablas.
  - 2.4.2 Carga de datos de prueba.
  - 2.4.3 Pruebas unitarias.
- 2.5 Creación del sistema de monitorización.
  - 2.5.1 Creación de disparadores para alimentar la tabla de logs.
  - 2.5.2 Pruebas unitarias.
- 2.6 Desarrollo Procedimientos para acceso a los datos.
  - 2.6.1 Scripts de creación de los procedimientos almacenados.
  - 2.6.2 Pruebas unitarias.
- 2.7 Entrega PAC2.

3. PAC 3. Creación del DWH. BD finalizada.

- 3.1 Revisión planificación.
- 3.2 Análisis y diseño del DWH a desarrollar.
- 3.3 Creación de las tablas del DWH.
  - 3.3.1 Scripts de creación de tablas.
  - 3.3.2 Scripts de carga de tablas de dimensiones.
- 3.4 Creación de la carga del DWH.
  - 3.4.1 Scripts de carga de la tabla de estadísticas.
  - 3.4.2 Pruebas unitarias.
- 3.5 Pruebas finales de la BD.
- 3.6 Análisis de rendimiento.
- 3.7 Entrega PAC3, BD finalizada.

4. Entrega final. Memoria y presentación virtual.

4.1 Creación de los informes con la suite Pentaho.
4.2 Elaboración de la memoria final.
4.3 Elaboración presentación virtual.
4.4 Revisión final y corrección de errores.
4.5 Entrega del proyecto

En la figura 1 podemos ver el diagrama de Gantt con la planificación de todas las tareas:

	Task Name	Duration	Start
1	**1 PAC1 - Plan de Trabajo**	**17,67 days**	**Thu 28/02/13**
2	1.1 Lectura de la documentación	1 day	Thu 28/02/13
3	**1.2 Preparación del entorno de trabajo**	**1 day**	**Fri 01/03/13**
4	1.2.1 Creación de la máquina virtual	1 day	Fri 01/03/13
5	1.2.2 Instalación de Oracle y SQL Developer	0 days	Sat 02/03/13
6	1.3 Análisis inicial de requerimientos	2 days	Wed 06/03/13
7	1.4 Diseño conceptual preliminar	1 day	Sat 09/03/13
8	1.5 Elaboración del plan de trabajo	2 days	Fri 15/03/13
9	1.6 Entrega del plan de trabajo	0 days	Sun 17/03/13
10	**2 PAC2**	**35 days**	**Sun 17/03/13**
11	2.1 Revisión planificación	1 day	Sun 17/03/13
12	2.2 Análisis de requerimientos	2 days	Tue 19/03/13
13	**2.3 Diseño de la BD**	**4 days**	**Thu 21/03/13**
14	2.3.1 Diseño conceptual	2 days	Thu 21/03/13
15	2.3.2 Diseño lógico	2 days	Sat 23/03/13
16	**2.4 Creación de la BD**	**11 days**	**Mon 25/03/13**
17	2.4.1 Scripts de creación de tablas	8 days	Mon 25/03/13
18	2.4.2 Carga de datos de prueba	2 days	Tue 02/04/13
19	2.4.3 Pruebas unitarias	1 day	Thu 04/04/13
20	**2.5 Creación del sistema de monitorización**	**4 days**	**Fri 05/04/13**
21	2.5.1 Creación de disparadores para alimentar la tabla de logs	3 days	Fri 05/04/13
22	2.5.2 Pruebas unitarias	1 day	Mon 08/04/13
23	**2.6 Desarrollo Procedimientos para acceso a los datos**	**12 days**	**Tue 09/04/13**
24	2.6.1 Scripts de creación de los procedimientos almacenados	11 days	Tue 09/04/13
25	2.6.2 Pruebas unitarias	1 day	Sat 20/04/13
26	2.7 Entrega PAC2	0 days	Sun 21/04/13
27	**3 PAC3**	**27 days**	**Tue 23/04/13**
28	3.1 Revisión planificación	0 days	Tue 23/04/13
29	3.2 Análisis y diseño del DWH a desarrollar	8 days	Tue 23/04/13
30	**3.3 Creación de las tablas del DWH**	**5 days**	**Wed 01/05/13**
31	3.3.1 Scripts de creación de tablas	3 days	Wed 01/05/13
32	3.3.2 Scripts de carga de tablas de dimensiones	2 days	Sat 04/05/13
33	**3.4 Creación de la carga del DWH**	**7 days**	**Mon 06/05/13**
34	3.4.1 Scripts de carga de la tabla de estadísticas	5 days	Mon 06/05/13
35	3.4.2 Pruebas unitarias	2 days	Sat 11/05/13
36	3.5 Pruebas finales de la BD	4 days	Mon 13/05/13
37	3.6 Análisis de rendimiento	2 days	Fri 17/05/13
38	3.7 Entrega PAC3, BD finalizada	0 days	Sun 19/05/13
39	**4 Entrega final**	**25 days?**	**Sun 19/05/13**
40	4.1 Creación de los informes con la suite Pentaho	5 days	Sun 19/05/13
41	4.2 Elaboración de la memoria final	10 days	Fri 24/05/13
42	4.2 Elaboración presentación virtual	4 days	Mon 03/06/13
43	4.3 Revisión final y corrección de errores	5 days	Fri 07/06/13
44	4.4 Entrega del proyecto	0 days	Wed 12/06/13

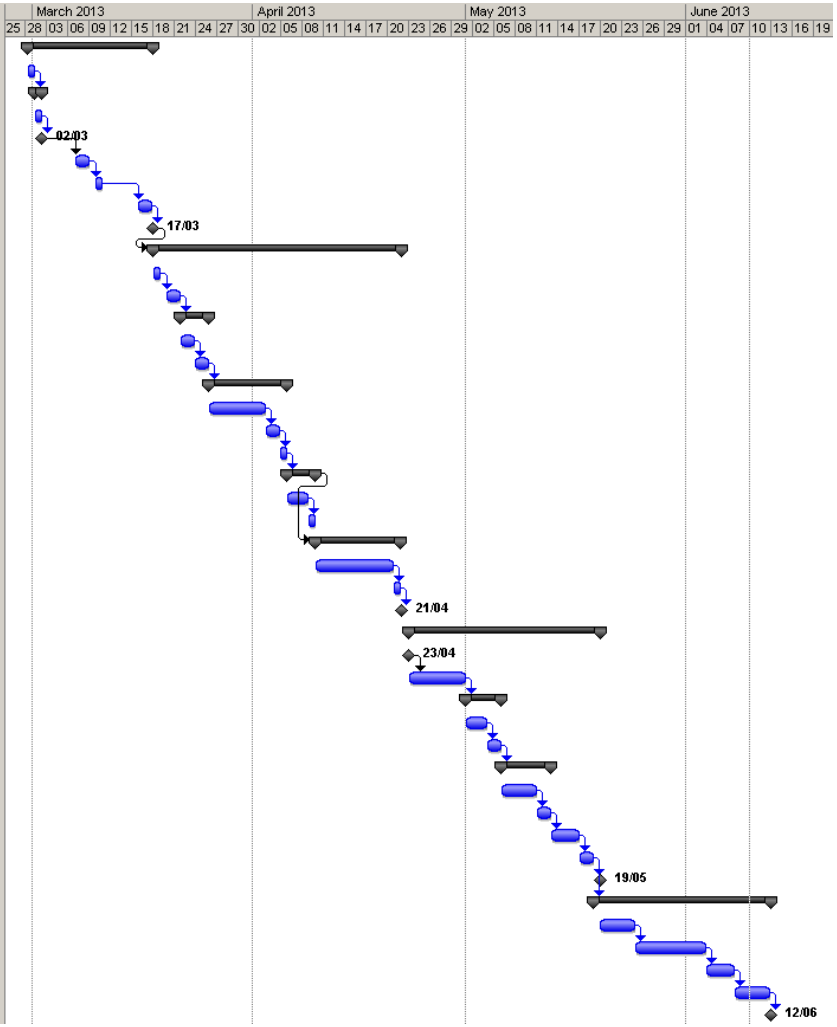

Figura 1. Diagrama de Gantt.

1.5. Productos obtenidos

Los productos obtenidos tras la finalización de este proyecto son los siguientes:

1. Plan de trabajo. Documento en el que se refleja el análisis preliminar del proyecto y una planificación del mismo fijando las fechas más importantes.

2. Producto. El producto comprende todos los ficheros que contienen el código necesario para la creación de la base de datos, el almacén de datos y los informes necesarios para su explotación. Incluye desde la creación de las tablas; con sus relaciones, secuencias y disparadores; los procedimientos almacenados; y los esquemas y análisis de la herramienta Pentaho.

3. Memoria. El presente documento donde reflejamos todo el trabajo realizado a lo largo de este proyecto.

4. Presentación virtual. Presentación de 20 diapositivas en las que se resume el proyecto realizado.

1.6. Descripción breve de los otros capítulos de la memoria

Hemos plasmado el desarrollo del proyecto en los siguientes apartados:

2. Análisis de requisitos. En este apartado analizamos detalladamente los requisitos establecidos por el cliente. A partir de estos, especificamos los distintos casos de uso e indicamos los distintos actores que intervienen.

3. Diseño de la base de datos. Aquí exponemos el diseño de la base de datos a partir de los requisitos analizados en el apartado anterior. El diseño se realiza en tres fases, diseño conceptual, lógico y físico.

En el diseño conceptual utilizaremos un modelo Entidad-relación (E/R) para organizar los requerimientos y especificaciones expuestos por el cliente.

En el diseño lógico adaptamos el diseño conceptual al tipo de tecnología que vamos a utilizar para implementar la base de datos.

En el diseño físico explicamos la implementación de la base de datos a partir del diseño lógico y teniendo en cuenta las características del SGBD elegido.

4. Almacén de datos. En este apartado se desarrolla el almacén datos que nos solicita el cliente. También realizamos este trabajo en tres fases, el diseño conceptual, el lógico y el físico.

En el diseño conceptual establecemos los requisitos del cliente y definimos los elementos de análisis: hechos, dimensiones, indicadores y celdas.

En el diseño lógico, a partir de los hechos y las dimensiones definidas en el diseño conceptual; definimos las tablas de hecho y las tablas de dimensiones.

En el diseño físico se expone la implementación de las tablas del almacén de datos.

Por último, en este apartado vemos cómo explotar el almacén de datos con la herramienta Pentaho.

5. Carga inicial y pruebas. Explicamos cómo hemos realizado la carga inicial de los datos de prueba; y las pruebas que hemos realizado para comprobar el correcto funcionamiento de la base de datos.

6. Conclusiones. Valoración personal de la ejecución del proyecto.

7. Glosario. Definición de algunos términos utilizados en el cuerpo de la memoria.

8. Bibliografía. Lista de las fuentes de información consultadas.

2. Análisis de requisitos

En este apartado analizaremos detalladamente los requisitos establecidos por nuestro cliente. La BD debe diseñarse para poder soportar la siguiente funcionalidad:

-Deberá registrar todas las consultas externas que se realicen en todos los hospitales, ya sean consultas normales o con carácter de urgencia, almacenando tanto los datos del paciente como del médico, así como las pruebas médicas y los medicamentos que son recetados.

-La gestión de los pacientes, además de las consultas externas que reciban en cualquier hospital, registrará las enfermedades que han padecido a lo largo de su historial médico, los médicos que les han atendido, las pruebas diagnósticas que les han realizado y las medicinas que les han sido recetadas, así como el médico que tienen asignado actualmente. También recogerá una relación de todas las bajas laborales, con su duración, la enfermedad que las provocó y el médico que concedió la baja.

-La gestión de los médicos registrará los hospitales en los que han sido destinados, los pacientes que han atendido, y los que tienen actualmente asignados.

-Sobre las farmacias, se registrarán todos los medicamentos con receta expedidos a los distintos pacientes.

También se debe desarrollar un *data warehouse* (DWH) para extraer estadísticas de una forma rápida, y así poder analizar la situación de la gestión sanitaria de forma global.

La BD deberá ser lo más escalable posible para permitir nuevos requisitos que pudieran surgir posteriormente a la finalización del proyecto.

Por último, la BD deberá disponer de un sistema de generación de *logs*, registrando todas las acciones que se realicen sobre ella. De esta forma se podrá realizar un mejor mantenimiento de la misma y facilitará la integración con el resto de sistemas externos.

El proyecto únicamente contempla el diseño e implementación de la BD, incluyendo todos los procedimientos almacenados necesarios para permitir un acceso a los datos a través de ellos y el registro de toda la actividad realizada sobre ella. No se permitirá ningún acceso a la información que no sea a través de dichos procedimientos.

No está incluido en el alcance del proyecto ningún desarrollo adicional para crear una interfaz de usuario mediante la cual consultar o mantener la información.

2.1. Lista detallada de requisitos

Esta es la lista de requisitos que extraemos a partir de lo expuesto por nuestro cliente:

1. La base de datos a diseñar deberá gestionar todo el sistema sanitario, desde los hospitales hasta las farmacias.

2. El acceso a la información se realizará a través de procedimientos almacenados, no pudiendo acceder directamente a las tablas para consultar o modificar la información.

3. La base de datos deberá gestionar tanto las consultas externas como las urgencias que se den en los hospitales.

4. La base de datos mantendrá información sobre enfermedades conocidas, y los medicamentos que se han prescrito para combatirlas.

5. Se registrarán todos aquellos productos comerciales o genéricos que existan de un mismo medicamento.

6. Soportará la gestión de las farmacias que están de guardia en cada población, así como los medicamentos que dispense cada una a los pacientes. También se almacenará información sobre los farmacéuticos que poseen las farmacias.

7. De cada médico, se mantiene la información de los hospitales a los que ha sido destinado, las consultas realizadas y los pacientes que tiene actualmente asignados. También se registran las consultas que haya solicitado para que un paciente visite otro médico especialista, las recetas que expidan y las revisiones de bajas que hagan.

8. De los hospitales, se gestionará las consultas que se realicen, ya sean consultas externas o urgencias; los tipos de pruebas diagnósticas que pueden realizar y las que realmente realicen.

9. De cada paciente tendremos una relación de las enfermedades contraídas, con los medicamentos recetados, las consultas médicas, las pruebas diagnósticas realizadas y el estado de las posibles bajas que le sean prescritas para cada una de dichas enfermedades.

2.2. Casos de uso

En este apartado identificaremos los distintos actores que intervendrán en el sistema y la definición de los distintos casos de uso.

En nuestro modelo reflejamos varios actores distintos. No tendremos un actor principal, ya que varios de ellos desempeñan tareas importantes en el sistema. Estos son los actores que intervienen:

Administrador. Será el administrador del sistema informático. Únicamente intervendrá en el mantenimiento de los tipos y estados de las consultas y destinos; y en el mantenimiento de los países y las poblaciones. Podrá ejecutar de forma manual los procesos de extracción de datos y carga del almacén de datos (ETL), o bien planificarlos.

Gestor Sanidad. Este actor puede ser cualquier trabajador del ministerio de Sanidad que dispone de acceso a información sensible. Estas tareas normalmente las harán usuarios de perfiles diferentes pero, para simplificar el modelo, los hemos reunido todos en un mismo actor. Será el encargado de mantener la información referente a enfermedades, medicamentos, hospitales, pruebas diagnósticas, médicos, especialidades y pacientes.

Gestor Colegio Farmacéutico. Corresponderá a personal del Colegio de Farmacéuticos encargado de mantener la información de las farmacias, las guardias y los farmacéuticos.

Enfermera. Personal sanitario que encargado de mantener la información de los pacientes y de las consultas. Podrá trabajar tanto en urgencias como en consultas externas.

Médico. Profesional que realizará las consultas, tanto las externas como las de urgencias. Es el responsable del diagnóstico de las enfermedades, de la receta de medicamentos y de solicitar las pruebas médicas. También es el encargado de prescribir las bajas laborales.

Usuario laboratorio. Será cualquier trabajador de los laboratorios dónde se preparan los resultados de las pruebas médicas.

Farmacéutico. Profesional de una Farmacia. Será el responsable de expedir los medicamentos a los pacientes, siempre y cuando éstos tengan una receta médica.

Paciente. Usuario del sistema sanitario que acude ante cualquier problema de salud.

En el modelo no contemplamos el caso de uso de autenticación de los usuarios del sistema, ya que suponemos que esta gestión la harán los distintos sistemas externos.

Debido al gran número de casos de uso, no hemos entrado al detalle de los mismos. Por ejemplo, en lugar de especificar los distintos casos de uso necesarios en la gestión de las especialidades y la asignación de estas a los médicos, únicamente definimos el caso de uso “Gestionar especialidades”.

En la figura 2 podemos ver el diagrama de casos de uso.

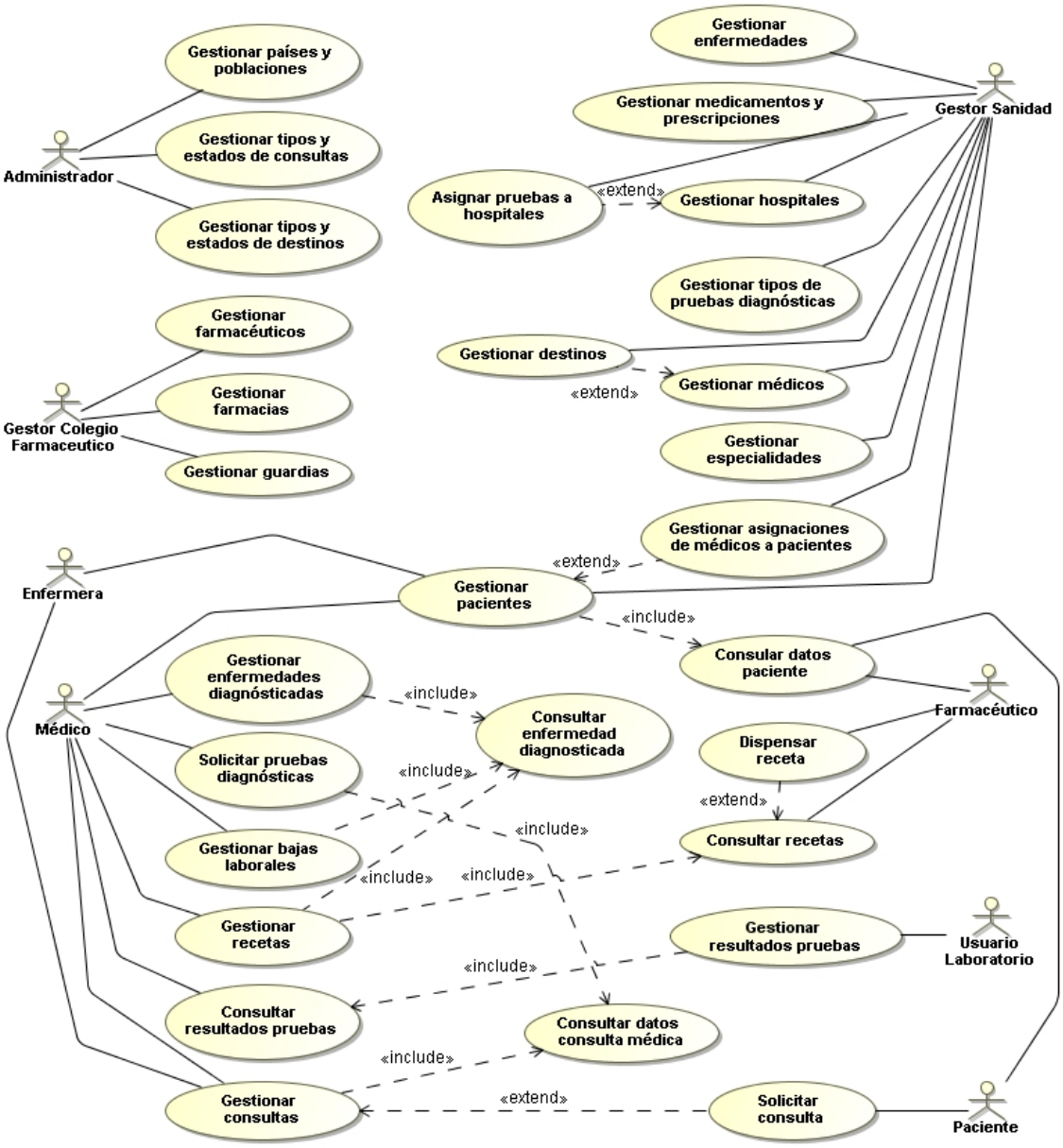

Figura 2. Diagrama de casos de uso.

El caso de uso “Asignar pruebas a hospitales” puede ser iniciado directamente por el Gestor de Sanidad, aunque también puede contemplarse como una extensión de “Gestionar hospitales”. De la misma forma, también extiende el caso de uso “Gestionar destinos” el de “Gestionar médicos”; o el caso de uso “Gestionar asignaciones de médicos a pacientes” extiende el de “Gestionar pacientes”.

Por otra parte, el caso de uso “Consultar datos paciente” está incluido dentro de “Gestionar pacientes”. Los actores Enfermera, Médico y Gestor Sanidad iniciarán directamente el segundo, pero los pacientes sólo podrán iniciar el primero. Pasa lo mismo para los casos “Consultar recetas”, “Consultar resultados pruebas” y “Solicitar consulta”.

A continuación especificaremos cada uno de los casos de uso.

1. Gestionar países y poblaciones.

Funcionalidad: mediante este caso de uso el administrador podrá dar de alta las poblaciones y los países a los que pertenecen. También podrá mantener las agrupaciones para poder crear provincias u otras zonas.

Flujo principal: para dar de alta los países el usuario indicará el nemotécnico y el nombre del país. Para dar de alta las poblaciones, además de indicar el nombre y código postal, tendrá que elegir el país al que pertenece la población.

2. Gestionar tipos y estados de consultas.

Funcionalidad: permite al administrador dar de alta los distintos tipos y estados de las consultas.

En principio una consulta puede ser de dos tipos: de urgencias o consulta externa. Pero en un futuro se podría necesitar algún tipo más. Sobre los estados, inicialmente una consulta podrá estar Planificada o Realizada.

3. Gestionar tipos y estados de destinos.

Funcionalidad: permite al administrador crear nuevos tipos y estados de destinos.

4. Gestionar farmacéuticos.

Funcionalidad: permitirá al usuario dar de alta a los farmacéuticos y modificar sus datos.

5. Gestionar farmacias.

Funcionalidad: permite al usuario dar de alta y modificar los datos de las farmacias, así como indicar a qué farmacéutico pertenece.

6. Gestionar guardias.

Funcionalidad: permitirá planificar las guardias de cada farmacia.

Flujo principal: el usuario elegirá la farmacia, la población y establecerá el inicio y el fin de la guardia.

