Universidad de las Ciencias Informáticas
Facultad 7

Título: Sistema de almacenamiento de información del módulo de cuerpo de guardia

Trabajo de Diploma para optar por el título de

Ingeniero en ciencias Informáticas

Autora: Leydanis Cantillo Quintana

Tutor: Ing. Yubismel Perdomo Velázquez

Asesora: Yudary Rojas Molina

Ciudad de La Habana, Junio 2007

DECLARACIÓN DE AUTORÍA

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los 30 días del mes de Junio del año 2007.

Leydanis Cantillo Quintana

_____

Firma de la Autora

Ing. Yubismel Perdomo Velázquez

_____

Firma del Tutor

DATOS DE CONTACTO

Nombre y apellidos del tutor: Yubismel Perdomo Velázquez
Email: yubismel@uci.cu
Profesor graduado de Ing. Informático en el año 2006. Ha impartido asignaturas como Introducción a la programación, Programación II y Sistemas Gestores de Bases de Datos. Posee categoría docente de adiestrado y actualmente cursa el diplomado de Docencia Universitaria.

Nombre y apellidos de la asesora: Yudary Rojas Molina
Email: yudany@uci.cu
Licenciada en Letras (Filología) por la Universidad Central de las Villas, profesora instructora recién graduada.

AGRADECIMIENTOS

A mi mamá, por ser una excelente guía en mi vida,

por su fe en mí, amor, cariño, y toda su constante dedicación.

A mi papá, por su apoyo durante todos estos años.

A mis tías Annis, Leydis, Nena y Loris por su apoyo durante estos cinco años.

A mi tío Iván, hermanos y demás familiares, por su ayuda en lo que estuvo a su alcance.

A mis amigas Lily, Yuya, Kirenia, Keila y Yenny (bombón), por aguantarme, tener paciencia y por pasar conmigo, algunos de los mejores momentos de mi vida.

A mis compañeros de grupo, por ser los mejores.

A Sule y Yenny, por ayudarme con mi tesis.

A mi tutor, por su apoyo, gracias por dedicarme su tiempo.

A mi asesora, por ser tan preocupada y ayudarme con el documento de tesis.

A todos los profesores que de una forma u otra han contribuido a mi formación.

Muchas gracias

DEDICATORIA

Dedico mi trabajo de diploma a mi madre y a mi tía Annis,
que siempre estuvieron presentes y me dieron fuerzas en los momentos más difíciles.
A mi padre, tías y demás familiares por la ayuda.
A mis abuelos Regina, Silvia, Congo y Chentón por su amor y su ejemplo.
Siempre los llevo en mi pensamiento y mi corazón.
A todos , sientan este trabajo como suyo.

RESUMEN

En la actualidad, los servicios que se proporcionan en los hospitales cubanos no están automatizados, lo que provoca deficiencias en la gestión de la información y en la atención a la población. El presente trabajo tiene como objetivo, realizar el diseño e implementación una base de datos que organice y almacene de forma segura la información procesada en el módulo de cuerpo de guardia.

Para el diseño de la base de datos se utilizó el Case Studio 2.22, herramienta que posibilita diseñar problemas que existen en la realidad y el Posgresql 8.2 como gestor de base de datos, permitió ratificar o reutilizar datos del área donde se aplica el sistema. Se tuvieron en cuenta diferentes factores para la creación de la solución propuesta, los requisitos funcionales y no funcionales, su integración con otros módulos o sistemas y la arquitectura de la base de datos empleada para su buen funcionamiento. Se tuvieron en cuenta aspectos fundamentales, como la integridad y seguridad de los datos.

Con el diseño de base de datos propuesto se logrará una eficaz implementación del sistema, organizando y asegurando la información relacionada con los procesos que se desarrollan en el cuerpo de guardia. Permitirá eliminar la redundancia de información, además de normalizar la base de datos a un nivel factible para su desarrollador. También será posible determinar la trazabilidad de las acciones, lo que permite establecer y controlar las acciones que los usuarios realizan al acceder a la misma.

ÍNDICE

ÍNDICE DE TABLAS

INTRODUCCIÓN

El contexto actual del sistema sanitario cubano demanda un manejo equitativo y efectivo. En el país, la sociedad le brinda gran importancia a esta esfera, es por eso que los descubrimientos tecnológicos, la mayor expectativa de vida, el nivel de información de la sociedad en su conjunto y por lo tanto la mayor exigencia justifican la realización de sistemas con alto grado de eficiencia que solucionen la situación existente (ASSETTA 2004).

El desarrollo de la informática ha beneficiado el sector sanitario, creando herramientas para hacerlo cada vez más productivo. En el área de la salud existen actualmente muchas actividades que necesitan ser automatizadas, es por ello que la aplicación de las nuevas tecnologías al proceso de automatización de los diferentes servicios que se desenvuelven dentro de este sector traería consigo un rotundo éxito en la atención a la población.

En Cuba, la revolución otorga gran importancia a dicho sector, por lo que el desarrollo del mismo es muy notable. Actualmente, existen grandes problemas de gestión de los servicios que se prestan en las instituciones hospitalarias: hospitales, policlínicos, consultorio de la familia, etc. Estos problemas vienen arraigados por la relación que existe entre los diferentes servicios que se prestan, los cuales no están automatizados y la mayoría se realizan de forma manual, archivando toda la información en papel. Esto puede provocar deficiencias a la hora de atender a la población, haciendo que el trabajo de los médicos y trabajadores de la salud sea más difícil.