7. Gestionar enfermedades.

Funcionalidad: permitirá al usuario dar de alta enfermedades nuevas conocidas, pudiendo indicar una descripción y los síntomas.

8. Gestionar medicamentos y prescripciones.

Funcionalidad: permitirá dar de alta medicamentos y los tipos de productos que haya en el mercado para cada medicamento.

Flujo principal: al dar de alta un medicamento, se podrá indicar para qué enfermedades ha sido prescrito y la posología idónea para cada una de ellas.

Para crear un producto, antes se ha debido crear el medicamento. Para el producto será necesario indicar el tipo, así como el nombre del mismo.

9. Gestionar hospitales.

Funcionalidad: permitirá dar de alta los hospitales y mantener la información de todos ellos.

10. Asignar pruebas a hospitales.

Funcionalidad: permitirá indicar qué pruebas médicas se pueden realizar en cada hospital. Para ello será necesario que tanto el hospital como la prueba médica se hayan creado previamente.

Flujo principal: podrá ejecutarse directamente o que se ejecute desde el caso de uso “Gestionar hospitales” al dar de alta un hospital.

11. Gestionar tipos de pruebas diagnósticas.

Funcionalidad: permitirá dar de alta las distintas pruebas diagnósticas conocidas, indicando un nombre, una descripción y las posibles contraindicaciones que pudiera tener

12. Gestionar médicos.

Funcionalidad: permitirá dar de alta y modificar la información de cada médico, así como indicar qué especialidades ha adquirido cada uno de ellos.

13. Gestionar destinos.

Funcionalidad: permitirá crear y modificar cada uno de los destinos que tenga cada médico.

Flujo principal: Podrá ejecutarse directamente o desde el caso de uso “Gestionar médicos”. Para crear un destino, deberá existir tanto el médico como el hospital.

14. Gestionar asignaciones de médicos a pacientes.

Funcionalidad: permitirá asignar a cada paciente su médico.

Flujo principal: podrá ejecutarse directamente o desde el caso de uso “Gestionar pacientes”. Para crear la asignación, previamente debe existir el médico y el paciente.

15. Gestionar especialidades.

Funcionalidad: permite dar de alta las distintas especialidades que existen en la medicina.

16. Gestionar pacientes.

Funcionalidad: permite dar de alta y modificar la información sobre un paciente. Incluye el caso de uso “Consultar datos paciente”.

17. Consultar datos paciente.

Funcionalidad: permitirá ver los datos de cualquier paciente de la base de datos.

18. Dispensar receta.

Funcionalidad: permite dar como dispensada una receta.

Flujo principal: el usuario podrá consultar las recetas que tiene dadas de alta un paciente (este caso de uso extiende el de “Consultar recetas”), y dispensar una de ellas. De esta forma un paciente no podrá adquirir de nuevo el mismo medicamento sin tener otra receta.

19. Consultar recetas.

Funcionalidad: permite consultar todas las recetas que tiene pendientes el paciente por parte de cualquier médico.

20. Gestionar resultados pruebas.

Funcionalidad: permitirá adjuntar los resultados a las pruebas médicas por parte del personal del laboratorio.

21. Solicitar consulta.

Funcionalidad: permite solicitar una consulta para un médico.

Flujo principal: a la hora de solicitar una consulta, le aparecerán al usuario aquellas franjas horarias que el médico, con que se quiere concertar la consulta, tiene libres. Podrá elegir una franja horaria y

confirmar la cita previa. En caso del actor ser el paciente, únicamente le aparecerá el médico que tenga asignado. En caso de ser otro médico o una enfermera el que solicite la consulta, podrá buscar de entre todos los médicos de la base de datos que estén activos.

22. Gestionar enfermedades diagnosticadas.

Funcionalidad: permite al médico diagnosticar una enfermedad y modificar la información de una enfermedad ya diagnosticada.

Flujo principal: el usuario buscará el paciente al que le está haciendo la consulta. Después realizará una búsqueda de entre las enfermedades conocidas y creará el diagnóstico.

23. Consultar enfermedad diagnosticada.

Funcionalidad: permite buscar las enfermedades diagnosticadas de un paciente.

24. Consultar datos consulta médica.

Funcionalidad: permite buscar las consultas que tiene cada paciente, ya sean planificadas o históricas.

25. Solicitar pruebas diagnósticas.

Funcionalidad: permite solicitar una prueba diagnóstica para un paciente.

Flujo principal: el usuario seleccionará la consulta que está realizando al paciente. A continuación buscará una de las pruebas diagnósticas conocidas y creará la solicitud de la prueba. A continuación verá los hospitales en los que se puede realizar la prueba, e indicará uno de ellos.

26. Gestionar bajas laborales.

Funcionalidad: crear una revisión para una baja laboral de un paciente.

Flujo principal: el médico seleccionará una enfermedad de las diagnosticadas al paciente. Creará un tipo de revisión que podrá ser de alta, prórroga o finalización. Dependiendo del tipo de revisión, el estado de la baja cambiará. Podrá añadir comentarios a los motivos de la revisión.

27. Gestionar recetas.

Funcionalidad: el usuario podrá recetar medicamentos a un paciente.

Flujo principal: el usuario seleccionará una de las enfermedades diagnosticadas a un paciente. Creará una receta indicando el producto y la posología.

28. Consultar resultados pruebas.

Funcionalidad: permite al usuario ver los resultados de las pruebas realizadas a un paciente.

29. Gestionar consultas.

Funcionalidad: dar de alta consultas, modificar la planificación y borrar una consulta planificada.

Flujo principal: el usuario podrá buscar un paciente y ver las consultas que se le han realizado y las que tiene planificadas. Podrá cambiar la planificación de la consulta seleccionada.

3. Diseño de la base de datos

En este apartado explicaremos el diseño de la base de datos a partir de los requisitos analizados en el apartado anterior. El diseño se realiza en tres fases: diseño conceptual, lógico y físico.

3.1. Modelo conceptual

A partir del análisis previo de requisitos, definimos la estructura que tendrá la base de datos. Este diseño es independiente de la tecnología que vayamos a utilizar para la implementación. Para la realización del diseño conceptual, nos ayudaremos del modelo semántico Entidad-Relación (E/R), para organizar los requerimientos y especificaciones del cliente que hemos visto en apartado 2 "Análisis de requisitos".

En la figura 3 podemos ver el modelo conceptual de la base de datos.

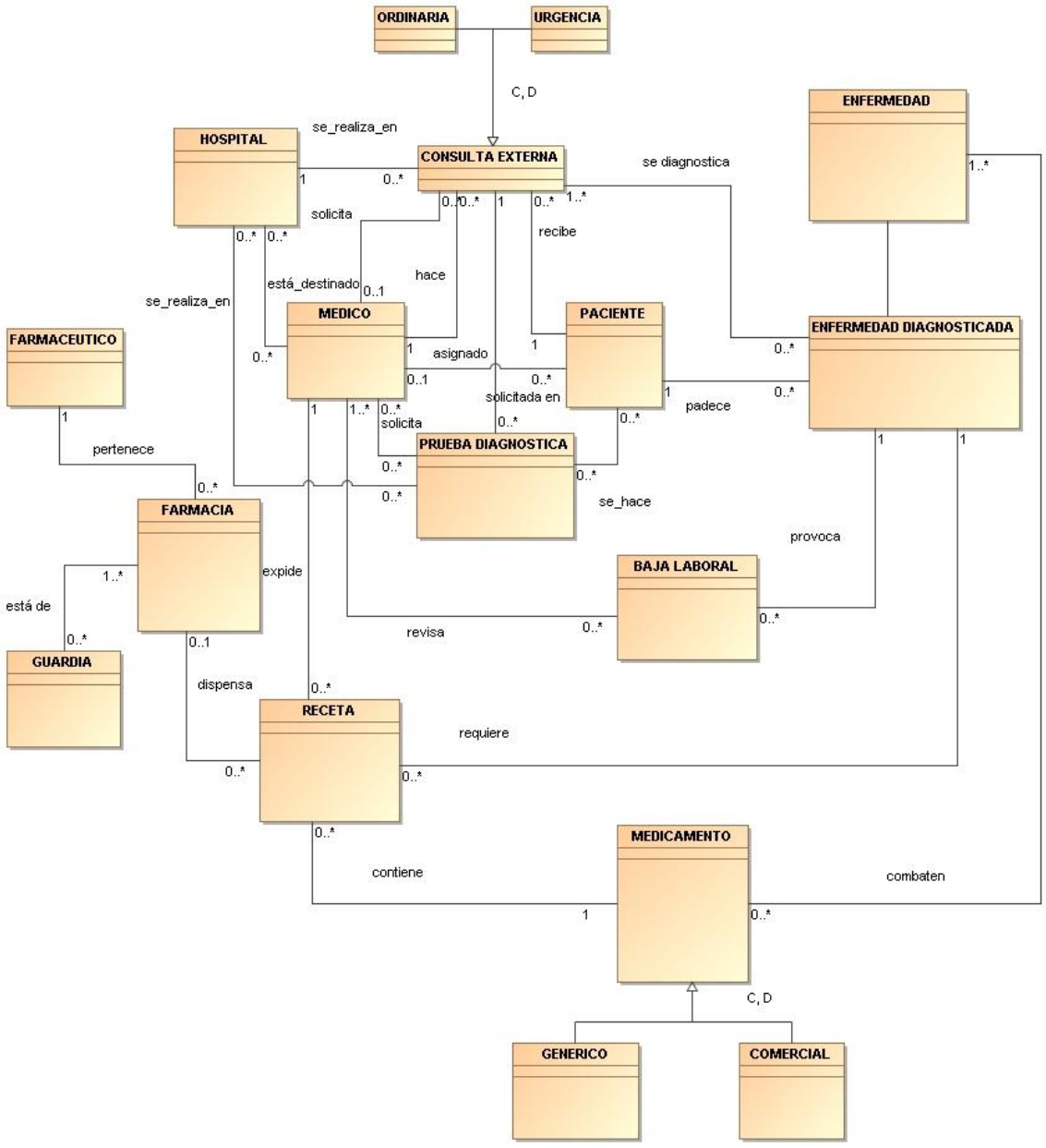

Figura 3. Modelo Conceptual.

A continuación, detallaremos las entidades destacadas en el diseño conceptual y las relaciones entre ellas:

CONSULTA

Reflejamos las consultas externas y las urgencias como una especialización de la entidad consulta. Es una generalización completa y disjunta.

La consulta está relacionada con el médico dos veces. Por una parte es quién la realiza y por otra puede ser que la consulta la haya solicitado otro médico. Una consulta la puede solicitar un paciente o bien que sea el médico de cabecera el que la solicita para un especialista.

Esta entidad también está relacionada con el paciente que solicita y recibe la consulta.

La consulta también está relacionada con la entidad PRUEBA DIAGNOSTICA, ya que sólo a través de una consulta se podrán solicitar dichas pruebas.

También está relacionada con la entidad ENFERMEDAD DIAGNOSTICADA. Tras una consulta se puede diagnosticar una enfermedad, varias o ninguna, de ahí la multiplicidad '0..*'. Una consulta antigua se podrá asignar posteriormente a una enfermedad que se diagnostique después.

PACIENTE

La entidad PACIENTE recoge los datos de los pacientes. Está relacionada con las consultas que ha solicitado y recibido, las enfermedades que le han diagnosticado, y con las pruebas médicas que le han hecho.

También está relacionado con el médico que tiene asignado.

Los medicamentos que le han sido recetados y las bajas que le han prescrito se relacionan a través de las enfermedades diagnosticadas.

MEDICO

La entidad MEDICO está relacionada con el hospital al que ha sido destinado, con las consultas externas que ha realizado y con los pacientes que tiene asignados. También se relaciona con los medicamentos que ha recetado, las pruebas diagnósticas que ha solicitado y con las bajas que ha prescrito o revisado.

ENFERMEDAD DIAGNOSTICADA

Esta entidad está relacionada con el paciente que la padece y con las consultas externas que ha tenido en relación a ella. También está relacionada con la entidad ENFERMEDAD, que mantiene una relación de las enfermedades conocidas.

Mantiene, además, relación con la entidad BAJA LABORAL, ya que sin una enfermedad diagnosticada un médico no puede prescribir una baja. Asimismo, está relacionada también con la entidad RECETA ya que, tal y como pasa con las bajas laborales, un médico no puede recetar sin diagnosticar antes una enfermedad.

HOSPITAL

Esta entidad mantiene la información de los distintos hospitales que se gestionan desde el Ministerio de Sanidad. Está relacionada con las consultas que se realizan en ellos; también con el tipo de pruebas que se pueden realizar; y con las consultas que se planifican y realizan. También se relaciona con los médicos destinados.

FARMACIA

Entidad que registra la información de las farmacias. Está relacionada con la entidad RECETA, que corresponde a las recetas creadas por los médicos; y con la entidad GUARDIA, que mantiene la relación de guardias que corresponden a cada farmacia.

FARMACEUTICO

Entidad referente a los farmacéuticos que están encargados de las distintas farmacias. Una farmacia puede pertenecer a un único farmacéutico y un farmacéutico puede no tener ninguna o tener varias farmacias.

ENFERMEDAD

Entidad que almacena todas las enfermedades conocidas. Está relacionada con la entidad ENFERMEDAD DIAGNOSTICADA y con la entidad MEDICAMENTO, que son los que la combaten.

MEDICAMENTO

Entidad que registra los medicamentos disponibles actualmente. Es una generalización completa y disjunta; de los medicamentos genéricos y de los comerciales. Está relacionada con la entidad RECETA.

RECETA

Entidad que corresponde a las recetas prescritas por los médicos para combatir las enfermedades contraídas por los pacientes. Está relacionada con la entidad ENFERMEDAD DIAGNOSTICADA, ya que como decimos sin ésta no se puede recetar. También lo está con el médico que la ha creado, así como con el medicamento que incluye. Cada receta corresponde a un único medicamento.

PRUEBA DIAGNÓSTICA

Relación que mantiene las pruebas diagnósticas que se ha hecho el paciente. Se relaciona con el médico que la pide y el paciente que se la hace. También se relaciona con la consulta en la que se solicita dicha prueba y con el hospital dónde se realiza. Como ya hemos comentado anteriormente, no es necesario diagnosticar una enfermedad para solicitar una prueba.

BAJA LABORAL

La entidad baja laboral está relacionada con el médico que la concede con una relación de muchos a muchos; ya que es posible que no sea el mismo médico el que la conceda, el que la revise y el que la finalice. También está relacionada con la enfermedad que la provoca.

3.2. Diseño lógico

En este apartado adaptaremos el diseño conceptual al tipo de tecnología que vamos a utilizar para implementar la base de datos. Oracle es el SGBD elegido, y es de tipo relacional, por lo que en el diseño lógico tendremos como resultado un modelo relacional.

Después de refinar el modelo conceptual y resolver las relaciones de muchos a muchos y otras particularidades del diseño conceptual, en la figura 4 podemos ver el diseño lógico de la base de datos. En él se pueden ver las distintas relaciones con sus atributos; indicando en cada una de ellas qué atributos son claves primarias, cuáles son claves ajenas y cuáles claves alternativas.

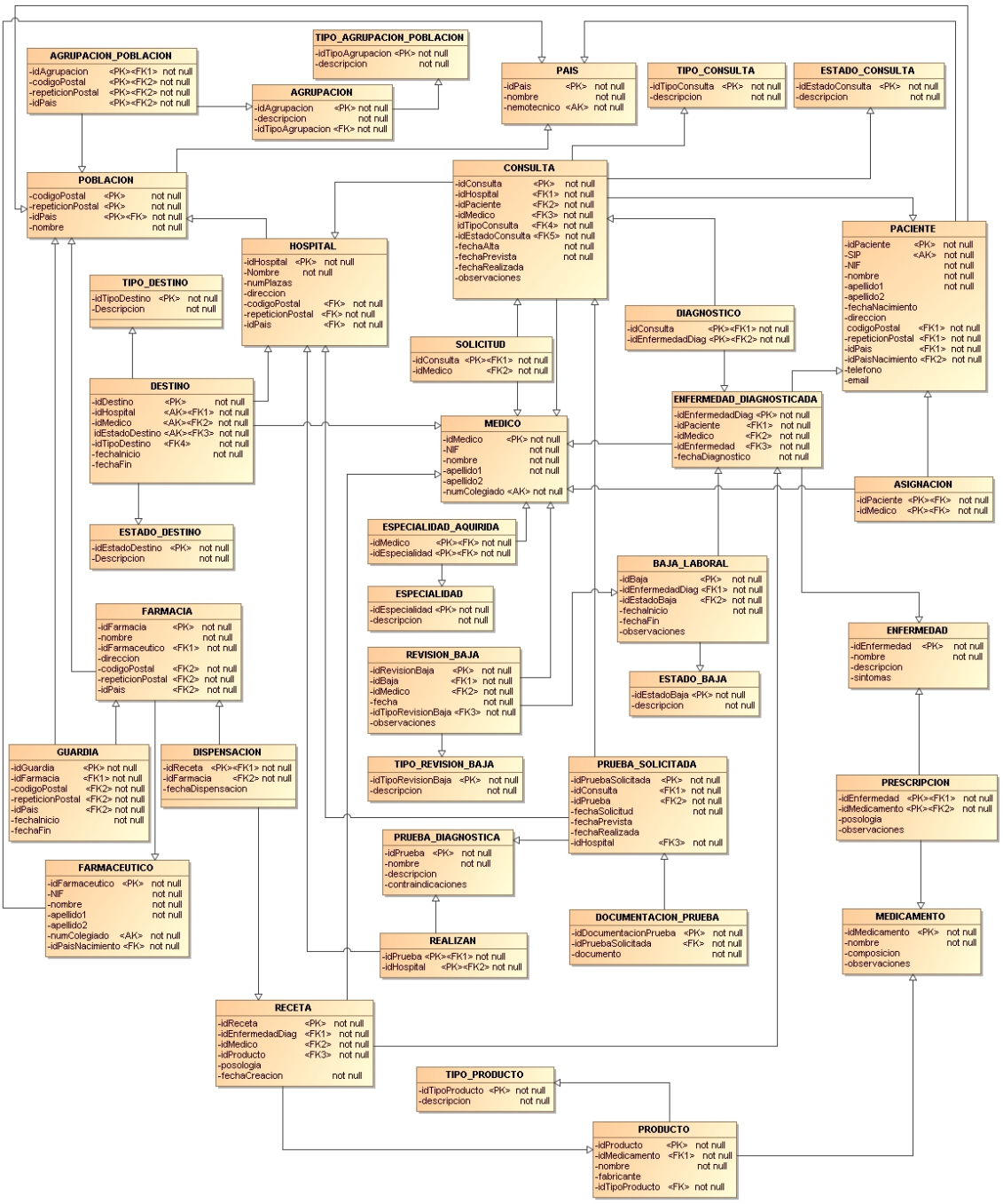

Figura 4. Diseño lógico.

3.2.1. Reglas de negocio

A continuación detallamos las reglas de negocio de nuestro modelo relacional, algunas de ellas no están explícitas en diagrama del diseño lógico.

1. Las consultas externas y las urgencias se gestionarán de la misma forma, distinguiendo únicamente el tipo de consulta.

2. Puede haber consultas en las que no se diagnostique ninguna enfermedad, pero sí en la que se soliciten pruebas diagnósticas. Por ello, se podrán solicitar pruebas médicas antes de diagnosticar una enfermedad.

3. Para una misma enfermedad diagnosticada, puede haber una o varias consultas médicas. Una consulta ya realizada, podrá relacionarse posteriormente con una enfermedad ya contraída para mantener la relación entre las pruebas realizadas y las enfermedades. Las consultas pueden ser realizadas por médicos distintos, se registrará la información de los médicos que ha visitado un paciente para cada enfermedad y qué médico realizó el primer diagnóstico (no tiene porqué coincidir con la primera consulta realizada para esta enfermedad).

4. Algunas pruebas médicas pueden realizarse en todos los hospitales y otras sólo en algunos. Además, las pruebas que se solicitan tienen un hospital asignado. El resultado de aquellas pruebas que se realicen en un hospital que no sea gestionado por el Ministerio de Sanidad, no se registrarán en el sistema.

5. No se puede conceder una baja sin tener una enfermedad diagnosticada. El médico que diagnostica la enfermedad puede no ser el mismo que prescriba la baja, la prorrogue o la finalice.

6. No se podrá recetar medicamentos sin que exista una enfermedad diagnosticada.

7. No se guarda la información de en qué consulta externa se prescribió cada receta o cada baja; sólo se mantiene qué médico la recetó o la revisó y sobre qué enfermedad diagnosticada están relacionadas.

8. Los pacientes sólo tendrán un médico asignado, que será el médico de cabecera. No se registra el histórico de médicos asignados a un paciente. El historial de médicos de un paciente se podrá obtener a partir de las consultas realizadas.

9. Un médico no tiene porqué estar asignado a un paciente para hacer una consulta médica, de hecho puede ser el médico de cabecera el que solicite una consulta para un especialista cuando lo crea conveniente.

10. Las consultas se darán de alta o bien al solicitarlo el paciente, al solicitarlo un médico de cabecera para un especialista. En caso de solicitarlo el médico de cabecera, se registra esta solicitud.

11. Un médico está destinado a uno o varios hospitales. Se recoge el histórico de destinos de un médico y el tipo; ya sea fijo, interinidad o sustitución.

12. Un medicamento puede estar prescrito para varias enfermedades, y una enfermedad puede tener distintos medicamentos que la combatan.

13. Una receta sólo contempla un producto. Este producto podrá ser un medicamento genérico o de una marca comercial.

14. Una farmacia sólo puede tener a un farmacéutico titular y un farmacéutico puede no tener ninguna farmacia.

15. En una misma población, pueden estar varias farmacias de guardia al mismo tiempo y una farmacia tener distintas guardias planificadas.

3.2.2. Descripción de las relaciones

En este apartado detallamos las relaciones obtenidas, indicando las decisiones de implementación.

PACIENTE

Podríamos haber creado una relación "AGENDA" para guardar los datos de los médicos, los farmacéuticos y los pacientes; ya que tanto los médicos como los farmacéuticos pueden ser también pacientes. No lo hemos planteado así por simplificar el modelo de datos.

La relación PACIENTE está enlazada por la relación ENFERMEDAD_DIAGNOSTICADA. Se han obtenido las mismas interrelaciones que en el modelo conceptual. A través de la relación ENFERMEDAD_DIAGNOSTICADA mantenemos la información referente a las bajas y recetas de cada paciente.

En el modelo conceptual la entidad PACIENTE estaba relacionada con la entidad PRUEBA_DIAGNOSTICA. En el modelo lógico hemos transformado esa relación de muchos a muchos enlazando a través de relación CONSULTA. Para que se solicite una prueba diagnóstica a un paciente debe existir una consulta de por medio.

La asignación de un médico a un paciente la mantenemos a través de la relación ASIGNACION. En esta relación guardamos el médico que tiene asignado cada paciente. Como hemos indicado en una de las reglas de negocio, no registramos el historial de médicos asignados a un paciente. Utilizamos una relación intermedia ya que es posible que un paciente no tenga ningún médico asignado.

ENFERMEDAD_DIAGNOSTICADA

La relación ENFERMEDAD DIAGNOSTICADA mantiene las interrelaciones con las relaciones BAJA_LABORAL Y RECETA. Se mantienen igual que en el modelo conceptual, ya que la multiplicidad de éstas lo permite.

Tenemos una clave ajena a la relación ENFERMEDAD que registra todas las enfermedades conocidas.

Para resolver la relación de muchos a muchos entre la enfermedad diagnosticada y la consulta, se ha creado la relación DIAGNOSTICO; y así mantener las consultas que ha tenido un paciente para una enfermedad en concreto.

DIAGNOSTICO

La relación DIAGNOSTICO relaciona una ENFERMEDAD_DIAGNOSTICADA con una CONSULTA. Se puede dar de alta una consulta inicialmente cuando se solicita una visita al médico. En ese caso todavía no habrá ningún diagnóstico, pero ya debemos poder registrar el paciente, el hospital y el médico que intervienen, así como la fecha prevista. Una vez se produzca el diagnóstico por parte del médico, éste podrá crear una nueva ENFERMEDAD_DIAGNOSTICADA o bien relacionar dicha consulta con una enfermedad ya existente. Una misma enfermedad diagnosticada puede tener varias consultas relacionadas a través de la relación DIAGNOSTICO. Una consulta puede no tener ninguna enfermedad diagnosticada relacionada.

BAJA_LABORAL, ESTADO_BAJA, REVISION_BAJA y TIPO_REVISION_BAJA

La relación entre la entidad BAJA_LABORAL y ENFERMEDAD_DIAGNOSTICADA se mantiene en el modelo lógico. Para resolver la multiplicidad entre la entidad BAJA_LABORAL y el MEDICO, creamos la relación REVISION_BAJA. De esta forma un médico puede dar de alta una baja y un médico distinto prorrogarla o finalizarla.

Con la relación TIPO_REVISION_BAJA mantenemos los posibles tipos de revisión: alta, prórroga y finalización; y con ESTADO_BAJA los estados por los que pasa una baja: ACTIVA o FINALIZADA.

RECETA

Cada receta se relaciona con una enfermedad contraída. En caso de tener dos enfermedades distintas y el médico hacer una receta para cada una de ellas en la misma consulta, se crean dos entradas en la tabla ENFERMEDAD_DIAGNOSTICADA y otras dos recetas en la tabla RECETAS.

Se mantienen las relaciones entre las entidades MEDICO y RECETA, ya que para una misma enfermedad contraída, distintos médicos pueden recetar medicamentos a lo largo del tiempo.

Una receta contiene sólo un medicamento, cuando un médico tiene que recetar varios medicamentos, se crea una receta para cada medicamento indicando la posología.

Una receta puede ser dispensada en una única farmacia, o no ser dispensada. Para resolver la relación de muchos a muchos del modelo conceptual creamos la relación DISPENSACION. Aquí registraremos la dispensación de los medicamentos por parte de las farmacias.

MEDICAMENTO y PRODUCTO

La especialización de MEDICAMENTO en GENERICO Y COMERCIAL, la resolvemos con dos relaciones: MEDICAMENTO Y PRODUCTO. En MEDICAMENTO se mantendrá la información de la composición del medicamento; y en PRODUCTO las características de cada uno de ellos, ya sea comercial o genérico. Esto lo hacemos así porque hay información que varía entre los distintos de productos de un mismo medicamento.

Los productos están relacionados con la receta y ésta con la enfermedad diagnosticada.

Los medicamentos también están relacionados con la enfermedad ya que estarán destinados a combatir alguna de ellas.

ENFERMEDAD y PRESCRIPCION

Un medicamento puede estar prescrito para una serie de enfermedades, y para una enfermedad puede haber más de un medicamento que la combata. Para resolver esta relación múltiple del modelo conceptual, creamos la relación PRESCRIPCION.

CONSULTA, PRUEBA_DIAGNOSTICA y PRUEBA_SOLICITADA

La generalización consulta externa y urgencia la resolvemos con la relación TIPO_CONSULTA, que indicará el tipo. El resto de datos a registrar en los dos tipos de consultas son los mismos.

Una consulta se creará al ser solicitada por un paciente a través de una cita previa, o por un médico cuando lo desvíe a un especialista. Para distinguir las consultas planificadas de las ya realizadas creamos la relación ESTADO_CONSULTA.

En el modelo conceptual tenemos una doble relación entre CONSULTA y MEDICO. Por una parte un médico hará una consulta y por otra un médico de cabecera puede solicitarla. Además como esta relación no siempre existirá (cuando la solicitud la haga directamente el paciente), creamos una nueva relación SOLICITUD para registrar esta información.

La relación entre la entidad CONSULTA y las pruebas diagnósticas solicitadas se mantiene, pero ahora hemos creado nuevas relaciones. Tenemos la relación PRUEBA_SOLICITADA, en las que se registrarán todas las pruebas que se soliciten, y la relación PRUEBA_DIAGNOSTICA, que mantendrá información sobre los distintos tipos de pruebas que existen. A través de la relación REALIZAN sabremos qué pruebas se pueden realizar en cada hospital. Por último tenemos la relación DOCUMENTACION_PRUEBA dónde registraremos los resultados de todas las pruebas realizadas. En principio almacenaremos únicamente la ruta del archivo.

MEDICO, ESPECIALIDAD y ESPECIALIDAD ADQUIRIDA

Además de la relaciones indicadas anteriormente (RECETA, BAJA LABORAL Y ENFERMEDAD_DIAGNOSTICADA), la entidad MEDICO tenía en el modelo conceptual una relación de muchos a muchos con la entidad HOSPITAL. En la transformación al modelo lógico hemos creado una nueva interrelación: DESTINO. De esta forma registraremos los distintos destinos que tenga el médico. Como un médico puede estar destinado a varios hospitales al mismo tiempo y queremos registrar tanto los que tiene actualmente como los que tuvo en el pasado, creamos dos nuevas relaciones. La primera de ellas es TIPO_DESTINO; un destino podrá ser de plaza fija, interinidad o una sustitución. La otra relación es ESTADO_DESTINO, donde indicaremos si el destino está vigente o corresponde al pasado.

Con el modelo de datos definido, un médico puede estar destinado a un hospital, y hacer una consulta externa o atender una urgencia en otro distinto.

Además, para una misma enfermedad de un paciente, un médico puede diagnosticar la enfermedad, otro solicitar las pruebas, otro recetar (o repetir receta) y otro distinto prescribir la baja o revisarla.

En la relación ESPECIALIDAD se registran las distintas especialidades médicas y en ESPECIALIDAD_ADQUIRIDA, las que tiene cada médico.

HOSPITAL

La relación HOSPITAL mantiene la interrelación con CONSULTA tal y como la teníamos en el modelo conceptual.

Como ya hemos comentado, en el modelo teníamos una relación de muchos a muchos con la entidad PRUEBA_DIAGNOSTICA. Ahora la interrelación es con las nuevas relaciones PRUEBA_SOLICITADA Y PRUEBA_DIAGNOSTICA. La primera indica las pruebas que se van a realizar o se han realizado en cada hospital y se interrelaciona directamente con la relación HOSPITAL. La segunda indica qué pruebas se pueden hacer en cada hospital que, al ser una relación de muchos a muchos, se resuelve a través de la relación REALIZAN.

FARMACIA

La relación FARMACIA mantiene la interrelación con la relación FARMACEUTICO.

En el caso de la interrelación con la relación RECETA, para poder modelar que una receta puede no estar dispensada por ninguna farmacia (cuando el paciente aún ha ido a la farmacia a comprar los medicamentos, o directamente nunca los compra), hemos creado la relación DISPENSACIÓN. Esta relación mantendrá la información de en qué farmacia es dispensado el medicamento de cada receta.

En el modelo lógico teníamos una relación de muchos a muchos con la entidad GUARDIA. Es posible que queramos planificar distintas guardias con distintas fechas para una misma farmacia, o que una farmacia no tenga ninguna guardia planificada.

PAIS, POBLACION, AGRUPACION, AGRUPACION_POBLACION Y TIPO_AGRUPACION_POBLACION

Aunque en el modelo conceptual no aparecían, hemos creado cuatro relaciones para mantener la información de los códigos postales.

Por parte tenemos la relación PAIS con información de todos los países. Aunque vamos a trabajar únicamente con el sistema sanitario español, es preferible contemplar destinos fuera de nuestro país; ya que debemos tener en cuenta que habrá pacientes, médicos y farmacéuticos que no sean españoles.

La siguiente relación es POBLACION. Una población está identificada por el código postal y el país en el que se encuentra. Hay poblaciones que comparten código postal, de ahí que hayamos incluido un campo en la clave primaria llamado repeticionPostal. Con el código postal, el país y la repetición postal identificamos unívocamente una población. Por eso en las relaciones en las que debemos registrar información de una localización (HOSPITAL, PACIENTE, FARMACIA y GUARDIA) tenemos claves ajenas a la clave primaria de la relación POBLACION.

Por otra parte tenemos la relación AGRUPACIÓN. Esta relación mantiene las agrupaciones que podemos tener. Por ejemplo, podemos tener todas las provincias de España. En este caso cada uno de estos registros sería del tipo “Provincia”. El tipo de la agrupación lo recogemos en la relación TIPO_AGRUPACION_POBLACION.

En la relación AGRUPACION_POBLACION mantenemos qué poblaciones están relacionadas con qué agrupaciones. Es decir, aquí registraríamos qué poblaciones pertenecen a la misma provincia.

Esto será útil a la hora de mostrar una dirección, indicando la provincia a la que pertenece la población; o bien para sacar estadísticas agrupando por provincia. Lo hemos modelado así ya que, a través de las relaciones AGRUPACIÓN y AGRUPACION_POBLACION, en un futuro se podrían crear otro tipo de agrupaciones de poblaciones por zonas que no se correspondieran a divisiones geográficas como las provincias. Creando un nuevo tipo de agrupación e insertando los registros correspondientes a las zonas tendremos todas las clasificaciones que necesitemos.

3.3. Diseño físico

En este apartado explicamos el diseño físico de nuestra base de datos. A partir del diseño lógico definido en el apartado anterior y teniendo en cuenta las características del SGBD elegido, creamos el diseño físico de la base de datos.

El acceso a la información de la base de datos, por parte de los sistemas externos, se realizará únicamente a través de procedimientos almacenados. Hemos creado los distintos procedimientos para dar de alta la información, modificarla y borrarla.

Todos los procedimientos se hay implementado de forma que admiten varios parámetros de entrada y dos salida. Los dos parámetros de salida corresponden al código y al mensaje de error que se pudiera producir tras la ejecución del procedimiento.

Se han creado procedimientos de alta de registros para todas las tablas de la base de datos. No ha sido así en el caso de los de modificación, ya que para algunas de ellas hemos creído conveniente disponer únicamente de procedimientos de baja.

En los procedimientos comprobamos si la restricción de nulos se cumple en los parámetros de entrada, y mostramos el error en caso de cumplirse; para el resto de restricciones recogemos la excepción y mostramos la que nos dará el Oracle a partir de las reglas de integridad que hemos establecido en el diseño de la BD. Sólo en el caso de la gestión de las revisiones y de las bajas, la integridad de la información se controla desde el procedimiento. También se realiza esta comprobación con las pruebas diagnósticas que se pueden hacer en cada hospital, de forma que no se permita planificar una prueba en un hospital en el que no se realiza dicha prueba.

En los procedimientos de modificación y baja, creamos el procedimiento de forma que se debe identificar el elemento que queremos modificar a través de la clave primaria. Para consultar la clave primaria de un elemento, tendremos los procedimientos de selección.

Se ha implementado un sistema *logs* mediante el cual se registran todas las acciones que se realizan contra la base de datos; incluyendo los parámetros con los que se ejecuta el procedimiento y el resultado del mismo. Una opción de mejora sería, en los procedimientos de modificación y de baja, guardar en el registro de *logs* la información que había antes de modificarlo o darla de baja. Esto ampliaría las posibilidades de monitorización de la base datos y ayudaría a resolver problemas que provoquen los sistemas externos.

3.3.1. Errores en la ejecución de los procedimientos

Como decíamos anteriormente, en los procedimientos se realiza una pequeña comprobación de errores en los parámetros utilizados en la invocación de los mismos. La mayor parte de la seguridad recae en la gestión hecha por el propio Oracle, a través de las reglas de integridad indicadas en la creación de las distintas tablas, pero controlamos algunas situaciones erróneas. En caso de ocurrir un error, generamos una excepción indicando el código del error y una descripción del mismo. En la descripción incluimos el elemento que provocó el error.

Esta es una relación de los errores que controlamos en los procedimientos y que no corresponden a la salida estándar de Oracle. De cada uno de ellos indicamos el código de error y la descripción.

-30000 Campo nulo que no acepta nulos.

-30001 El paciente ya tiene una baja activa.

-30002 No existen datos con los parámetros indicados.

-30003 No se puede dar de alta una revisión de tipo ALTA, para hacerlo debe utilizar el procedimiento ALTA_BAJA_LABORAL.

-30004 No se puede realizar la prueba diagnóstica en el hospital indicado.

-30005 No existe la tabla indicada.

-30006 No existe el campo en la tabla indicada.

-30007Error al cargar las dimensiones del DWH. Este código error se utiliza en los procedimientos creados para la carga del almacén de datos.

3.3.2. Creación de las tablas

En la creación de las tablas hemos reflejado las relaciones definidas en el modelo lógico, utilizando las restricciones de integridad de las que dispone Oracle. De esta forma en la creación de la tabla especificamos las claves primarias, las claves alternativas y las claves ajenas. También se indican qué campos no pueden contener nulos. Además, hemos desarrollado el código necesario para la creación de las secuencias y los disparadores que se encargarán de mantener los identificadores de las tablas.

No incluimos en la memoria todo el código desarrollado para su implementación, se puede consultar en el fichero SENTENCIAS CREACIÓN TABLAS.SQL suministrado en la entrega del Proyecto, ubicado en la carpeta *"Producto\1. Diseño fisico Base de datos"*. Las sentencias de creación de todas las tablas se encuentran en este fichero.

3.3.3. Procedimientos de alta

Como hemos explicado en el apartado 3.3 "Diseño Físico" hemos creado un procedimiento de alta para insertar información en cada una de las tablas de nuestra base de datos. Queda como una opción de mejora el crear procedimientos que den de alta información en más una tabla a la vez, cosa facilitaría el desarrollo necesario en los sistemas externos para la gestión de la base de datos. Esto sólo lo hemos hecho para el caso de las revisiones de las bajas, debido a su complejidad.

Cada procedimiento se ha implementado en un fichero distinto. No incluimos el código en la memoria, éstos se pueden ver en la carpeta *"Producto\1. Diseño fisico Base de datos\Procedimientos almacenados\ALTA\"*

Como también hemos comentado en el apartado 3.3, en la mayoría de los procedimientos de alta únicamente controlaremos que no sean nulos los parámetros obligatorios. El resto de restricciones las mantendremos a partir de las claves primarias, alternativas y ajenas establecidas en las tablas de la BD. Si en algún procedimiento hemos incluido alguna comprobación adicional, la especificamos en la definición del mismo.

En cada procedimiento se realiza la inserción de un la tabla de LOG para poder realizar un mantenimiento de la base de datos y controlar los errores que pudieran surgir. La inserción se realiza tanto si la ejecución ha terminado correctamente como si ha fallado.

A continuación incluimos la definición de cada uno de los procedimientos de alta, indicando para cada uno de ellos: una descripción, los parámetros de entrada y salida, y las comprobaciones que se realizan.

ALTA	SP_ALTA_PAIS
Descripción	Procedimiento que da de alta un país en la tabla PAIS.
Parámetros de entrada	p_nombre, nombre del país. p_nemotecnico, nemotécnico del país.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los campos p_nombre y p_nemotecnico no sean nulos.

ALTA	**SP_ALTA_POBLACION**
Descripción	Procedimiento que da de alta una población en la tabla POBLACION.
Parámetros de entrada	p_codigoPostal, código postal de la población p_repeticionPostal, la repetición postal, en caso de que exista ya otra población con el mismo código postal. p_idPais, identificador del país. p_nombre, nombre de la población.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada no sean nulos.

ALTA	**SP_ALTA_TIP_AGR_POB**
Descripción	Procedimiento que da de alta un tipo agrupación en la tabla TIPO_AGRUPACION_POBLACION.
Parámetros de entrada	p_descripcion, descripción de la agrupación.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada no sea nulo.

ALTA	**SP_ALTA_AGRUPACION**
Descripción	Procedimiento que da de alta una agrupación en la tabla AGRUPACION.
Parámetros de entrada	p_descripcion, descripción de la agrupación. p_idTipoAgrupacion, tipo de agrupación.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada no sean nulos.

ALTA	**SP_ALTA_AGRUPACION_POBLACION**
Descripción	Procedimiento que da de alta un registro en la tabla AGRUPACION_POBLACION para agrupar las distintas poblaciones en una misma agrupación.
Parámetros de entrada	p_idAgrupacion, identificador de la agrupación. p_codigoPostal, p_repeticionPostal y p_idPais. Campos de la clave primaria de la población.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada no sean nulos.

ALTA	**SP_ALTA_PACIENTE**
Descripción	Procedimiento que da de alta un paciente en la tabla PACIENTE.
Parámetros de entrada	p_SIP, Código SIP del paciente. p_NIF, p_nombre, p_apellido1, p_apellido2, p_fechaNacimiento, p_direccion, p_codigoPostal, p_repeticionPostal, p_idPais, campos para indicar el domicilio del paciente. p_idPaisNacimiento, p_telefono, p_email,
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_SIP, p_NIF, p_nombre, p_apellido1, p_codigoPostal, p_repeticionPostal, p_idPais y p_idPaisNacimiento, no sean nulos.

ALTA	**SP_ALTA_MEDICO**
Descripción	Procedimiento que da de alta un médico en la tabla MEDICO.
Parámetros de entrada	p_NIF, p_nombre, p_apellido1, p_apellido2, p_numColegiado, p_idPaisNacimiento
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_NIF, p_nombre, p_apellido1, p_numColegiado y p_idPaisNacimiento no sean nulos.

ALTA	**SP_ALTA_ASIGNACION**
Descripción	Procedimiento que da de alta un registro en la tabla ASIGNACION para reflejar la asignación de un médico a un paciente.
Parámetros de entrada	p_idPaciente, identificador del paciente. p_idMedico, identificador del médico.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada no sean nulos.

ALTA	**SP_ALTA_HOSPITAL**
Descripción	Procedimiento que da de alta un hospital en la tabla HOSPITAL.
Parámetros de entrada	p_nombre, p_numPlazas, número de plazas disponibles en el hospital. p_direccion, p_codigoPostal, p_repeticionPostal, p_idPais.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_nombre, p_codigoPostal, p_repeticionPostal, y p_idPais no sean nulos.

ALTA	**SP_ALTA_TIPO_DESTINO**
Descripción	Procedimiento que da de alta un tipo de destino en la tabla TIPO_DESTINO.
Parámetros de entrada	p_descripcion, descripción del tipo de destino.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada no sea nulo.

ALTA	**SP_ALTA_ESTADO_DESTINO**
Descripción	Procedimiento que da de alta un estado de destino en la tabla ESTADO_DESTINO.
Parámetros de entrada	p_descripcion, descripción del estado de destino.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada no sea nulo.

ALTA	**SP_ALTA_GUARDIA**
Descripción	Procedimiento que da de alta una guardia en la tabla GUARDIA. Para ello se pasa como parámetro el id de la farmacia, la población, y las fechas de inicio y fin de la guardia.

Parámetros de entrada	p_idFarmacia, p_codigoPostal, p_repeticionPostal, p_idPais, p_fechaInicio, p_fechaFin.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros p_idFarmacia, p_codigoPostal, p_repeticionPostal, p_idPais y p_fechaInicio no sean nulos.

ALTA	**SP_ALTA_FARMACIA**
Descripción	Procedimiento que da de alta una farmacia en la tabla FARMACIA.
Parámetros de entrada	p_nombre, nombre de la farmacia. p_idFarmaceutico, farmacéutico titular de la farmacia. p_direccion, p_codigoPostal, p_repeticionPostal, p_idPais, ubicación de la farmacia.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros p_nombre, p_idFarmaceutico, p_codigoPostal, p_repeticionPostal y p_idPais, no sean nulos.

ALTA	**SP_ALTA_DESTINO**
Descripción	Procedimiento que da de alta un destino en la tabla DESTINO. Es necesario pasar como parámetro el identificador del hospital, el del médico el tipo y estado del destino, y la fecha de inicio. La fecha de finalización es opcional.
Parámetros de entrada	p_idHospital, p_idMedico, p_idEstadoDestino, p_idTipoDestino, p_fechaInicio, p_fechaFin.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_idHospital, p_idMedico, p_idEstadoDestino, p_idTipoDestino y p_fechaInicio no sean nulos.

ALTA	**SP_ALTA_FARMACEUTICO**
Descripción	Procedimiento que da de alta un farmacéutico en la tabla FARMACEUTICO.

Parámetros de entrada	p_NIF, p_nombre, p_apellido1, p_apellido2, p_numColegiado, p_idPaisNacimiento.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_NIF, p_nombre, p_apellido1, p_numColegiado, y p_idPaisNacimiento, no sean nulos.

ALTA	**SP_ALTA_MEDICAMENTO**
Descripción	Procedimiento que da de alta un medicamento en la tabla MEDICAMENTO.
Parámetros de entrada	p_nombre, p_composicion, p_observaciones.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada p_nombre no sea nulo.

ALTA	**SP_ALTA_TIPO_PRODUCTO**
Descripción	Procedimiento que da de alta un tipo de producto en la tabla TIPO_PRODUCTO.
Parámetros de entrada	p_descripcion, descripción del tipo de producto.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada no sea nulo.

ALTA	**SP_ALTA_PRODUCTO**
Descripción	Procedimiento que da de alta un producto en la tabla PRODUCTO.
Parámetros de entrada	p_idMedicamento, medicamento al que corresponde el producto. p_nombre, nombre del producto p_fabricante, p_idTipoProducto.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_idMedicamento, p_nombre y p_idTipoProducto no sean nulos.

ALTA	SP_ALTA_ENFERMEDAD
Descripción	Procedimiento que da de alta una enfermedad conocida en la tabla ENFERMEDAD.
Parámetros de entrada	p_nombre, p_descripcion, p_sintomas.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada p_nombre no sea nulo.

ALTA	SP_ALTA_PRESCRIPCION
Descripción	Procedimiento que da de alta una prescripción en la tabla PRESCRIPCION. Para una enfermedad conocida se pueden prescribir varios medicamentos. Para cada prescripción se puede indicar un medicamento, una posología y unas observaciones.
Parámetros de entrada	p_idEnfermedad, p_idMedicamento, p_posología, p_observaciones.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_idEnfermedad y p_idMedicamento no sean nulos.

ALTA	SP_ALTA_ENF_DIAG
Descripción	Procedimiento que da de alta una enfermedad diagnosticada en la tabla ENFERMEDAD_DIAGNOSTICADA. Cuando un médico diagnostica una enfermedad, se crea un registro en esta tabla, para ello hay que pasar como parámetro el identificador del paciente, el del médico, el de la enfermedad y la fecha de diagnóstico.
Parámetros de entrada	p_idPaciente, p_idMedico, p_idEnfermedad, p_fechaDiagnostico.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ningún parámetro de entrada no sea nulo.

ALTA	SP_ALTA_RECETA
Descripción	Procedimiento que da de alta una receta en la tabla RECETA. En cada receta se incluye un único medicamento. Cada receta se realiza para combatir una enfermedad.

Parámetros de entrada	p_idEnfermedadDiag, enfermedad para la que se receta. p_idMedico, médico que hace la receta. p_idProducto, producto que se receta. p_posologia, posología recomendada. p_fechaCreacion, fecha de creación de la receta..
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_idEnfermedadDiag, p_idMedico, p_idProducto, p_fechaCreacion no sean nulos.

ALTA	**SP_ALTA_DISPENSACION**
Descripción	Procedimiento que da de alta una dispensación en la tabla DISPENSACIÓN. Esto se realiza cuando el farmacéutico dispensa el medicamento de una receta.
Parámetros de entrada	p_idReceta, receta que se dispensa. p_idFarmacia, farmacia dónde se dispensa. p_fechaDispensacion.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_idReceta y p_idFarmacia no sean nulos.

ALTA	**SP_ALTA_CONSULTA**
Descripción	Procedimiento que da de alta una consulta médica en la tabla CONSULTA.
Parámetros de entrada	p_idHospital, hospital donde se hace la consulta. p_idPaciente, p_idMedico, p_idTipoConsulta, p_idEstadoConsulta, p_fechaAlta, creación de la consulta. p_fechaPrevista, fecha que la consulta está planificada. p_fechaRealizada, fecha en la que finalmente se realiza la consulta. Normalmente será nulo. p_observaciones.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que los parámetros de entrada p_idHospital, p_idPaciente, p_idMedico, p_idTipoConsulta, p_idEstadoConsulta, p_fechaAlta y p_fechaPrevista no sean nulos.
ALTA	**SP_ALTA_TIPO_CONSULTA**

Descripción	Procedimiento que da de alta un tipo de consulta en la tabla TIPO_CONSULTA. Se utilizará cuando sea necesario disponer de un tipo nuevo de consulta. Inicialmente tendremos dos tipos: Consulta externa y Urgencia.
Parámetros de entrada	p_descripcion, descripción del tipo de consulta.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada p_descripcion no sea nulo.

ALTA	**SP_ALTA_ESTADO_CONSULTA**
Descripción	Procedimiento que da de alta un estado de consulta en la tabla ESTADO_CONSULTA. Inicialmente tendremos dos estados: Planificada y Realizada.
Parámetros de entrada	p_descripcion, descripción del estado de la consulta.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada p_descripcion no sea nulo.

ALTA	**SP_ALTA_DIAGNOSTICO**
Descripción	Procedimiento que da de alta un diagnóstico en la tabla DIAGNOSTICO. Las enfermedades diagnosticadas se relacionan con el paciente a través de las consultas médicas.
Parámetros de entrada	p_idConsulta, en la que se diagnostica una enfermedad. p_idEnfermedadDiag, enfermedad diagnosticada.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada sea nulo.

ALTA	**SP_ALTA_SOLICITUD**
Descripción	Procedimiento que da de alta una solicitud en la tabla SOLICITUD. Cuando se un médico el que se solicita una consulta para un médico especialista, además de ejecutar el procedimiento de dar de alta una consulta, a continuación hay que ejecutar este para reflejar que la consulta la ha solicitado un médico.
Parámetros de entrada	p_idConsulta, identificador de la consulta que solicita el médico para un especialista. p_idMedico, médico que solicita la consulta.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.

Comprobaciones	Que ninguno de los parámetros de entrada sea nulo.

ALTA	**SP_ALTA_ESPECIALIDAD**
Descripción	Procedimiento que da de alta una especialidad en la tabla ESPECIALIDAD.
Parámetros de entrada	p_descripcion, nombre de la especialidad.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada no sea nulo.

ALTA	**SP_ALTA_ESP_ADQ**
Descripción	Procedimiento que da de alta una especialidad adquirida en la tabla ESPECIALIDAD_ADQUIRIDA. De esta forma se registrarán cada una de las especialidades adquiridas por cada médico.
Parámetros de entrada	p_idMedico, p_idEspecialidad.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada sea nulo.

ALTA	**SP_ALTA_TIPO_REVISION_BAJA**
Descripción	Procedimiento que da de alta un tipo de revisión de baja en la tabla TIPO_REVISION_BAJA. Inicialmente tendremos tres tipos: Alta, Prorroga y Finalización.
Parámetros de entrada	p_descripcion, descripción del tipo de revisión.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada no sea nulo.

ALTA	**SP_ALTA_ESTADO_BAJA**
Descripción	Procedimiento que da de alta un estado de baja en la tabla ESTADO_BAJA. Inicialmente tendremos dos estados: Activa y Finalizada.
Parámetros de entrada	p_descripcion, descripción del estado de la baja.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada no sea nulo.

ALTA	**SP_ALTA_BAJA_LABORAL**

Descripción	Procedimiento que se ejecuta cuando un médico concede una baja laboral. Da de alta un registro en la tabla BAJA_LABORAL y otro en la tabla REVISION_BAJA. En la tabla BAJA_LABORAL se crea la baja con la situación Activa. En la tabla REVISION_BAJA se crea un tipo de revisión de Alta.
Parámetros de entrada	p_idEnfermedadDiag, enfermedad por la que se diagnostica la baja. p_idMedico, médico que concede la baja. p_fechaInicio, fecha de inicio de la baja. p_observaciones, observaciones.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, a excepción de p_observaciones, sea nulo. Además, también comprueba que el mismo paciente no tenga otra baja activa.

ALTA	**SP_ALTA_REVISION_BAJA**
Descripción	Procedimiento que da de alta una revisión en la tabla REVISION_BAJA. Se utiliza al insertar una nueva revisión (de prorroga o finalización), no al dar de alta una baja. Además de insertar una revisión, si esta es de tipo Finalización, se modifica el estado de la baja a Finalizada.
Parámetros de entrada	p_idBaja, baja que se quiere revisar. p_idMedico, médico que realiza la revisión de la baja. p_fecha, fecha de revisión. p_idTipoRevisionBaja, tipo de revisión. p_observaciones,
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que ninguno de los parámetros de entrada, a excepción de p_observaciones, sea nulo. -Que no se esté insertando una revisión de tipo Alta. -Que una baja finalizada no se intente revisar de nuevo.

ALTA	**SP_ALTA_PRUEBA_DIAGNOSTICA**
Descripción	Procedimiento que da de alta una nueva prueba diagnóstica en la tabla PRUEBA_DIAGNOSTICA.
Parámetros de entrada	p_nombre, nombre de la prueba diagnóstica p_descripcion, p_contraindicaciones, contraindicaciones que tenga la prueba.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada p_nombre no sea nulo.

ALTA	SP_ALTA_PRUEBA_SOLICITADA
Descripción	Procedimiento que da de alta una prueba diagnóstica solicitada en la tabla PRUEBA_SOLICITADA. Se ejecutará cuando un médico solicite una prueba para un paciente.
Parámetros de entrada	p_idConsulta, consulta en la que se solicita la prueba. p_idPrueba, el tipo de prueba que se solicita. p_fechaSolicitud, fecha en la que se solicita la prueba. p_fechaPrevista, fecha prevista de realización de una prueba diagnóstica. p_idHospital, hospital en el que se realizará la prueba
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. En caso de no poderse realizar la prueba, el código de error es el 30004. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada p_idConsulta, p_idPrueba, p_fechaSolicitud y p_idHospital no sean nulos. -Se comprueba que en el hospital se pueda hacer realmente la prueba.

ALTA	SP_ALTA_DOCUMENTACION_PRUEBA
Descripción	Procedimiento que da de alta un registro en la tabla DOCUMENTACION_PRUEBA, para enlazar un documento a una prueba realizada.
Parámetros de entrada	p_idPruebaSolicitada, identificador de la prueba realizada a la que corresponde el documento. p_documento, ruta del documento.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada sean nulos.

ALTA	SP_ALTA_REALIZAN
Descripción	Procedimiento que da de alta un registro en la tabla REALIZAN. Se utiliza para indicar en qué hospitales se pueden realizar las pruebas médicas.
Parámetros de entrada	p_idPrueba, tipo de prueba diagnóstica. p_idHospital, identificador del hospital.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada sea nulo.

3.3.4. Procedimientos de baja

De la misma forma que hemos creado un procedimiento para dar de alta datos en cada una de las tablas de la BD, también hemos creado un procedimiento para eliminar la información de las mismas. Sólo en el caso de las bajas laborales no permitimos eliminar una baja. Por su complejidad, la gestión de las

bajas laborales la explicamos en el apartado 3.3.7"Gestión de las bajas laborales y las revisiones de las bajas".

Todos los procedimientos de baja se han creado siguiendo el mismo esquema. Primero comprobamos si el parámetro de entrada es nulo y después si existe un registro con el identificador pasado como parámetro. En caso de no ser así, provocamos la excepción correspondiente.

Si la eliminación de un registro entra en contradicción con una restricción de la base de datos, se genera la excepción y mostramos directamente el error que nos pasa el SGBD.

A continuación veremos la definición de cada uno de los procedimientos de baja.

BAJA	SP_BAJA_PAIS
Descripción	Procedimiento que da de baja un país en la tabla PAIS.
Parámetros de entrada	p_idPais, identificador del país.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los campos p_idPais no sea nulo. -Que el país indicado exista.

BAJA	SP_BAJA_POBLACION
Descripción	Procedimiento que da de baja una población en la tabla POBLACION.
Parámetros de entrada	p_codigoPostal, código postal de la población p_repeticionPostal, la repetición postal. p_idPais, identificador del país.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que exista una población con los parámetros indicados.

BAJA	SP_BAJA_TIP_AGR_POB
Descripción	Procedimiento que elimina un tipo de agrupación en la tabla TIPO_AGRUPACION_POBLACION.
Parámetros de entrada	p_idTipoAgrupacion, identificador del tipo de agrupación.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el tipo de agrupación indicado exista.

BAJA	SP_BAJA_AGRUPACION
Descripción	Procedimiento que elimina una agrupación en la tabla AGRUPACION.
Parámetros de entrada	p_idAgrupacion, identificador de la agrupación que se desea eliminar.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la agrupación indicada exista.

BAJA	**SP_BAJA_AGRUPACION_POBLACION**
Descripción	Procedimiento que elimina un registro en la tabla AGRUPACION_POBLACION.
Parámetros de entrada	p_idAgrupacion, p_codigoPostal, p_repeticionPostal y p_idPais. Todos los campos que definen la población agrupada. Es la clave primaria de la tabla.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que la agrupación de población indicada exista.

BAJA	**SP_BAJA_PACIENTE**
Descripción	Procedimiento que elimina un paciente en la tabla PACIENTE.
Parámetros de entrada	p_idPaciente, identificador del paciente que se desea eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada p_idPaciente no sea nulos. -Que el paciente exista.

BAJA	**SP_BAJA_MEDICO**
Descripción	Procedimiento que elimina un médico en la tabla MEDICO.
Parámetros de entrada	p_idMedico, identificador del médico que se desea eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada p_idMedico no sea nulos. -Que el médico exista.

BAJA	**SP_BAJA_ASIGNACION**
Descripción	Procedimiento que elimina un registro en la tabla ASIGNACION, que refleja la asignación de un médico a un paciente.
Parámetros de entrada	p_idPaciente, identificador del paciente. p_idMedico, identificador del médico.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que la asignación indicada exista.

BAJA	SP_BAJA_HOSPITAL
Descripción	Procedimiento que elimina un hospital en la tabla HOSPITAL.
Parámetros de entrada	p_idHospital, identificador del hospital que se desea eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el hospital indicado exista.

BAJA	SP_BAJA_TIPO_DESTINO
Descripción	Procedimiento que elimina un tipo de destino de la tabla TIPO_DESTINO.
Parámetros de entrada	p_idTipoDestino, identificador del tipo de destino a eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el tipo de destino exista.

BAJA	SP_BAJA_ESTADO_DESTINO
Descripción	Procedimiento que elimina un estado de destino en la tabla ESTADO_DESTINO.
Parámetros de entrada	p_idEstadoDestino, identificador del estado de destino a eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el estado de destino exista.

BAJA	SP_BAJA_GUARDIA
Descripción	Procedimiento que elimina una guardia en la tabla GUARDIA. Para ello se pasa como parámetro el identificador de la guardia.
Parámetros de entrada	p_idGuardia.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la guardia exista.

BAJA	**SP_BAJA_FARMACIA**
Descripción	Procedimiento que elimina una farmacia en la tabla FARMACIA.
Parámetros de entrada	p_idFarmacia, identificador de la farmacia.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la farmacia exista.

BAJA	**SP_BAJA_DESTINO**
Descripción	Procedimiento que elimina un destino en la tabla DESTINO. Con este procedimiento se eliminará uno de los destinos que tenga establecidos un médico.
Parámetros de entrada	p_idDestino, identificador del destino que se desea eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el destino exista.

BAJA	**SP_BAJA_FARMACEUTICO**
Descripción	Procedimiento que elimina un farmacéutico en la tabla FARMACEUTICO.
Parámetros de entrada	p_idFarmaceutico, identificador del farmacéutico a eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el farmacéutico exista.

BAJA	**SP_BAJA_MEDICAMENTO**
Descripción	Procedimiento que elimina un medicamento en la tabla MEDICAMENTO.
Parámetros de entrada	p_idMedicamento, identificador del medicamento a eliminar.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el medicamento exista.

BAJA	**SP_BAJA_TIPO_PRODUCTO**
Descripción	Procedimiento que elimina un tipo de producto en la tabla TIPO_PRODUCTO.
Parámetros de entrada	p_idTipoProducto, identificador del tipo de producto.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el tipo de producto exista.

BAJA	**SP_BAJA_PRODUCTO**
Descripción	Procedimiento que elimina un producto en la tabla PRODUCTO.
Parámetros de entrada	p_idProducto, identificador del producto.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el producto exista.

BAJA	**SP_BAJA_ENFERMEDAD**
Descripción	Procedimiento que elimina una enfermedad conocida en la tabla ENFERMEDAD.
Parámetros de entrada	p_idEnfermedad, identificador de la enfermedad a eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la enfermedad exista.

BAJA	**SP_BAJA_PRESCRIPCION**
Descripción	Procedimiento que elimina una prescripción en la tabla PRESCRIPCION. Para eliminarla hay que indicar tanto la enfermedad como el medicamento.
Parámetros de entrada	p_idEnfermedad, p_idMedicamento, ambos campos forman la clave primaria de la tabla.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada p_idEnfermedad y p_idMedicamento no sean nulos. -Que la prescripción exista.

BAJA	**SP_BAJA_ENF_DIAG**
Descripción	Procedimiento que elimina una enfermedad diagnosticada en la tabla ENFERMEDAD_DIAGNOSTICADA. Se realizará cuando un diagnóstico no se haya realizado correctamente.
Parámetros de entrada	p_idEnfermedadDiag, identificador de la enfermedad diagnosticada.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la enfermedad diagnosticada exista.

BAJA	**SP_BAJA_RECETA**
Descripción	Procedimiento que elimina una receta en la tabla RECETA. Por las restricciones de las tablas, no se podrá eliminar una receta dispensada.
Parámetros de entrada	p_idReceta, identificador de la receta.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la receta exista.

BAJA	**SP_BAJA_DISPENSACION**
Descripción	Procedimiento que elimina una dispensación en la tabla DISPENSACIÓN. Esto se realiza cuando el farmacéutico dispensa por error una receta.
Parámetros de entrada	p_idReceta, identificador de la receta.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la dispensación exista.

BAJA	**SP_BAJA_CONSULTA**
Descripción	Procedimiento que elimina una consulta médica en la tabla CONSULTA. No se podrá eliminar una consulta si tiene una solicitud, un diagnóstico o una prueba solicitada.
Parámetros de entrada	p_idConsulta, identificador de la consulta.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la consulta exista.

BAJA	**SP_BAJA_TIPO_CONSULTA**
Descripción	Procedimiento que elimina un tipo de consulta en la tabla TIPO_CONSULTA. No se podrá eliminar si existe alguna consulta de ese tipo.
Parámetros de entrada	p_idTipoConsulta, identificador del tipo de consulta.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el tipo de consulta exista.

BAJA	**SP_BAJA_ESTADO_CONSULTA**
Descripción	Procedimiento que elimina un estado de consulta en la tabla ESTADO_CONSULTA. No se podrá eliminar si existe alguna consulta de este estado.
Parámetros de entrada	p_idEstadoConsulta, identificador del estado de la consulta.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el estado de consulta exista.

BAJA	**SP_BAJA_DIAGNOSTICO**
Descripción	Procedimiento que elimina un diagnóstico en la tabla DIAGNOSTICO. Las enfermedades diagnosticadas se relacionan con el paciente a través de las consultas médicas. Para eliminar un diagnóstico, es necesario identificar tanto la consulta como la enfermedad.
Parámetros de entrada	p_idConsulta, en la que se diagnostica una enfermedad. p_idEnfermedadDiag, enfermedad diagnosticada.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que ninguno de los parámetros de entrada sea nulo. -Que el diagnóstico exista.

BAJA	SP_BAJA_SOLICITUD
Descripción	Procedimiento que elimina una solicitud en la tabla SOLICITUD. Se utilizará cuando se solicite una consulta por error. Únicamente se necesita identificar la consulta.
Parámetros de entrada	p_idConsulta, identificador de la consulta que se desea eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la solicitud de la consulta exista.

BAJA	SP_BAJA_ESPECIALIDAD
Descripción	Procedimiento que elimina una especialidad en la tabla ESPECIALIDAD. No se podrá eliminar una especialidad que esté asignada a un médico.
Parámetros de entrada	p_idEspecialidad, identificador de la especialidad.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la especialidad exista.

BAJA	SP_BAJA_ESP_ADQ
Descripción	Procedimiento que elimina una especialidad adquirida en la tabla ESPECIALIDAD_ADQUIRIDA. De esta forma se eliminarán cada una de las especialidades adquiridas por cada médico. Hay que identificar tanto el médico como la especialidad que queremos eliminarle.
Parámetros de entrada	p_idMedico, p_idEspecialidad.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que ninguno de los parámetros de entrada sea nulo. -Que la especialidad esté asignada al médico.

BAJA	SP_BAJA_TIPO_REVISION_BAJA
Descripción	Procedimiento que elimina un tipo de revisión de baja en la tabla TIPO_REVISION_BAJA. Inicialmente tendremos tres tipos: Alta, Prorroga y Finalización.
Parámetros de entrada	p_idTipoRevisionBaja, identificador del tipo de revisión.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.

Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el tipo de revisión de baja exista.

BAJA	SP_BAJA_ESTADO_BAJA
Descripción	Procedimiento que elimina un estado de baja en la tabla ESTADO_BAJA. Inicialmente tendremos dos estados: Activa y Finalizada.
Parámetros de entrada	p_idEstadoBaja, identificador del estado de la baja.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que el estado de revisión de baja exista.

BAJA	SP_BAJA_REVISION_BAJA
Descripción	Procedimiento que elimina una revisión en la tabla REVISION_BAJA. Además: -Si la revisión es de tipo ALTA, se elimina también la baja. -Si la revisión es de tipo FINALIZACION, se cambia el estado de la baja a ACTIVA y se borra la fecha de finalización. -Si la revisión es de tipo PRORROGA, no se cambia nada en la tabla BAJA_LABORAL.
Parámetros de entrada	p_idRevisionBaja, identificador de la revisión.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la revisión de baja exista.

BAJA	SP_BAJA_PRUEBA_DIAGNOSTICA
Descripción	Procedimiento que elimina una prueba diagnóstica en la tabla PRUEBA_DIAGNOSTICA. No se podrá eliminar una prueba que haya sido solicitada para algún paciente, o que se haya indicado que se puede realizar en algún hospital.
Parámetros de entrada	p_idPrueba, identificador de la prueba diagnóstica a eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la prueba diagnóstica exista.

BAJA	SP_BAJA_PRUEBA_SOLICITADA
Descripción	Procedimiento que elimina una prueba diagnóstica solicitada en la tabla PRUEBA_SOLICITADA. Se ejecutará cuando un médico haya solicitado una prueba por error. No se podrá eliminar si tiene algún documento asociado.
Parámetros de entrada	p_idPruebaSolicitada, identificador de la prueba solicitada.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. En caso de no poderse realizar la prueba, el código de error es el 30004. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la prueba diagnóstica exista.

BAJA	**SP_BAJA_DOCUMENTACION_PRUEBA**
Descripción	Procedimiento que elimina un registro en la tabla DOCUMENTACION_PRUEBA.
Parámetros de entrada	p_idDocumentacionPrueba, identificador del documento a eliminar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que la documentación exista.

BAJA	**SP_BAJA_REALIZAN**
Descripción	Procedimiento que elimina un registro en la tabla REALIZAN. Se utiliza para indicar qué un hospital no puede realizar una prueba médica. Es necesario indicar tanto la prueba como el hospital.
Parámetros de entrada	p_idPrueba, tipo de prueba diagnóstica. p_idHospital, identificador del hospital.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que ninguno de los parámetros de entrada sea nulo. -Que la prueba se esté realizando en el hospital.

3.3.5. Procedimientos de modificación

Después de ver los procedimientos de alta y baja de registros, en este apartado veremos los procedimientos para modificar la información ya insertada en las tablas.

En el caso de los procedimientos de modificación, sólo permitimos modificar la información de ciertas tablas. Además en algunos procedimientos sólo permitimos modificar ciertos campos, para mantener la integridad referencial. Si se necesita modificar otros campos, será necesario borrar el registro y volverlo a crear. Aquí actuarán de nuevo las restricciones aplicadas al crear las tablas, que mantendrán la consistencia de la información.

MODIFICACIÓN	**SP_MODIF_PAIS**
Descripción	Procedimiento que modifica un país en la tabla PAIS. Se puede modificar el nombre y el nemotécnico.
Parámetros de entrada	p_idPais, identificador del país que se desea modificar. p_nombre, nombre a actualizar. p_nemotecnico, nemotécnico a actualizar.

Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los campos de entrada no sean nulos. -Que el país indicado exista.

MODIFICACIÓN	SP_MODIF_POBLACION
Descripción	Procedimiento que modifica una población en la tabla POBLACION. Sólo se puede modificar el nombre.
Parámetros de entrada	p_codigoPostal, código postal de la población p_repeticionPostal, la repetición postal. p_idPais, identificador del país. p_nombre, nombre de la población a actualizar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que exista una población con los parámetros indicados.

MODIFICACIÓN	SP_MODIF_TIP_AGR_POB
Descripción	Procedimiento que modifica un tipo de agrupación en la tabla TIPO_AGRUPACION_POBLACION. Sólo se puede modificar la descripción.
Parámetros de entrada	p_idTipoAgrupacion, identificador del tipo de agrupación. p_descripcion, descripción a actualizar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que el tipo de agrupación indicado exista.

MODIFICACIÓN	SP_MODIF_AGRUPACION
Descripción	Procedimiento que modifica una agrupación en la tabla AGRUPACION. Se puede modificar tanto la descripción como el tipo.
Parámetros de entrada	p_idAgrupacion, identificador de la agrupación. p_descripcion, descripción a actualizar. p_idTipoAgrupacion, tipo de agrupación a actualizar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que la agrupación indicada exista.

MODIFICACIÓN	SP_MODIF_PACIENTE

Descripción	Procedimiento que modifica un paciente en la tabla PACIENTE. Se pueden modificar todos los campos, a excepción del identificador (idPaciente). Éste se utiliza para buscar el paciente a modificar.
Parámetros de entrada	p_idPaciente, identificador del paciente que se desea eliminar. p_SIP, p_NIF, p_nombre, p_apellido1, p_apellido2, p_fechaNacimiento, p_direccion, p_codigoPostal, p_repeticionPostal, p_idPais, p_idPaisNacimiento, p_telefono, p_email.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada p_idPaciente, p_SIP, p_NIF, p_nombre, p_apellido1, p_codigoPostal, p_repeticionPostal, p_idPais y p_idPaisNacimiento, no sean nulos. -Que el paciente exista.

MODIFICACIÓN	**SP_MODIF_MEDICO**
Descripción	Procedimiento que modifica un médico en la tabla MEDICO. Se pueden modificar todos los campos, a excepción del identificador (idMedico), éste se utiliza para buscar el médico a modificar.
Parámetros de entrada	p_idMedico, identificador del médico que se desea eliminar. p_NIF, p_nombre, p_apellido1, p_apellido2, p_numColegiado, p_idPaisNacimiento
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada p_ idMedico, p_NIF, p_nombre, p_apellido1, p_numColegiado y p_idPaisNacimiento no sea nulos. -Que el médico exista.

MODIFICACIÓN	**SP_MODIF_HOSPITAL**
Descripción	Procedimiento que modifica un hospital en la tabla HOSPITAL. Se pueden modificar todos los campos, a excepción del identificador (idHospital). Éste se utiliza para buscar el hospital a modificar.

Parámetros de entrada	p_idHospital, identificador del hospital que se desea eliminar. p_nombre, p_numPlazas, número de plazas disponibles en el hospital. p_direccion, p_codigoPostal, p_repeticionPostal, p_idPais.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada p_idHospital, p_nombre, p_codigoPostal, p_repeticionPostal y p_idPais no sean nulos. -Que el hospital indicado exista.

MODIFICACIÓN	**SP_MODIF_TIPO_DESTINO**
Descripción	Procedimiento que modifica un tipo de destino de la tabla TIPO_DESTINO. Sólo se puede modificar la descripción.
Parámetros de entrada	p_idTipoDestino, identificador del tipo de destino a eliminar. p_descripcion.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que el tipo de destino exista.

MODIFICACIÓN	**SP_MODIF_ESTADO_DESTINO**
Descripción	Procedimiento que modifica un estado de destino en la tabla ESTADO_DESTINO. Sólo se puede modificar la descripción.
Parámetros de entrada	p_idEstadoDestino, identificador del estado de destino a eliminar. p_descripcion.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que el estado de destino exista.

MODIFICACIÓN	**SP_MODIF_GUARDIA**
Descripción	Procedimiento que modifica una guardia en la tabla GUARDIA. Para ello se pasa como parámetro el identificador de la guardia para buscar la que queremos modificar. El resto de campos se pueden modificar.

Parámetros de entrada	p_idGuardia, p_idFarmacia, p_codigoPostal, p_repeticionPostal, p_idPais, p_fechaInicio, p_fechaFin
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada, a excepción de p_fechaFin, no sean nulos. -Que la guardia exista.

MODIFICACIÓN	**SP_MODIF_FARMACIA**
Descripción	Procedimiento que modifica una farmacia en la tabla FARMACIA. Se pueden modificar todos los campos, a excepción del identificador (idFarmacia). Éste se utiliza para buscar la farmacia a modificar.
Parámetros de entrada	p_idFarmacia, identificador de la farmacia. p_nombre, nombre de la farmacia. p_idFarmaceutico, farmacéutico titular de la farmacia. p_direccion, p_codigoPostal, p_repeticionPostal, p_idPais, ubicación de la farmacia.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada, a excepción de la dirección, no sean nulos. -Que la farmacia exista.

MODIFICACIÓN	**SP_MODIF_DESTINO**
Descripción	Procedimiento que modifica un destino en la tabla DESTINO. Con este procedimiento se modificará uno de los destinos que tenga establecidos un médico. Se pueden modificar todos los campos, a excepción del identificador (idDestino). Éste se utiliza para buscar el destino a modificar.
Parámetros de entrada	p_idDestino, identificador del destino que se desea eliminar. p_idHospital, p_idMedico, p_idEstadoDestino, p_idTipoDestino, p_fechaInicio, p_fechaFin.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada, a excepción de p_fechaFin, no sean nulos. -Que el destino exista.

MODIFICACIÓN	SP_MODIF_FARMACEUTICO
Descripción	Procedimiento que modifica un farmacéutico en la tabla FARMACEUTICO. Se pueden modificar todos los campos, a excepción del identificador (idFarmaceutico). Éste se utiliza para buscar el farmacéutico a modificar.
Parámetros de entrada	p_idFarmaceutico, identificador del farmacéutico a eliminar. p_NIF, p_nombre, p_apellido1, p_apellido2, p_numColegiado, p_idPaisNacimiento
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada, a excepción de p_apellido_2, no sean nulos. -Que el farmacéutico exista.

MODIFICACIÓN	SP_MODIF_MEDICAMENTO
Descripción	Procedimiento que modifica un medicamento en la tabla MEDICAMENTO. Se pueden modificar todos los campos, a excepción del identificador (idMedicamento). Éste se utiliza para buscar el medicamento a modificar.
Parámetros de entrada	p_idMedicamento, identificador del medicamento a eliminar. p_nombre, p_composicion, p_observaciones
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada p_idMedicamento y p_nombre, no sean nulos. -Que el medicamento exista.

MODIFICACIÓN	SP_MODIF_PRODUCTO
Descripción	Procedimiento que modifica un producto en la tabla PRODUCTO. Se pueden modificar todos los campos, a excepción del identificador (idProducto). Éste se utiliza para buscar el producto a modificar.
Parámetros de entrada	p_idProducto, identificador del producto. p_idMedicamento, medicamento al que corresponde el producto. p_nombre, nombre del producto p_fabricante, p_idTipoProducto
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.

Comprobaciones	-Que los parámetros de entrada, a excepción de p_fabricante, no sean nulos. -Que el producto exista.

MODIFICACIÓN	SP_MODIF_ENFERMEDAD
Descripción	Procedimiento que modifica una enfermedad conocida en la tabla ENFERMEDAD. Se pueden modificar todos los campos, a excepción del nombre y del identificador (idEnfermedad). Éste se utiliza para buscar la enfermedad a modificar.
Parámetros de entrada	p_idEnfermedad, identificador de la enfermedad a eliminar. p_descripcion, p_sintomas.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada p_idEnfermedad no sea nulo. -Que la enfermedad exista.

MODIFICACIÓN	SP_MODIF_PRESCRIPCION
Descripción	Procedimiento que modifica una prescripción en la tabla PRESCRIPCION. Para modificarla hay que indicar tanto la enfermedad como el medicamento. Sólo se pueden modificar la posología y las observaciones.
Parámetros de entrada	p_idEnfermedad, p_idMedicamento, ambos campos forman la clave primaria de la tabla. p_posología, p_observaciones.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada p_idEnfermedad y p_idMedicamento no sean nulos. -Que la prescripción exista.

MODIFICACIÓN	SP_MODIF_RECETA
Descripción	Procedimiento que modifica una receta en la tabla RECETA. Sólo se puede modificar la posología de una receta. Ante cualquier otro cambio, hay que eliminar la receta.
Parámetros de entrada	p_idReceta, identificador de la receta que se desea modificar. p_posologia, la posología a actualizar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada p_idReceta no sea nulo. -Que la receta exista.

MODIFICACIÓN	SP_MODIF_CONSULTA

Descripción	Procedimiento que modifica una consulta médica en la tabla CONSULTA. No se podrán modificar todos los campos. Los campos que relacionan al hospital, al paciente, al médico al tipo, y a la fecha de alta, no se podrán modificar. Si ha habido un error al darla de alta, se deberá borrar.
Parámetros de entrada	p_idConsulta, identificador de la consulta. p_idEstadoConsulta, p_fechaPrevista, p_fechaRealizada, p_observaciones.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada p_idConsulta, p_idEstadoConsulta y p_fechaPrevista no sean nulos. -Que la consulta exista.

MODIFICACIÓN	**SP_MODIF_BAJA**
Descripción	Procedimiento que modifica únicamente las observaciones de una baja laboral en la tabla BAJA_LABORAL. Para modificar el estado o la fecha de finalización, hay que utilizar el procedimiento que da de alta o elimina las revisiones de las bajas.
Parámetros de entrada	p_idBaja, identificador de la baja. p_observaciones, campo de observaciones a actualizar.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada p_idBaja no sea nulo. -Que la revisión de baja exista.

MODIFICACIÓN	**SP_MODIF_REVISION_BAJA**
Descripción	Procedimiento que modifica únicamente las observaciones y la fecha de una revisión en la tabla REVISION_BAJA. Además: -Si la revisión es de tipo ALTA, se modifica la fecha de alta de la baja laboral. -Si la revisión es de tipo FINALIZACION, se cambia la fecha de finalización de la baja. -Si la revisión es de tipo PRORROGA, no se cambia nada en la tabla BAJA_LABORAL.
Parámetros de entrada	p_idRevisionBaja, identificador de la revisión. p_fecha, p_observaciones.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.

Comprobaciones	-Que los parámetros de entrada p_idRevisionBaja y p_fecha no sean nulos. -Que la revisión de baja exista.

MODIFICACIÓN	SP_MODIF_PRUEBA_DIAGNOSTICA
Descripción	Procedimiento que modifica una prueba diagnóstica en la tabla PRUEBA_DIAGNOSTICA. Sólo se podrá modificar la descripción y las contraindicaciones. El resto de campos no se podrán modificar. En caso de ser necesario, se deberá borrar la prueba.
Parámetros de entrada	p_idPrueba, identificador de la prueba diagnóstica a eliminar. p_descripcion, p_contraindicaciones.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada p_idPrueba no sea nulo. -Que la prueba diagnóstica exista.

MODIFICACIÓN	SP_MODIF_PRUEBA_SOLICITADA
Descripción	Procedimiento que modifica una prueba solicitada en la tabla PRUEBA_SOLICITADA. Sólo se podrá modificar la fecha prevista y la de realización.
Parámetros de entrada	p_idPruebaSolicitada, identificador de la prueba solicitada, p_fechaPrevista, p_fechaRealizada.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. En caso de no poderse realizar la prueba, el código de error es el 30004. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada p_idPruebaSolicitada no sea nulo. -Que la prueba solicitada exista.

3.3.6. Procedimientos de selección

El acceso a la base de datos por parte de sistemas externos se va a realizar únicamente a través de procedimientos almacenados. Hasta ahora hemos creado todos los procedimientos que permitirán dar de alta, eliminar y modificar cualquier registro de cualquier tabla de nuestra base de datos. Para poder acceder a la información, es necesario que creemos procedimientos de selección que puedan ser invocados y que devuelvan los datos solicitados. Algunos de estos procedimientos únicamente devolverán un valor en un parámetro de salida, pero otros devolverán todos los registros que cumplan con los parámetros de entrada. A modo de ejemplo, si un sistema externo debe dar opción a modificar un medicamento de entre los que existen en la base de datos; primero deberá invocar un procedimiento que le devuelva todos los medicamentos que están dados de alta en la tabla MEDICAMENTO. De esta forma, necesitaremos un proceso de selección para cada una de las tablas que tenemos en la base de datos.

Vamos ahora a poner otro ejemplo. Un sistema externo puede necesitar obtener información sobre el historial médico de un paciente y sobre qué enfermedades han provocado que el paciente esté de baja. Para obtener esta información podemos proceder de dos formas:

1. Teniendo en cuenta que ya disponemos de los procedimientos de selección para cada una de las tablas, la primera forma sería realizar distintas consultas a través de estos procedimientos a las tablas PACIENTE, BAJA_LABORAL y ENFERMEDAD_DIAGNOSTICADA, para así sacar la información.

2. Una segunda opción, que facilitaría la integración entre el sistema externo y la base de datos, sería el disponer de un procedimiento de selección que, a partir del nombre o identificador de un paciente, devolviera toda la información requerida.

Como podemos deducir, el número de procedimientos de selección necesarios para permitir consultar toda la información de la base de datos es muy grande. A diferencia de lo que hemos hecho para los procedimientos de Alta, Baja y Modificación; en este caso sólo implementamos un reducido número de muestra, el resto de procedimientos serán similares a éstos.

Para implementar los procedimientos de selección, hemos creado unos tipos de datos con la estructura de los registros que devolverán los procedimientos; así como unos tipos de que datos que serán de tipo *tabla* de los tipos anteriores. Los procedimientos de selección tendrán un parámetro de salida que será de uno de estos tipos *tabla* que hemos creado.

Todos los procedimientos de selección están diseñados de una forma similar:

1. Primero comprobamos que existen datos que cumplan con los criterios de los parámetros de entrada.

2. Si es cierto, se ejecuta la consulta para obtener los datos y se inserta el resultado en el parámetro de salida.

3. En caso de no existir datos para los parámetros de entrada, generamos la excepción correspondiente.

A continuación veremos la definición de los procedimientos que hemos creado a modo de ejemplo. Como en el resto de procedimientos, no incluimos el código en la memoria, se puede consultar en la carpeta "*Producto\1. Diseño fisico Base de datos\Procedimientos almacenados\SELECCION*"

SELECCIÓN	SP_ULTIMO_REGISTRO
Descripción	Este procedimiento nos ayudará en la mayoría de trabajos de inserción de datos en la BD. Se trata de un procedimiento para obtener el último ID insertado en una tabla. En los procedimientos de alta no devolvemos el ID del registro creado, pero con este procedimiento lo podremos obtener. De esta forma, al dar de alta un registro en una tabla, sabremos su indicador a través de este procedimiento. Como parámetros de entrada tiene el nombre de la tabla y el del campo del que queremos obtener el máximo.
Parámetros de entrada	p_nombreTabla, p_nombreCampo.
Parámetros de salida	p_ultimoID OUT VARCHAR2, parámetro en el que se devuelve el valor del máximo identificador. En caso de error devuelve el valor -1. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. Si no existe la tabla, el código es -30005. Si no existe el campo, el código es -30006. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que los parámetros de entrada no sean nulos. -Que la tabla exista. -Que el campo exista en la tabla indicada.

SELECCIÓN	SP_SEL_PACIENTE
Descripción	Procedimiento que devuelve todos los registros de la tabla PACIENTE que cumplen los criterios que se han pasado como parámetros de entrada. Todos los parámetros de entrada aceptan nulos, el procedimiento se podrá invocar indicando uno, varios o ningún parámetro de entrada. Si se indica más de un parámetro, el parámetro de salida contendrá los registros que cumplan con todos los criterios indicados. Como parámetros de salida, tenemos los dos campos que estamos utilizando en todos los procedimientos para controlar la correcta finalización de la ejecución, más uno de tipo tabla que contendrá los registros solicitados.
Parámetros de entrada	p_idPaciente, p_SIP, p_NIF, p_nombre, p_apellido1, p_apellido2.
Parámetros de salida	p_tPaciente t_objPaciente, parámetro de tipo tabla en el que se devuelven los registros solicitados. Cada registro contiene todos los campos de la tabla PACIENTE. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. En caso de no existir datos con parámetros de entrada, el código de error es el 30002. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que existan registros en la tabla que cumplan con los parámetros indicados.

SELECCIÓN	SP_SEL_MEDICO
Descripción	Procedimiento que devuelve todos los registros de la tabla MEDICO que cumplen los criterios que se han pasado como parámetros de entrada. Todos los parámetros de entrada aceptan nulos, el procedimiento se podrá invocar indicando uno, varios o ningún parámetro de entrada. Si se indica más de un parámetro, el parámetro de salida contendrá los registros que cumplan con todos los criterios indicados. Como parámetros de salida, tenemos los dos campos que estamos utilizando en todos los procedimientos para controlar la correcta finalización de la ejecución, más uno de tipo tabla que contendrá los registros solicitados.
Parámetros de entrada	p_idMedico, p_NIF, p_nombre, p_apellido1, p_apellido2, p_numColegiado.

Parámetros de salida	p_tMedico t_objMedico, parámetro de tipo tabla en el que se devuelven los registros solicitados. Cada registro contiene todos los campos de la tabla MEDICO p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. En caso de no existir datos con parámetros de entrada, el código de error es el 30002. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que existan registros en la tabla que cumplan con los parámetros indicados.

SELECCIÓN	**SP_SEL_ENFERMEDAD**
Descripción	Procedimiento que devuelve todos los registros de la tabla ENFERMEDAD que cumplen los criterios que se han pasado como parámetros de entrada. Todos los parámetros de entrada aceptan nulos, el procedimiento se podrá invocar indicando uno, varios o ningún parámetro de entrada. Si se indica más de un parámetro, el parámetro de salida contendrá los registros que cumplan con todos los criterios indicados. Como parámetros de salida, tenemos los dos campos que estamos utilizando en todos los procedimientos para controlar la correcta finalización de la ejecución, más uno de tipo tabla que contendrá los registros solicitados.
Parámetros de entrada	p_idEnfermedad, p_nombre, p_descripcion, p_sintomas.
Parámetros de salida	p_tEnfermedad t_objEnfermedad, parámetro de tipo tabla en el que se devuelven los registros solicitados. Cada registro contiene todos los campos de la tabla ENFERMEDAD p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. En caso de no existir datos con parámetros de entrada, el código de error es el 30002. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que existan registros en la tabla que cumplan con los parámetros indicados.

SELECCIÓN	SP_SEL_HISTORIAL_PACIENTE
Descripción	Procedimiento que devuelve el historial del paciente que se pasa como parámetro. El historial incluye las enfermedades diagnosticadas; el médico que las diagnosticó y la fecha del diagnóstico; si la enfermedad provocó que el paciente estuviera de baja; la fecha de inicio y de finalización de la baja, y los días de duración. El parámetro de salida es de tipo tabla, y cada registro contiene los siguientes campos: idPaciente: identificador del paciente. idEnfermedadDiag: identificador de la enfermedad diagnosticada. idEnfermedad: identificador de la enfermedad. nombreEnfermedad. idMedico. nombreMedico. fechaDiagnostico. provocaBaja: parámetro que si es igual a 0 indica la enfermedad provocó una baja. fechaInicio: inicio de la baja, si la hubiera. fechaFin: fin de la baja si la hubiera y además si el cliente ya tuviera el alta. diasBaja: días de duración de la baja. El parámetro de entrada es el identificador del paciente, por lo que habrá que buscar antes este identificador con el procedimiento correspondiente Como parámetros de salida, tenemos los dos campos que estamos utilizando en todos los procedimientos para controlar la correcta finalización de la ejecución, más uno de tipo tabla que contendrá los registros solicitados.
Parámetros de entrada	p_idPaciente: identificador del paciente del que queremos obtener su historial.
Parámetros de salida	p_tHistorial t_objHistorial, parámetro de tipo tabla en el que se devuelven los registros solicitados. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. En caso de no existir datos con parámetros de entrada, el código de error es el 30002. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	-Que el parámetro de entrada no sea nulo. -Que existan registros que cumplan con los parámetros indicados.

3.3.7. Gestión de las bajas laborales y las revisiones de las bajas

Una vez que hemos visto la definición de todos los procedimientos de alta, de baja, de modificación y de selección implementados; vamos a detallar la gestión que se hace de las bajas laborales a partir de la creación o eliminación de las revisiones.

Estas son las reglas que hemos establecido a partir del diseño lógico explicado en el apartado 3.2.

1. En los procedimientos de alta hay que controlar que no existan dos bajas laborales del mismo paciente activas. Un paciente puede tener muchas bajas, pero sólo una puede estar ACTIVA.

2. Impedimos que se pueda dar de alta una revisión de tipo ALTA, sólo se crearán al ejecutar el procedimiento de alta de una baja laboral.

3. Impedimos que se pueda cambiar el estado de una revisión o de una baja laboral. El estado cambia al dar de alta o de baja las distintas revisiones. De esta forma al dar de alta una revisión de tipo finalización, el estado de la baja pasa a ser FINALIZADA, y se cumplimenta la fecha de finalización de la baja.

4. Una baja ya finalizada no podrá volver a revisarse. En caso de que sea necesario, antes se deberá borrar la revisión que finalizó dicha baja.

A continuación, veremos los procedimientos almacenados que se encargan de la gestión de las bajas, y lo que se realiza en cada uno de ellos.

SP_ALTA_BAJA_LABORAL.
1. Se comprueba que no exista otra baja activa para el mismo paciente, en ese caso se genera una excepción y se devuelve un mensaje de error.
2. Se crea la baja laboral indicando la fecha de inicio de la misma y dejando la baja en situación ACTIVA.
3. Se crea una revisión de tipo ALTA.

SP_ALTA_REVISION_BAJA
1. No se permite dar de alta una revisión de tipo ALTA. Se genera una excepción en tal caso.
2. Se comprueba que la baja no esté finalizada, en ese caso no se puede volver a revisar.
3. Después de insertar la revisión, si ésta es de FINALIZACIÓN, modifico el estado de la baja e indico la fecha de finalización.

SP_MODIF_BAJA_LABORAL
1. No se permite modificar directamente ni la fecha de inicio, ni la de fin, ni el estado de la baja.
2. Tampoco permitimos cambiar la enfermedad diagnosticada. Únicamente se pueden cambiar las observaciones.

SP_MODIF_REVISION_BAJA
1. No permitimos cambiar el tipo de revisión, ni el médico que la ha hecho, ni la baja relacionada. Es decir, no permitimos cambiar ninguna clave ajena. Si algún campo de estos no es correcto, tendremos que dar de baja la revisión y volverla a crear.
2. Sólo permitimos cambiar las observaciones y la fecha. Si la revisión es de tipo ALTA, modificaremos también la fecha de inicio de la baja. Por otra parte, si la revisión es de tipo FINALIZACIÓN, también modificamos la fecha de finalización de la baja.

SP_BAJA_REVISION_BAJA
1. No permitimos dar de baja una baja laboral directamente. Para ello primero hay que dar de baja todas las revisiones.
2. Cuando se da de baja la revisión de tipo ALTA, entonces se borra la baja de la tabla.
3. Cuando se borra una revisión de tipo FINALIZADA, se modifica el estado de la baja a activa, y se borra la fecha de finalización.
4. Cuando se borra una revisión de tipo PRÓRROGA, la baja no se modifica.

4. Almacén de datos

Una vez que ya hemos completado la implementación de la base de datos, vamos ahora a definir el almacén de datos que nos solicita el cliente. El almacén de datos se desarrolla por la necesidad del Ministerio de Sanidad, de disponer de estadísticas para poder analizar la situación global de Sistema Sanitario. De esta forma se facilitará a los responsables de analizar los datos, una herramienta para que puedan realizar ellos mismos las consultas que necesiten más fácilmente y en menor tiempo.

La fuente de información de nuestro almacén de datos va a ser la base de datos operacional que hemos diseñado en este mismo proyecto. Esto facilita la extracción de los datos, ya que ésta tendrá el mismo gestor de base de datos que el DWH. Esto no quiere decir que el almacén tenga que estar en el mismo servidor, podremos tener dos SGBD en dos máquinas diferentes; esto dependerá de la carga del sistema y de las características del servidor utilizado. En la implementación de los procesos de carga de los datos del DWH hemos supuesto que ambos sistemas se encuentran en la misma máquina.

4.1. Modelo conceptual

Para el diseño del almacén de datos nos basaremos en un modelo multidimensional, teniendo en cuenta las necesidades del cliente y la estructura de la base de datos operacional. Deberemos identificar los procesos de negocio que necesitamos incorporar a nuestro modelo para satisfacer los requisitos del cliente. A la hora de diseñar el almacén de datos hay que tener en cuenta tanto las necesidades actuales, como prever las necesidades futuras. De este modo, a la hora de establecer las dimensiones y los atributos de éstas, las definiremos de forma que se puedan ampliar en un futuro.

4.1.1 Análisis de requisitos

De la misma forma que hacíamos al diseñar la base de datos, lo que primero que debemos hacer es especificar los requisitos del cliente.

El ministerio de Sanidad nos hace llegar la necesidad de analizar varios procesos de negocio. Pasamos a detallarlos a continuación.

1. Análisis de las consultas. Se necesita analizar las consultas médicas, ya sean de urgencias o externas. A partir de las estadísticas tendremos que poder saber las épocas del año en las que hay más consultas y también qué horas del día son las de mayor afluencia. Esto facilitará la gestión de los servicios de urgencias, y los de las consultas externas.

2. Análisis del consumo de medicamentos. Analizaremos el consumo de medicamentos facilitando la extracción de información sobre qué enfermedades necesitan más medicamentos o a qué edad el paciente tiene un consumo mayor.

3. Análisis de las bajas laborales. En el caso de las bajas laborales, facilitaremos el análisis del tiempo medio en el que una persona está de baja, qué enfermedades provocan más bajas, etc.

4.1.2. Elementos de análisis

Para diseñar nuestro DWH vamos a utilizar un modelo de estrella. Separaremos los hechos definidos en tres estrellas distintas. A continuación vamos a analizar cada una de las estrellas por separado, teniendo en cuenta que varias estrellas pueden tener la misma dimensión.

4.1.2.1. Hechos

A raíz de los requisitos del cliente, identificamos tres hechos que debemos reflejar en nuestro modelo. Por una parte tendremos el hecho de las **consultas**, por otra también tendremos en cuenta el hecho del **consumo de medicamentos**, y por último tendremos las **bajas laborales**.

4.1.2.2. Estrella consulta

Una vez identificado el hecho, el siguiente paso es elegir la granularidad para definir el detalle con el que se resumirá la información en las celdas del DWH. Deberemos encontrar un nivel en el que no perdamos detalle, pero tampoco aumentar el número de registros en el almacén de datos provocando una ralentización en la ejecución de las consultas por parte de los usuarios.

Tras analizar los requisitos, vemos que necesitamos registrarlas consultas realizadas por hora; de esta forma se podrá analizar a qué hora del día se producen más urgencias. En el caso de las consultas externas este dato no es importante, ya que están todas planificadas.

Para cada estrella de nuestro modelo detallaremos las dimensiones que necesitamos indicando sus atributos. Las dimensiones serán las diferentes perspectivas a través de las cuales los usuarios podrán analizar la información.

Dimensiones

Fecha de la consulta

Es la fecha y hora en la que se crea las consultas. Esta dimensión constará de varios niveles, con esta jerarquía de agregación:

Año
Mes
Semana
Día
Hora.

Paciente

Es el paciente que recibe la consulta. Como identificadores de esta dimensión tendremos la edad del paciente en el momento de recibir la consulta, y el país de nacimiento del mismo.

Médico

Es el médico que realiza la consulta. En estos momentos no vemos la necesidad de incorporar ningún identificador a esta dimensión, pero la creamos por si surgen necesidades en un futuro. En tal caso, bastará con ampliar la dimensión que enriquecer el modelo con la información necesaria. Por ejemplo, en un futuro puede interesar analizar el número de consultas que realizan médicos que están interinos y los que están fijos.

Hospital

Es el h**ospital en el que se realiza la consulta. Lo identificaremos por el nombre y por la ubicación del mismo**. Para identificar la ubicación necesitaremos tanto el código como la repetición postal, y el país. De esta forma se podrán analizar las poblaciones en las que hay un mayor número de consultas.

Tipo de consulta

Inicialmente las consultas pueden ser de urgencias o consultas externas. Esta dimensión es necesaria para poder analizar estos dos tipos de consultas por separado.

Indicadores

Lo único que mediremos en este caso es el número de consultas.

Celdas

En el caso de las consultas la celda elegida es aquella que almacena el número de consultas por paciente, médico, hospital y tipo de consulta en una hora. Se trata de una granularidad con mucho detalle, de hecho supondrá tener un registro en la tabla de hecho por cada consulta; pero si resumimos con una granularidad superior, perdemos información en nuestro almacén de datos. El resto de celdas del DWH serán derivadas de esta.

4.1.2.3. Estrella consumo de medicamentos

En el caso del consumo de medicamentos, la granularidad elegida es la del número de recetas dispensadas por día, no vemos necesario detallar los medicamentos que se consumen en cada hora.

Algunas de las dimensiones por las que analizaremos la información, coinciden con las de la estrella de las consultas.

Dimensiones

Fecha de dispensación

Es la fecha en la que se dispensa la receta. No contabilizaremos aquellas recetas que no hayan sido dispensadas. La estructura jerárquica es similar a la de la estrella anterior, si bien en este caso no tendremos el nivel 'Hora'.

Año
Mes
Semana
Día

Paciente

Es el paciente para el que se hace la receta. Los atributos serán los mismos que en la estrella de las consultas

Médico

Es el médico que realiza la receta. Tampoco definimos atributos adicionales, tal y como pasaba con las consultas.

Producto

Se trata del medicamento que se ha recetado. De esta dimensión identificaremos tanto el producto recetado como el medicamento que contiene dicho producto; y si éste es comercial o genérico.

Enfermedad

La última dimensión por la que podremos analizar el consumo de medicamentos es la enfermedad para que la que se receta el medicamento. Únicamente identificaremos el nombre de la misma.

Indicadores

En este caso mediremos el número de medicamentos dispensados.

Celda

La definida registrará los medicamentos dispensados por día, el resto de celdas serán derivadas de ésta.

4.1.2.4 Estrella bajas laborales

Para las bajas laborales la granularidad será la misma que para los medicamentos, no nos aporta más información el registrar la hora en la que se concedió la baja.

Dimensiones

Fecha inicio de la baja

Las bajas las contabilizaremos por la fecha en la que se finalizó. Por ello, sólo tendremos en cuenta aquellas que ya hayan finalizado. De esta forma no será necesario volver a calcular periodos antiguos para disponer de la información actualizada.

Paciente

Es el paciente al que se le concede la baja. Los atributos son los mismos que en las otras estrellas.

Médico

Es el médico que concede la baja. Tampoco definimos atributos adicionales, tal y como pasaba con las consultas y los medicamentos.

Enfermedad
La última dimensión por la que podremos analizar las bajas es por la enfermedad que provoca la baja. De esta forma sabremos qué enfermedades están provocando más bajas.

Indicadores
Aquí contabilizaremos tanto el número de bajas como la duración en días de las mismas.

Celdas
La celda definida será la baja junto con la duración de la misma, ya que como un paciente no puede tener activas dos bajas de forma simultánea, no se pueden finalizar dos bajas el mismo día.

4.1.3. Diagrama del Modelo Conceptual

Tras establecer los hechos y los indicadores; las dimensiones y los atributos y niveles de éstas; y las celdas de cada una de las estrellas; en la figura 5 podemos ver el diseño conceptual que hemos definido.

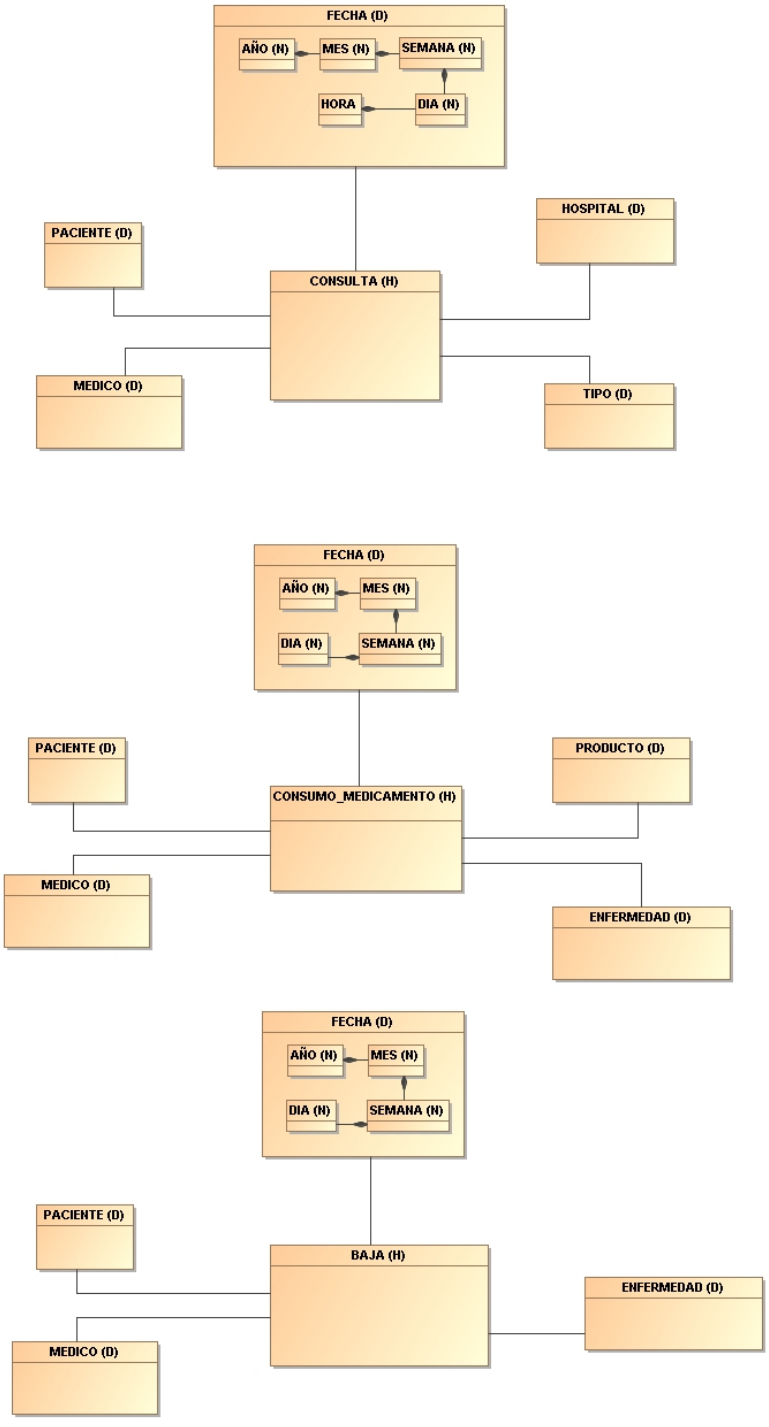

Figura 5. Modelo conceptual del almacén de datos.

4.2. Modelo Lógico

Una vez establecido el modelo conceptual, realizaremos el diseño lógico del DWH. A partir de los hechos y las dimensiones establecidas en el apartado anterior, crearemos las tablas de hecho y las tablas de las dimensiones, junto con los campos necesarios para contener los atributos definidos.

Cada tabla de hecho de cada una de las estrellas tendrá un campo identificador para cada registro. Además tendrán campos que serán claves ajenas a la clave primaria de cada una de las dimensiones. Estas claves ajenas formarán una clave alternativa de la tabla. Por último, tendremos un campo para cada uno de los indicadores definidos en cada estrella.

En las tablas de las dimensiones tenemos una clave primaria; y el resto de campos corresponden a los identificadores necesarios de cada dimensión. Los datos de cada dimensión los podríamos obtener directamente de las tablas de la base de datos operacional relacional, pero el diseño de DWH se hace para agilizar al máximo las consultas; por lo que se guarda información redundante en las tablas de dimensión.

En algunas dimensiones, a parte de la clave primaria de la misma, también guardaremos la clave primaria del elemento correspondiente en la base de datos relacional. Esto lo haremos para poder ampliar en un futuro los indicadores de cada dimensión.

Dimensión Paciente.

El identificador de la dimensión Paciente no coincide con el identificador de la tabla paciente de la base de datos. Cada vez que se consuma un medicamento por un paciente, si ha variado la edad de éste, se añadirá un registro nuevo a la tabla DWHPACIENTE. Guardamos el identificador de la tabla paciente por si en un futuro se desea ampliar los atributos de la dimensión. De esta forma tendremos un registro de los hechos y la edad del paciente al suceder éstos.

Dimensión Médico.

De la misma forma que hemos hecho con el paciente, en la dimensión Médico también hemos creado un identificador distinto al identificador de la tabla Medico de la BD. Actualmente esta dimensión no tiene ningún atributo, pero si en un futuro deseamos añadir atributos a la dimensión, tendremos el indicador de la tabla de la base de datos relacional.

Dimensión Tipo Consulta.

En la tabla de esta dimensión almacenaremos el identificador del tipo de consulta y la descripción. Estos tipos no variarán en la base de datos operacional, por lo que no es necesario tener un identificador distinto tal y como hemos establecido para las dimensiones de Paciente o Médico.

Dimensión Hospital

En esta tabla tampoco crearemos un identificador distinto al que tenemos en la base de datos operacional. Tenemos un campo para el nombre y otros tres para definir la ubicación: el código postal, la repetición y el país.

Dimensión enfermedad

Las enfermedades conocidas tampoco variarán su identificador en la base de datos operacional. Además del identificador, tendremos un campo para el nombre de la misma.

Dimensión Producto

En esta dimensión, además del identificador del producto, tendremos un campo para el nombre del mismo y otro con el nombre del medicamento al que corresponde dicho producto. No guardamos el identificador del medicamento, sólo el nombre, ya que es lo que necesitamos para mostrar los

resultados en las estadísticas. En caso de necesitar en un futuro más datos de los medicamentos, lo podremos obtener a partir del identificador del producto. Por último, tendremos un campo con la descripción del tipo de producto. No guardamos el identificador del tipo por la misma razón que hemos comentado para el caso del medicamento.

Dimensión Fecha

Tendremos un identificador para cada registro y el resto de campos en los que se ven reflejados los distintos niveles de esta dimensión. Esta es la única dimensión que es compartida por las tres tablas de hecho, y cuya dimensión tiene una granularidad distinta para una de ellas. Mientras en el caso de las consultas la granularidad es de la hora, para las otras dos es el día. Para resolver esta situación, establecemos que los registros de la tabla de hecho de los medicamentos y las bajas se relacionarán con un registro de esta tabla con la hora '00:00'. Haciendo esto permitimos que entre las distintas estrellas se pueda realizar la operación *Drill-across*. Esta operación permite, en una herramienta OLAP, cambiar la tabla de hecho por la que estamos analizando los datos. Esto será posible si el resto de las dimensiones que se están utilizando en el análisis, están disponibles en la tabla de hecho por la que se desea realizar la operación.

Además, añadiremos dos campos para tener la descripción del mes y del día; y un tercer campo para indicar el orden del día en la semana. Es posible que trabajemos con herramientas de inteligencia de negocio que ordenen los días de la semana como si fuera una cadena de texto; con este campo podremos ordenarlos de forma natural, no alfabéticamente.

Tabla de hecho Consulta

En esta tabla, además de las claves ajenas a las dimensiones, tendremos el campo numConsultas para registrar el número de consultas.

Tabla de hecho Medicamentos.

Además de tener las claves ajenas a las dimensiones, tendremos el campo numMedicamentos para registrar el número de medicamentos dispensados.

Tabla de hecho Baja

Por último, en esta tabla tendremos dos campos para registrar los dos indicadores del hecho: el campo numBajas para indicar el número de bajas finalizadas; y el campo duración dónde se registrará la duración media en días de las mismas.

4.2.1. Diagrama modelo lógico

En la figura 6, podemos ver el diagrama que hemos generado tras plasmar todo lo establecido en el apartado anterior.

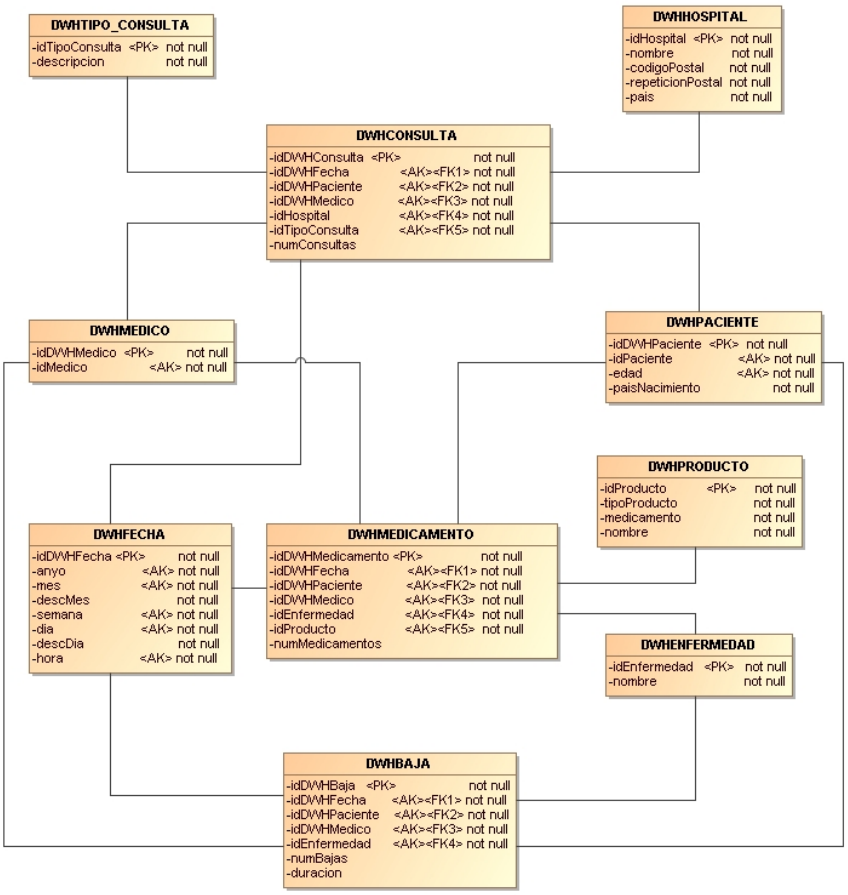

Figura 6. Diseño lógico del almacén de datos.

4.3. Diseño Físico

En la implementación de las tablas del DWH también establecemos las restricciones de integridad que hemos definido en el modelo lógico. De la misma forma que hemos hecho en la implementación de la base de datos operacional, algunos de los errores en la introducción de la información los controlamos por código, pero la gran mayoría recaerán sobre las reglas de integridad. Los errores en la introducción de datos en el DWH serán mucho menores, ya que en lugar de ser introducidos por los usuarios, serán procesos ejecutados periódicamente los que alimentarán las tablas de forma desatendida. Estos procesos de extracción, transformación y carga de datos, podrán ser programados para que se ejecuten de forma automática. Dichos procesos ETL se simplifican al trabajar con una única fuente de datos, como es el caso de este proyecto.

4.3.1. Creación de las tablas

De la misma forma que hemos hecho al crear las tablas de la BD, en las sentencias de creación estableceremos las restricciones de integridad que hemos definido en el diseño lógico. Tampoco incluiremos en la memoria el código que hemos desarrollado para crear las tablas, éste se puede consultar en el archivo *SCRIPTS CREACION TABLAS DWH.sql* de la carpeta *"Producto\3. Diseño fisico\Almacen de datos"* del proyecto.

4.3.2. Procedimientos almacenados

Como hemos comentado anteriormente, en el almacén de datos no tendremos procedimientos de alta, modificación y baja para cada una de las tablas; en lugar de estos, tendremos otros que se encargarán de extraer la información de la base de datos operacional y cargarla en las tablas del almacén de datos. Recordamos que la base de datos operacional será la que hemos creado en los apartados anteriores de este proyecto, siendo la única fuente de datos de este DWH.

Tendremos tres tipos de procedimientos diferentes:

1. Por una parte tendremos los procedimientos en los que comprobaremos si existe un registro en una tabla de una dimensión, en caso afirmativo devolverá el identificador de dicho registro. Si por el contrario no existe, lo creará y devolverá el identificador. Dentro de este tipo, también están los tres procedimientos encargados de realizar la misma tarea para las tablas de los hechos, en este caso, si existe el registro, se incrementará el valor del indicador correspondiente. Estos procedimientos empiezan por "SP_DWH_" y se utilizarán en el procedimiento que se encarga de realizar el proceso ETL.

2. Después tendremos los procedimientos que se encargan de realizar la carga de los datos en las dimensiones y las tablas de hecho. Tendremos uno para cada estrella y empiezan por "SP_ETL_". Desde estos procedimientos se obtendrán los datos de las tablas de la base de datos operacional; y a continuación se invocarán los procedimientos comentados anteriormente (los SP_DWH_), para realizar la carga en las tablas del DWH.

3. Por último tenemos los procedimientos para obtener el resultado de las consultas. Estos procedimientos de selección empiezan por "SP_SEL_". Sólo hemos creado tres consultas de ejemplo. Estos procedimientos muestran por la salida DBMS el resultado de la consulta.

La generación de las consultas contra la base de datos a través de procedimientos almacenados es poco útil y únicamente sirve para la comprobación de que los datos se han introducido correctamente. En el apartado 4.4 "Explotación del almacén de datos" veremos la creación de un esquema OLAP mediante la herramienta Pentaho Schema Workbench y el posterior análisis de los datos con esta herramienta.

A continuación vamos a ver las definiciones de todos los procedimientos que hemos creado para la carga del DWH.

CARGA DIMENSIONES	SP_DWH_COMPRUEBA_ENFERMEDAD
Descripción	Procedimiento encargado de cargar los datos en la tabla de dimensión ENFERMEDAD. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe devuelve el valor de la clave primaria de dicho registro, si no existe lo crea y devuelve el identificador.
Parámetros de entrada	p_idEnfermedad, identificador de la enfermedad. p_nombre, nombre de la enfermedad.
Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

CARGA DIMENSIONES	SP_DWH_COMPRUEBA_FECHA

Descripción	Procedimiento encargado de cargar los datos en la tabla de dimensión FECHA. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe devuelve el valor de la clave primaria de dicho registro, si no existe lo crea y devuelve el identificador.
Parámetros de entrada	p_anyo, p_mes, p_semana, p_dia, p_hora.
Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

CARGA DIMENSIONES	SP_DWH_COMPRUEBA_HOSPITAL
Descripción	Procedimiento encargado de cargar los datos en la tabla de dimensión HOSPITAL. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe devuelve el valor de la clave primaria de dicho registro, si no existe lo crea y devuelve el identificador.
Parámetros de entrada	p_idHospital, p_nombre, p_codigoPostal, p_repeticionPostal, p_pais.
Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

CARGA DIMENSIONES	SP_DWH_COMPRUEBA_MEDICO
Descripción	Procedimiento encargado de cargar los datos en la tabla de dimensión MEDICO. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe devuelve el valor de la clave primaria de dicho registro, si no existe lo crea y devuelve el identificador.
Parámetros de entrada	p_idMedico.

Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que el parámetro de entrada, no sea nulo.

CARGA DIMENSIONES	SP_DWH_COMPRUEBA_PACIENTE
Descripción	Procedimiento encargado de cargar los datos en la tabla de dimensión PACIENTE. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe devuelve el valor de la clave primaria de dicho registro, si no existe lo crea y devuelve el identificador.
Parámetros de entrada	p_idPaciente, p_edad, edad del paciente en el momento del hecho. p_paisNacimiento.
Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

CARGA DIMENSIONES	SP_DWH_COMPRUEBA_PRODUCTO
Descripción	Procedimiento encargado de cargar los datos en la tabla de dimensión PRODUCTO. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe devuelve el valor de la clave primaria de dicho registro, si no existe lo crea y devuelve el identificador.
Parámetros de entrada	p_idProducto, p_nombre, nombre del producto. p_medicamento, nombre del medicamento. p_tipoProducto, descripción del tipo de producto.
Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

CARGA DIMENSIONES	SP_DWH_COMPRUEBA_TIPO_CONSULTA
Descripción	Procedimiento encargado de cargar los datos en la tabla de dimensión TIPO CONSULTA. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe devuelve el valor de la clave primaria de dicho registro, si no existe lo crea y devuelve el identificador.

Parámetros de entrada	p_idTipoConsulta, identificador de la enfermedad. p_descripcion, descripción del tipo de consulta.
Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

CARGA HECHO	**SP_DWH_COMPRUEBA_CONSULTA**
Descripción	Procedimiento encargado de cargar los datos en la tabla de hecho DWHCONSULTA. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe suma el valor del parámetro p_numConsultas en el campo numConsultas de la tabla. Si no existe el registro, lo crea con los datos pasados como parámetro y devuelve el identificador.
Parámetros de entrada	p_idDWHFecha, p_idDWHPaciente, p_idDWHMedico, p_idHospital, p_idTipoConsulta, p_numConsultas.
Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

CARGA HECHO	**SP_DWH_COMPRUEBA_MEDICAMENTO**
Descripción	Procedimiento encargado de cargar los datos en la tabla de hecho DWHMEDICAMENTO. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe suma el valor del parámetro p_numMedicamentos en el campo numMedicamentos de la tabla. Si no existe el registro, lo crea con los datos pasados como parámetro y devuelve el identificador.
Parámetros de entrada	p_idDWHFecha, p_idDWHPaciente, p_idDWHMedico, p_idEnfermedad, p_idProducto, p_numMedicamentos.

Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

CARGA HECHO	**SP_DWH_COMPRUEBA_BAJA**
Descripción	Procedimiento encargado de cargar los datos en la tabla de hecho DWHBAJA. Para ello comprueba si existe ya un registro con los datos pasados como parámetro. Si existe suma el valor del parámetro p_numBajas y p_duracion en los campos numBajas y duración de la tabla. Si no existe el registro, lo crea con los datos pasados como parámetro y devuelve el identificador.
Parámetros de entrada	p_idDWHFecha, p_idDWHPaciente, p_idDWHMedico, p_idEnfermedad, p_numBajas, p_duracion.
Parámetros de salida	p_ID INTEGER, devuelve el valor del identificador del registro en la tabla. p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

PROCESO ETL	**SP_ETL_CARGA_DWHCONSULTA**
Descripción	Procedimiento encargado de realizar el proceso de carga de los datos de las consultas. Para ello recoge los datos de las tablas de la base de datos operacional, y registro a registro invoca a los procedimientos anteriores para alimentar las tablas de las dimensiones y las de hechos. Como parámetros de entrada tenemos dos fechas. La carga se realizará para aquellas consultas creadas en ese rango de fechas. Antes de realizar la carga de los datos, se borran los registros de la tabla DWHCONSULTA para que un mismo hecho no se registre dos veces.
Parámetros de entrada	p_fechaIni, fecha inicial de los datos a cargar. p_fechaFin, fecha final.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

PROCESO ETL	**SP_ETL_CARGA_DWHMEDICAMENTO**

Descripción	Procedimiento encargado de realizar el proceso de carga de los datos del consumo de medicamentos. Para ello recoge los datos de las tablas de la base de datos operacional, y registro a registro invoca a los procedimientos anteriores para alimentar las tablas de las dimensiones y las de hechos. Como parámetros de entrada tenemos dos fechas. La carga se realizará para aquellos medicamentos dispensados en ese rango de fechas. Antes de realizar la carga de los datos, se borran los registros de la tabla DWHMEDICAMENTO para que un mismo hecho no se registre dos veces.
Parámetros de entrada	p_fechaIni, fecha inicial de los datos a cargar. p_fechaFin, fecha final.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

PROCESO ETL	**SP_ETL_CARGA_DWHBAJA**
Descripción	Procedimiento encargado de realizar el proceso de carga de los datos de las bajas laborales. Para ello recoge los datos de las tablas de la base de datos operacional, y registro a registro invoca a los procedimientos anteriores para alimentar las tablas de las dimensiones y las de hechos. Como parámetros de entrada tenemos dos fechas. La carga se realizará para aquellas bajas finalizadas en ese rango de fechas. Antes de realizar la carga de los datos, se borran los registros de la tabla DWHBAJA para que un mismo hecho no se registre dos veces.
Parámetros de entrada	p_fechaIni, fecha inicial de los datos a cargar. p_fechaFin, fecha final.
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Que ninguno de los parámetros de entrada, sea nulo.

SELECCIÓN	**SP_SEL_URGENCIAS_MES**
Descripción	Procedimiento que ejecuta una sentencia de selección para obtener el número de urgencias que se han producido por mes. No acepta parámetros de entrada, la consulta es para todos los registros. Es un ejemplo de cómo se pueden obtener datos de las tablas del DWH. De hecho, tiene todas las tablas de las dimensiones vinculadas, cuando no sería necesario para realizar dicha consulta.
Parámetros de entrada	No tiene
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Ninguna

SELECCIÓN	**SP_SEL_EDAD_MAS_MEDICAMENTOS**

Descripción	Procedimiento que ejecuta una sentencia de selección para obtener la edad en la que se consumen más medicamentos. No acepta parámetros de entrada, la consulta es para todos los registros. Es un ejemplo de cómo se pueden obtener datos de las tablas del DWH. De hecho, tiene todas las tablas de las dimensiones vinculadas, cuando no sería necesario para realizar dicha consulta.
Parámetros de entrada	No tiene
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Ninguna

SELECCIÓN	SP_SEL_TIEMPO_MEDIO_BAJA
Descripción	Procedimiento que ejecuta una sentencia de selección para obtener el tiempo medio que los pacientes están de baja. No acepta parámetros de entrada, la consulta es para todos los registros. Es un ejemplo de cómo se pueden obtener datos de las tablas del DWH.
Parámetros de entrada	No tiene
Parámetros de salida	p_rstCodigo INTEGER, devuelve 0 en caso de que haya finalizado correctamente. En caso de que el procedimiento no haya podido completar su ejecución, devuelve el código del error. p_rstMensaje VARCHAR2, devuelve OK si la ejecución finaliza correctamente. En caso de error, devuelve un mensaje con el error que haya sucedido.
Comprobaciones	Ninguna

4.4. Explotación del almacén de datos

En este apartado veremos cómo podemos explotar la información de nuestro almacén de datos utilizando una herramienta OLAP. En el apartado anterior hemos visto el diseño de un modelo multidimensional para dar respuesta a las necesidades de nuestro cliente. Ahora diseñaremos un modelo ROLAP (*relational* OLAP) con la herramienta Pentaho Schema Workbench (PSW); para después poder hacer un análisis de los datos de las consultas, del consumo de medicamentos y las bajas laborales.

Una herramienta ROLAP es una de las implementaciones que existen en el mercado de herramientas OLAP. Se distingue por trabajar sobre un SGBD relacional. Recibe consultas multidimensionales que traduce a SQL y las ejecuta directamente sobre el SGBD.

Dentro del producto Pentaho Business Analytics vamos a utilizar la herramienta PSW para crear y publicar el esquema OLAP que después utilizaremos en Saiku Analytics para analizar los datos. Saiku Analytics es otra herramienta de la suite Pentaho.

4.4.1. Esquema en Pentaho Schema Workbench

Lo primero que hacemos es crear un esquema que incluya un cubo para cada una de las estrellas que hemos definido en el apartado 4.2 "Modelo lógico", con la herramienta PSW. En el esquema definiremos todos los hechos, las dimensiones, y las medidas. A la hora de crear un cubo en PSW, podemos utilizar las mismas dimensiones que tenemos en nuestro almacén de datos, obviar alguna que no nos interese en nuestro análisis o bien crear más elementos que sólo existirán en el esquema OLAP.

De esta forma tenemos que definir lo siguiente:

1. Cuáles son las tablas de los hechos, qué campos corresponden a las medidas que se utilizarán o qué tipo de función se desea utilizar para resumir las medidas (suma, media, cuenta, etc.).

2. También definiremos las tablas de las dimensiones, las claves primarias de estas tablas y las claves ajenas que relacionan estas tablas con las tablas de hechos.

3. De las dimensiones definiremos los distintos niveles, si los hubiera; y la jerarquía correspondiente.

En definitiva, creamos un esquema con la definición completa de los tres cubos que queremos modelar para que después, con la herramienta Saiku Analytics, el usuario encargado de analizar los datos pueda crearse las consultas de una forma autónoma. La herramienta es muy intuitiva, no siendo necesarios grandes conocimientos informáticos para poder utilizarla.

Las dimensiones que crearemos serán las mismas que hemos establecido en el diseño del almacén de datos, a excepción de la dimensión Fecha, a partir de la cual vamos a crear tres jerarquías distintas.

Uno de los análisis que nos pueden interesar, es el número de urgencias que tenemos por mes o por día de la semana. De esta forma, tras el análisis, se pueden reforzar los turnos de los días con más número de urgencias.

Si en la herramienta Saiku Analytics necesitamos sacar una estadística a partir de un nivel inferior de una dimensión, sin que agrupe los resultados por el nivel superior, tenemos dos soluciones:

1. Modificar manualmente la consulta MDX.

2. Crear una nueva jerarquía dentro de la dimensión, en la que ese nivel sea el nivel superior.

Como el mes o el día de la semana están en un nivel inferior al año en la dimensión fecha, necesitamos tomar una de estas dos opciones. La consulta MDX es la sentencia mediante la cual la herramienta extraerá los datos de la base de datos (después de traducirla a SQL). La modificación de estas consultas requiere unos conocimientos informáticos que un usuario encargado de analizar los datos no suele tener. Para facilitar el trabajo de éstos elegimos la segunda opción, la creación de una nueva dimensión.

De esta forma, añadiremos una jerarquía FechaDia en la que únicamente tendremos un nivel y corresponderá al campo de la tabla DWHFecha en el que tenemos la descripción del día de la semana. La segunda jerarquía será FechaMes, que incluirá únicamente un nivel con la descripción del mes.

En la figura 7mostramos el esquema generado con la herramienta PSW. En él podemos ver los tres cubos: CONSULTA, MEDICAMENTO Y BAJA_LABORAL.

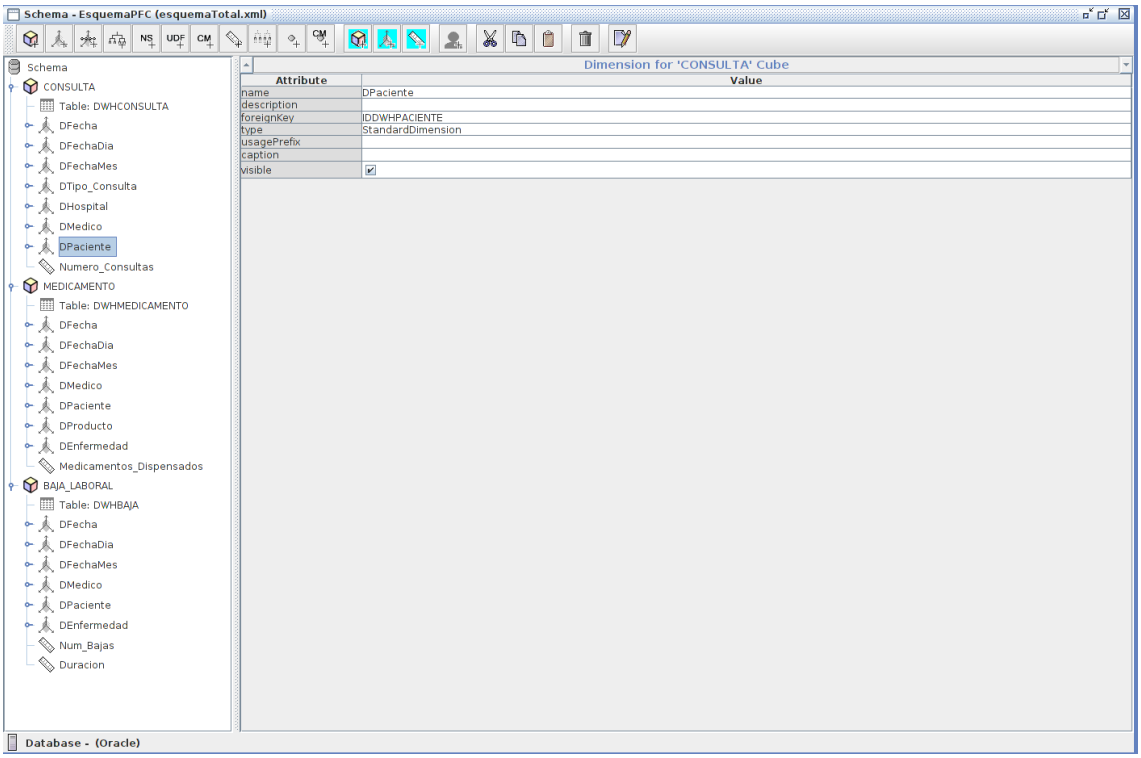

Figura 7. Esquema OLAP creado con Pentaho Schema Workbench.

De cada cubo vemos la tabla donde están los datos, las distintas dimensiones y las medidas. En el cubo de las bajas laborales tenemos dos medidas, una de ellas será la suma de las bajas laborales y la otra la media de la duración de las mismas:

En la figura 8 podemos ver un detalle de la dimensión Fecha, con las tres jerarquías y los distintos niveles distintos niveles de cada una de ellas.

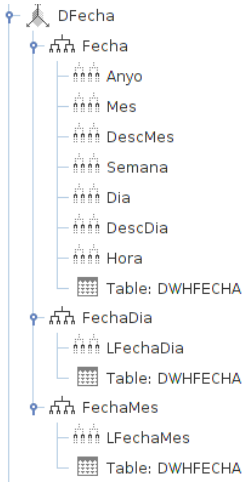

Figura 8. Dimensión Fecha.

En los niveles LFechaDia y LFechaMes es donde indicamos que el orden de los elementos debe hacerse por un campo distinto al que se muestra. Lo podemos ver en la figura 9 en el atributo "ordinalColumn".

Attribute	
name	LFechaDia
description	
table	
column	DESCDIA
nameColumn	
parentColumn	
nullParentValue	
ordinalColumn	NUMDIA
type	String
internalType	String
uniqueMembers	☐
levelType	Regular
hideMemberIf	Never
approxRowCount	
caption	
captionColumn	
formatter	
visible	☑

Figura 9. Nivel de la jerarquía FechaDia.

4.4.2. Análisis de los datos

Una vez creado y publicado el esquema, podemos crear los análisis que necesitemos con la herramienta Saiku Analytics. Empezaremos por analizar las urgencias médicas.

Para ello primero elegimos el cubo que deseamos utilizar: CONSULTA.

A continuación arrastramos la medida "Numero_Consultas" en la sección *Columnas* y en la *Filas* arrastramos el campo LFechaMes de la jerarquía FechaMes.

Por último, arrastramos la dimensión "Tipos_Consulta" a la sección *Filtro* y elegimos únicamente las consultas de tipo "URGENCIA". En tres simples pasos tenemos el análisis creado.

En la figura 10tenemos el análisis de los meses en los que más urgencias médicas ocurren. Como podemos ver, los meses están ordenados correctamente y no en orden alfabético.

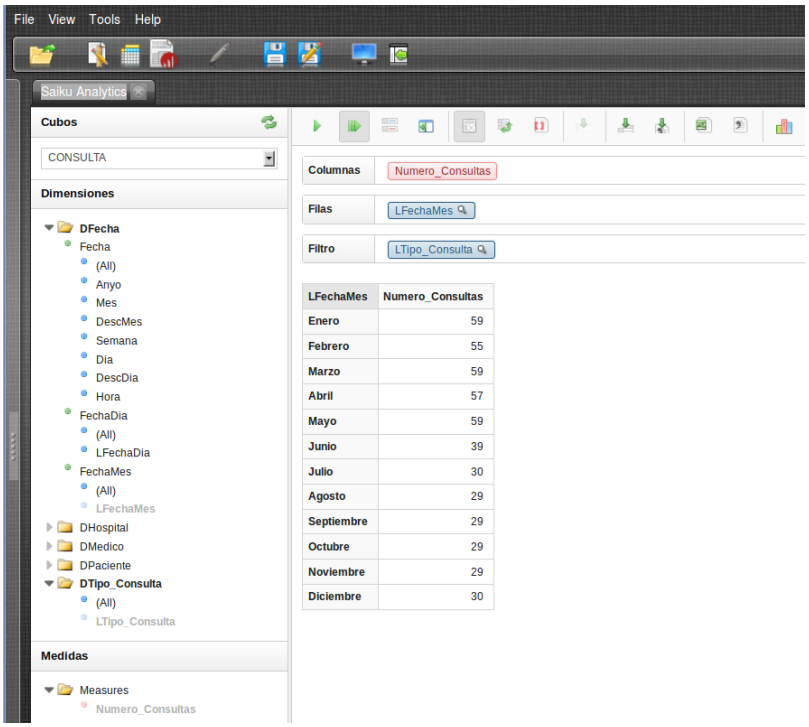

Figura 10. Análisis de las urgencias por mes.

También podemos utilizar un gráfico para analizar la evolución a lo largo del año, lo vemos en la figura 11.

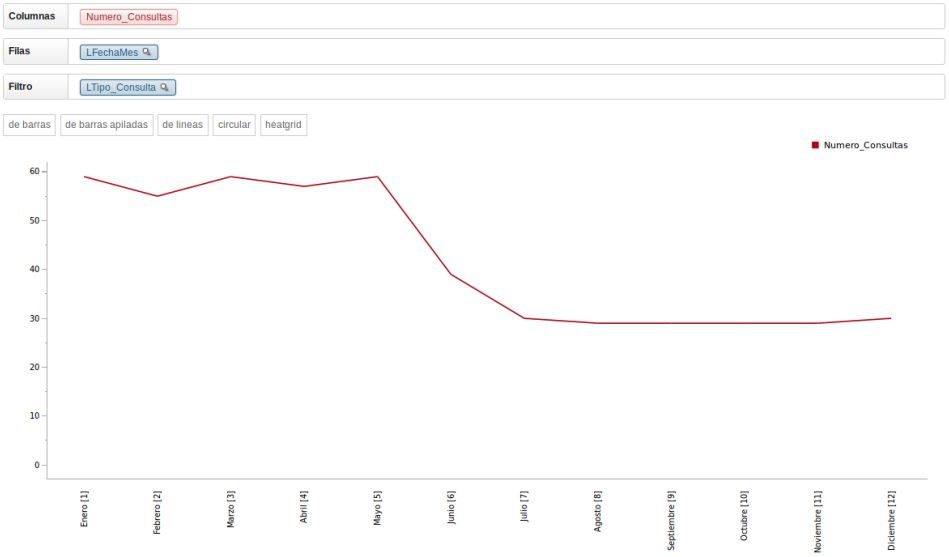

Figura 11. Gráfico de las urgencias por mes.

Si cambiamos la dimensión y en lugar de indicar en las filas LFechaMes, arrastramos LFechaDia; en la figura 12podemos ver en qué día de la semana se dan un mayor número de urgencias.

Columnas	Numero_Consultas
Filas	LFechaDia
Filtro	LTipo_Consulta

LFechaDia	Numero_Consultas
Lunes	73
Martes	73
Miércoles	72
Jueves	71
Viernes	71
Sábado	71
Domingo	73

Figura 12. Urgencias por día de la semana.

Cambiando el cubo podemos analizar el consumo de medicamentos por edad del paciente (figura 13) o bien verlo en forma de gráfico (figura 14).

Columnas	Medicamentos_Dispensados
Filas	EdadPaciente
Filtro	

EdadPaciente	Medicamentos_Dispensados
30	438
31	488
32	28
35	54

Figura 13. Consumo de medicamentos por edad del paciente.

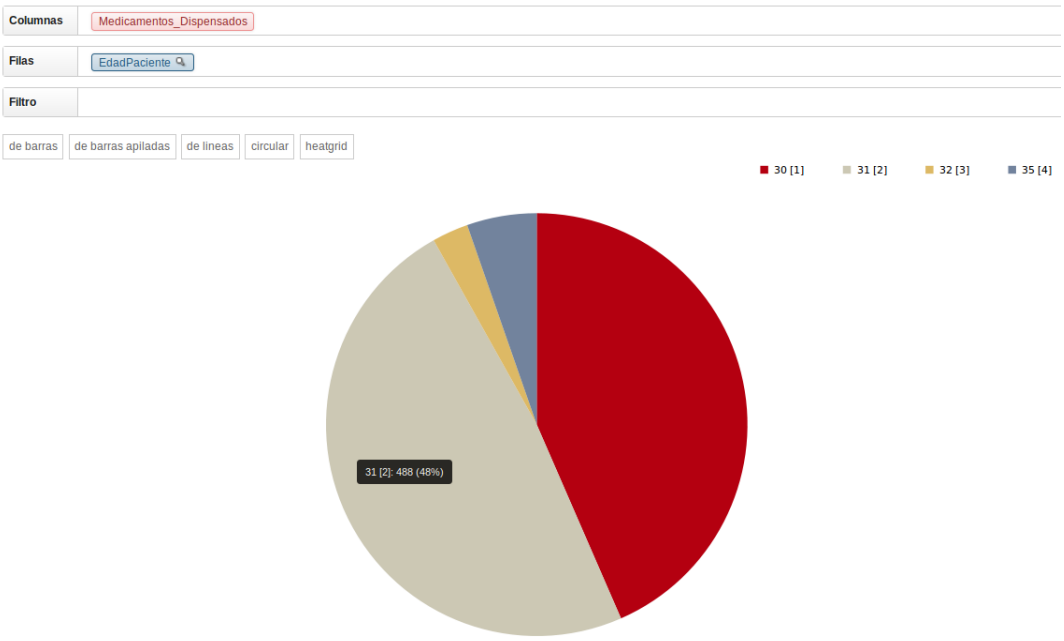

Figura 14. Gráfico del consumo de medicamentos por edad.

Para el análisis de las bajas laborales, podemos utilizar una de las dos medidas o las dos simultáneamente. En la figura 15 vemos a qué edad se suceden más bajas y la media de la duración para cada edad. En la figura 16 vemos un gráfico de barras de los resultados.

EdadPaciente	Num_Bajas	Duracion
30	15	32,733
31	17	50,588
32	1	29
35	2	53,5

Figura 15. Análisis de bajas laborales por edad.

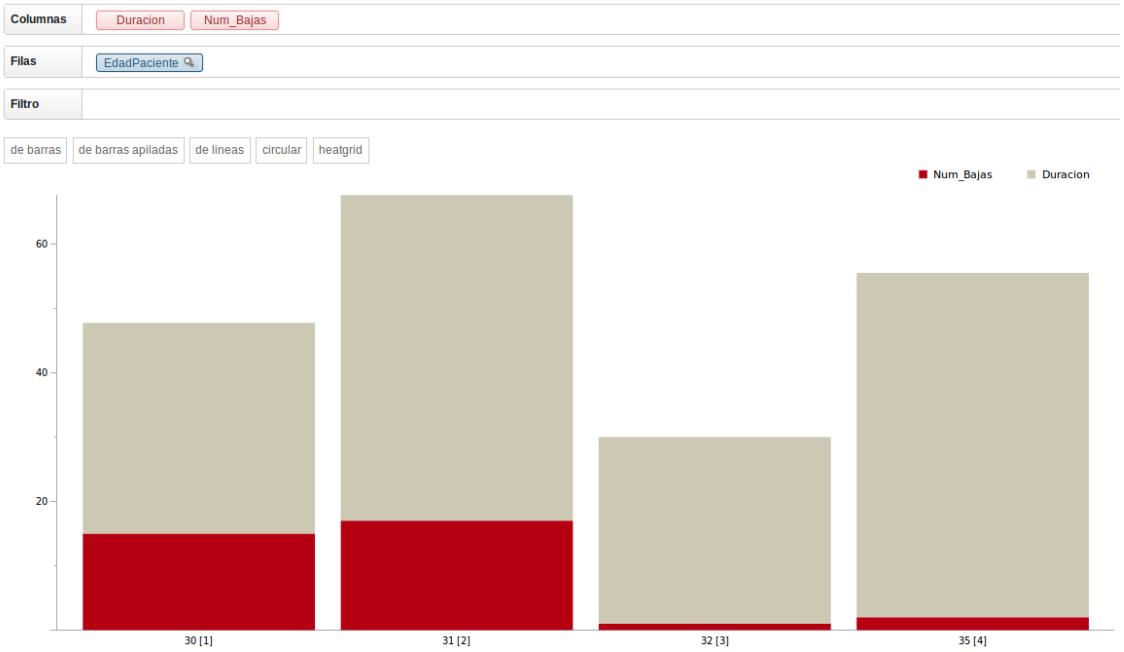

Figura 16. Gráfico de las bajas laborales por edad.

5. Carga inicial y pruebas

Una parte muy importante para llevar un proyecto a buen fin, es el hecho de contar una gran batería de datos y diseñar un buen plan de pruebas. Para ello, hemos preparado un conjunto importante de información para alimentar cada una de las tablas; para así poder realizar pruebas en la base de datos y que la información extraída sea reconocible en el entorno en que el que se va a utilizar. Además, esta información cargada en la base de datos operativa, nos servirá posteriormente para realizar el proceso ETL en el DWH y completar las pruebas del mismo.

En primer lugar hemos preparado una carga inicial de las tablas principales. Tablas como la de PAIS, POBLACION, MEDICO, PACIENTE, FARMACEUTICO, HOSPITAL o ENFERMEDAD. Estas cargas las hemos realizado con sentencias de inserción directamente. Estas cargas se encuentran en el archivo *"1. SCRIPTS CARGA INICIAL.SQL"* dentro de la carpeta *"Producto\2. Carga de datos\"*.

A continuación hemos preparado una carga para el resto de las tablas a través de la utilización de los procedimientos de alta. Estas cargas se encuentran en el fichero *"2. SCRIPTS CARGA CON PROCEDIMIENTOS.SQL"* dentro de la misma carpeta que el fichero anterior. Al realizar esta carga, al mismo tiempo estamos probando la correcta ejecución de todos los procedimientos de alta.

Posteriormente hemos probado los procedimientos de modificación y baja, forzando los errores para ver cómo obtenemos los reportes de los mismos, ya sea a través del código del procedimiento o de las restricciones referenciales establecidas en las tablas. Estas pruebas están en el fichero *"EJECUCIÓN PRUEBAS.SQL"* que se encuentra en la carpeta *"Producto\5. Pruebas\".*

Una vez cargados los datos en las tablas, hemos probado el proceso ETL del DWH. Este fichero es el siguiente: *"Producto\3. Diseño fisico\Almacen de datos\Proceso ETL\ PROCESO_ETL.SQL"*

Hemos comprobado la carga correcta del DWH a través de los procedimientos de selección del DWH; de los análisis Pentaho; y realizando consultas directamente contra la base de datos.

Con cada ejecución de los procedimientos de alta para la carga de datos; con cada prueba de los procedimientos de baja y modificación; y con cada la ejecución de los procesos ETL del DWH; se inserta un registro en la tabla LOGS. Como hemos explicado en el apartado 3.3 "Diseño físico", hemos preparado todos los procedimientos almacenados para que, además de devolver el resultado de la ejecución en los dos parámetros de salida, inserte un registro en la tabla LOGS con este resultado. En esta tabla se indican los parámetros utilizados y, en caso de error, el código de éste. En el archivo *"Producto\5. Pruebas\ EXPORT LOGS.txt"* hemos hecho una exportación de la tabla LOGS dónde se pueden ver los resultados de las pruebas realizadas.

Las pruebas unitarias realizadas a medida que se desarrollaban los procedimientos no se han incluido en ningún fichero, pero el resultado de la ejecución de éstas sí se puede ver en el fichero exportado de la tabla LOGS.

6. Valoración económica del proyecto

En este apartado vamos a hacer una valoración del coste total del proyecto, incluyendo tanto el trabajo del equipo que ha intervenido como la infraestructura necesaria.

6.1 Coste del trabajo realizado por el equipo del proyecto

Como hemos visto en el apartado 1.4 “Planificación del trabajo”, el coste de cada tarea está indicado en número de jornadas y hemos establecido una dedicación media de 4 horas por cada jornada laboral.

Vamos a valorar las distintas tareas, dependiendo del perfil de trabajador necesario para la realización de cada una de ellas. A continuación veremos cada perfil incluyendo el coste de su jornada laboral, contando que es de 4 horas.

· Jefe de proyecto: 280 €
· Consultor sénior interno: 210 €
· Técnico interno: 145 €

El jefe de proyecto ha realizado las tareas de planificación, análisis previo y documentación. En la planificación no hemos contemplado las reuniones periódicas que haya podido tener el jefe de proyecto con el cliente; ni las reuniones de seguimiento y control con el resto del equipo del proyecto; ya que, aunque contemplamos tres perfiles, todas las tareas las ha realizado la misma persona.

El consultor interno se ha encargado de las tareas de análisis y diseño.

Y por último, el técnico interno. En este perfil incluimos tanto el técnico de sistemas como el técnico de desarrollo. Por tanto se incluyen las tareas de preparación del entorno de trabajo y las de desarrollo de producto.

Como hemos comentado en el apartado 1.4 “Planificación del trabajo”, no se ha planificado la implantación ni el mantenimiento posterior de la solución.

6.1.1 Tareas realizadas

A continuación detallamos las tareas realizadas, indicando las jornadas necesarias para su finalización y el perfil que las realizó.

El coste de configuración y de puesta en marcha de la infraestructura los incluimos en el apartado siguiente, 6.2 “Coste infraestructura”.

TAREAS	JORNADAS	PERFIL
1 PAC1 - Plan de Trabajo		
1.1 Lectura de la documentación	1	Jefe de Proyecto
1.2 Preparación del entorno de trabajo	1	
1.2.1 Creación de la máquina virtual	1	Técnico
1.2.2 Instalación de Oracle y SQL Developer	0	Técnico
1.3 Análisis inicial de requerimientos	2	Jefe de Proyecto
1.4 Diseño conceptual preliminar	1	Jefe de Proyecto
1.5 Elaboración del plan de trabajo	2	Jefe de Proyecto
1.6 Entrega del plan de trabajo	0	Jefe de Proyecto
2 PAC2	35	
2.1 Revisión planificación	1	Jefe de Proyecto
2.2 Análisis de requerimientos	2	Consultor

2.3 Diseño de la BD	4	
2.3.1 Diseño conceptual	2	Consultor
2.3.2 Diseño lógico	2	Consultor
2.4 Creación de la BD	11	
2.4.1 Scripts de creación de tablas	8	Técnico
2.4.2 Carga de datos de prueba	2	Técnico
2.4.3 Pruebas unitarias	1	Técnico
2.5 Creación del sistema de monitorización	4	
2.5.1 Creación de disparadores para alimentar la tabla de logs	3	Técnico
2.5.2 Pruebas unitarias	1	Técnico
2.6 Desarrollo Procedimientos para acceso a los datos	12	
2.6.1 Scripts de creación de los procedimientos almacenados	11	Técnico
2.6.2 Pruebas unitarias	1	Técnico
2.7 Entrega PAC2	0	
3 PAC3	27	
3.1 Revisión planificación	0	Jefe de Proyecto
3.2 Análisis y diseño del DWH a desarrollar	8	Consultor
3.3 Creación de las tablas del DWH	5	
3.3.1 Scripts de creación de tablas	3	Técnico
3.3.2 Scripts de carga de tablas de dimensiones	2	Técnico
3.4 Creación de la carga del DWH	7	Técnico
3.4.1 Scripts de carga de la tabla de estadísticas	5	Técnico
3.4.2 Pruebas unitarias	2	Técnico
3.5 Pruebas finales de la BD	4	Consultor y técnico
3.6 Análisis de rendimiento	2	Consultor y técnico
3.7 Entrega PAC3, BD finalizada	0	
4 Entrega final	25	
4.1 Creación de los informes con la suite Pentaho	5	Consultor y técnico
4.2 Elaboración de la memoria final	10	Jefe de Proyecto
4.2 Elaboración presentación virtual	4	Jefe de Proyecto
4.3 Revisión final y corrección de errores	5	Jefe de Proyecto
4.4 Entrega del proyecto	0	Jefe de Proyecto

Resumiendo, este es el total de horas y de coste para cada perfil. Aunque las tareas las realiza una única persona, hemos contemplado las horas que realizan dos perfiles distintos.

PERFIL	JORNADAS	COSTE
Jefe de proyecto	26	7280 €
Consultor	25	5250 €
Técnico	58	8410 €
Total		**20.940 €**

6.2 Coste de la infraestructura

Después de valorar el coste de las horas del equipo del proyecto, vamos a valorar el *hardware* y las licencias necesarias para su implantación en el cliente.

La gestión de todo sistema sanitario requerirá de una infraestructura potente. Además, siendo un sistema crítico del que dependerá la gestión de todos los centros hospitalarios y farmacias del estado, se recomienda preparar un entorno de alta disponibilidad ante fallos. Hay varias formas de montar un entorno similar, en este caso hemos elegido la opción de la virtualización. Utilizaremos la solución VSphere de la empresa VMware.

Para ello necesitaremos los siguientes componentes:

1. Dos servidores con la siguiente configuración: 64 GB de *RAM*, dos procesadores *Xeon* a 3 GHz, dos discos duros en *RAID* y dos tarjetas *HBA* de fibra.

2. Dos *switches SAN* de fibra.

3. Cabina de discos de fibra con 5 discos de al menos 450 GB.

La estructura del sistema la podemos ver en la figura 17. Conectaremos las dos tarjetas de cada servidor a cada uno de los *switches* de fibra, y después los *switches* de fibra a la cabina. Dependiendo de la cabina, podemos conectar los *switches* a ésta a través de varios cables y configurar varias zonas, esto aumentará el rendimiento y la disponibilidad del entorno.

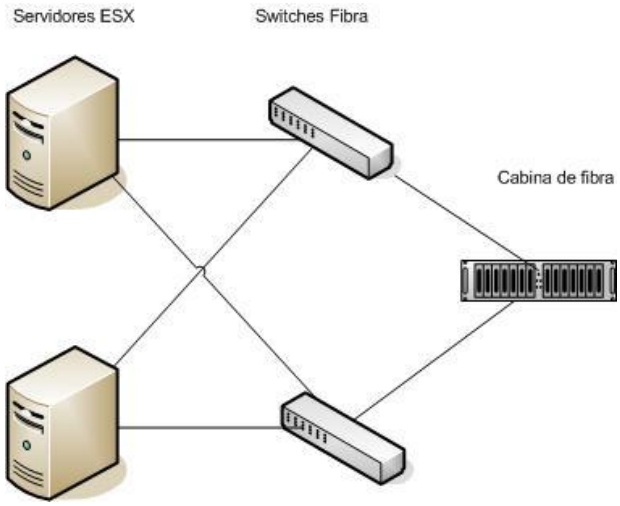

Figura 17. Esquema entorno virtualizado.

En cada servidor instalaremos el sistema ESX de VMware, para ello necesitaremos una licencia. Optamos por una licencia VSphere que incluye tres servidores ESX y el entorno de gestión VCenter. De esta forma podremos configurar los servicios vMotion y FaultTolerance. Estos servicios permiten que una máquina virtual que se está ejecutando en servidor, pase a ejecutarse en otro ante un fallo de *hardware* sin que los usuarios noten un corte en el servicio. El software de administración VCenter se podrá instalar en cualquier equipo de la red o en una máquina virtual, por lo que no será necesario adquirir otro equipo.

Crearemos una máquina virtual en la que instalaremos Oracle. Para crear la máquina virtual podemos optar por un sistema operativo de Microsoft o Unix. En este caso nos decantamos por un Windows Server 2008.

Por último, necesitamos una licencia de Oracle. Elegimos una licencia Oracle Database 11*g* Enterprise Edition, teniendo en cuenta el tamaño que alcanzará la base de datos y el número de sesiones concurrentes que podemos llegar a tener.

Sobre la suite Pentaho, empezaremos a trabajar con una licencia gratuita, que es con la que hemos creado los análisis; y en caso de necesitar en un futuro mayor funcionalidad, podremos adquirir una de pago. En cualquier caso el servidor Pentaho se puede instalar en una máquina virtual con una distribución Linux de Ubuntu, por lo que no tendría coste.

En este entorno no incluimos un sistema de copia de seguridad en cinta, optamos por hacer las copias en disco y copiarlas a través de la red a una ubicación remota de forma periódica.

Resumiendo, este es el total de coste de la infraestructura que necesitamos:

CONCEPTO	COSTE
2 Servidores	10.000 €
Switches de fibra y cableado	6.800 €
Cabina de discos	9.000 €
Licencia Vsphere	3.300 €
Licencia Windows Server 2008 Enterprise	3.500 €
Licencia Oracle 11*g* Enterprise Edition, licencia por procesador	57.000 €
Servicios de instalación y configuración	2.000 €
Total	**91.600 €**

6.3 Coste total del proyecto

Como hemos visto en los apartados anteriores, el coste total del proyecto asciende a 112.540 euros, e incluyes las partidas siguientes:

Coordinación del proyecto	7.280 €
Análisis	5.250 €
Desarrollo de la solución	8.410 €
Infraestructura	91.600 €

7. Conclusiones

La ejecución de este proyecto me ha supuesto un gran esfuerzo, no tanto por la complejidad del proyecto en sí, sino por la necesidad de dedicarle una gran cantidad de tiempo. El hecho de compaginar mis estudios con un trabajo en el que a menudo parece que no hay horarios, ha sido lo más complicado.

He intentado hacer un proyecto lo más profesional posible, sin llegar a un detalle imposible de acometer por una sola persona, en las horas que le he dedicado durante estos meses. En la memoria también he querido reflejar ese trabajo y aunque, si bien he razonado todas las decisiones tomadas, es posible que me haya faltado el citar la teoría en la que me basaba para realizar dichas afirmaciones. Quizás el hecho de trabajar diariamente con bases de datos y almacenes de datos, haya hecho que dé muchas cosas por sentado. Un reflejo de esto es la falta de bibliografía que he indicado en el apartado correspondiente, ya que para realizar el proyecto me ha bastado refrescar la teoría de las asignaturas Sistemas de Gestión de Bases de Datos y Modelos Multidimensionales de Datos; y realizar pequeñas consultas a la documentación de Oracle sobre el uso de algunas funciones.

El hecho de poder elegir el área de Bases de Datos para realizar el proyecto, ha sido toda una ayuda para mí. Es un área de la informática que realmente me apasiona.

Por último, comentar de nuevo que no ha entrado en el alcance de este proyecto la instalación y configuración del SGBD Oracle, ni de la herramienta ROLAP Pentaho.

8. Glosario

Almacén de datos. Colección de datos orientados al tema, integrados, no volátiles e históricos, organizados para dar soporte a procesos de ayuda a la toma de decisiones.

Base de datos operacional. Base de datos destinada a gestionar el día a día de una organización, almacena la información referente a la operativa diaria.

Celda. Punto unitario de datos que aparece en la intersección definida cuando se selecciona un valor para cada una de las dimensiones en un vector multidimensional.

Data Warehouse (DWH). Ver almacén de datos.

Dimensión. Punto de vista utilizado para el análisis de un hecho.

ETL (Extract Transform and Load). Proceso que permite extraer los datos de una base de datos operacional, transformarlos y cargarlos en un almacén de datos.

Hecho. Objeto de análisis del negocio de una organización.

Jerarquía. Conjunto de relaciones entre las instancias de una dimensión que indica cómo se agrupan los datos.

Medida. Dato numérico asociado a un hecho que queremos analizar.

OLAP (on-line analytical processing). Herramientas de análisis de datos multidimensionales.

ROLAP (relational OLAP). Es una de las implementaciones que existen en el mercado de herramientas OLAP. Se distingue por trabajar sobre un SGBD relacional.

SGBD (Sistemas de gestión de bases de datos). Software que gestiona y controla las bases de datos. Sus funciones principales son las de facilitar el uso simultáneo, independizar el usuario del mundo físico y mantener la integridad de los datos.

9. Bibliografía

1. Ángels Rius Gavídia, Montse Serra Vizern, Josep Curto Díaz. *Introducció a l'emmagatzematge de dades.* Barcelona: UOC.

2. Alberto Abelló Gamazo, José Samos Jiménez, Josep Curto Díaz. *La factoria d'informació corporativa.* Barcelona: UOC.

3. José Samos Jiménez, Josep Curto Díaz. *Construcció de la factoria d'informació corporativa.* Barcelona: UOC.

4. Alberto Abelló Gamazo. *Disseny Multidimensional.* Barcelona: UOC.

5. M. Elena Rodríguez González, Jaume Sistac i Planas. *Diseño conceptual y lógico de bases de datos.* Barcelona: UOC.

6. Alberto Abelló Gamazo, Jaume Sistac i Planas. *Reconsideración de los modelos conceptual y lógico.* Barcelona: UOC.

7. Blai Cabré i Segarra. *Diseño físico de bases de datos.* Barcelona: UOC.

8. *Documentación de Oracle.* <http://www.oracle.com/technetwork/es/documentation/index.html>