Debido a los problemas existentes, se ha estudiado la posibilidad de informatizar todos los servicios hospitalarios. Aunque se han hecho intentos, nunca se ha llegado a un producto que solucione su situación.

A continuación se tratan las cuestiones principales del trabajo en los hospitales. Dentro de este amplio marco se centrará la atención en el flujo de trabajo del módulo de cuerpo de guardia.

Cuando un paciente asiste al cuerpo de guardia, no existe una persona encargada de organizar la lista de espera de los pacientes de acuerdo a la gravedad de los síntomas con que se presentan, creándose descoordinaciones en el funcionamiento del cuerpo de guardia.

Los datos de los pacientes que son atendidos en los cuerpos de guardia se registran en formularios existentes en papel, la hoja de cargo médica y posteriormente al terminar un turno de guardia se archivan en estadística. De esta forma, pueden ocurrir pérdidas de datos, lo que impide la rápida búsqueda de determinada información.

Se puede dar el caso de que un paciente asista al cuerpo de guardia varias veces al día, lo que traería consigo la redundancia de información, pues no hay una constancia de la última consulta realizada al paciente y se tendría que llenar los formularios con los mismos datos cada ves que este se presente. Esto dificulta el seguimiento en relación a alguna patología que presente el paciente.

También, ocurre que si el médico no tiene la historia clínica a mano durante la consulta, no puede conocer el historial clínico del paciente y depende de la información que este le brinde, en muchos casos no ofrece toda la necesaria y que verdaderamente interesa para la realización de la consulta y la emisión de un buen diagnóstico.

Es por ello, que existe la gran necesidad de desarrollar un sistema de gestión que automatice todas las actividades que se realizan dentro del cuerpo de guardia, y que para su funcionamiento exista una base de datos que contenga toda la información necesaria para su seguridad y buen funcionamiento.

Así, se puede plantear que **el problema** a resolver es ¿Cómo erradicar los problemas en el módulo de cuerpo de guardia en los hospitales mediante la automatización de un sistema de gestión?

Teniendo en cuenta lo anterior y como solución al problema planteado, se define como
Objetivo General: desarrollar una base de datos para asegurar la información y facilitar el trabajo en la automatización de los procesos que se desarrollan en el módulo de cuerpo de guardia.

Objeto de Estudio:
Procesos de la automatización de los servicios en los hospitales.

Campo de acción:
Procesos de automatización del módulo de cuerpo de guardia en los hospitales cubanos.

Para cumplir el objetivo general se desarrollaron las siguientes **tareas de investigación**:

- Estudiar los procesos vinculados con el módulo de cuerpo de guardia en los hospitales cubanos.
- Realizar el levantamiento de todos lo requisitos necesarios para la creación de la solución propuesta.
- Estudiar las herramientas a utilizar para el desarrollo de la base de datos.
- Realizar las consultas necesarias para obtener los resultados deseados.

El presente documento se estructura en: resumen, introducción, tres capítulos de contenidos, conclusiones, recomendaciones, bibliografía y anexos donde se incluye todo lo relacionado con el trabajo realizado. A continuación, se describe el contenido de los capítulos.

El Capítulo 1 Fundamentación teórica.

En este capitulo se hace referencia a los principales conceptos relacionados con la solución propuesta. Además se citan algunos ejemplos de Sistemas de Informacion Hospitalaria (SIH) que existen en la actualidad y se hace alusión a las herramientas utilizadas para el desarrollo de la solución propuesta**.**

El Capítulo 2: Descripción y análisis de la solución propuesta

En este capitulo se explicará la estrategia de integración de la solución con otros módulos o sistemas, se describirá la arquitectura de la base de datos, se argumentarán los requisitos funcionales y no funcionales del sistema propuesto, se plasmará el diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño realizado por el analista, además el diseño de la base de datos del módulo de cuerpo de guardia con la respectiva descripción de sus tablas.

El Capítulo 3: Validación teórica del diseño realizado.

En este capitulo se tendrán en cuenta los aspectos fundamentales para el buen funcionamiento de la base de datos como son la integridad de los datos, la redundancia de información, la normalización y seguridad de la base de datos así como la trazabilidad de las acciones para controlar todas las actividades que se realicen en la misma.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se hará un estudio detallado del estado del arte del tema de las base de datos dando a conocer de forma breve la historia de los gestores de base de datos y de los SIH. También se hace mención a algunos de estos sistemas que existen con sus respectivos gestores de base de datos, además de las herramientas utilizadas para lograr la solución propuesta.

1.1 Base de datos

En la actualidad, las bases de datos constituyen uno de los recursos más importantes en el desarrollo de cualquier esfera. Se ha demostrado que son de gran influencia el avance de los países a nivel mundial, admitiendo el almacenamiento y acceso confiable, eficiente y práctico de la información que se produce.

Una base de datos es un conjunto de información almacenada en memoria auxiliar que permite acceso directo y manipulan esos datos un conjunto de programas. Surgen desde mediados de los años sesenta y en años posteriores se propuso el modelo relacional, uno de los más utilizado actualmente, que ha marcado la línea de investigación por muchos años.

En consecuencia con esto, surgieron los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

1.2 Sistemas de gestión de base de datos

Los sistemas de gestión de base de datos son un tipo de software específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de

consulta. El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de información (WIKIPEDIA 2007b).

1.2.1 Breve historia de los sistemas de gestión de base de datos[1]

El uso de sistemas de bases de datos automatizados, se desarrolló a partir de la necesidad de almacenar grandes cantidades de datos, para su posterior consulta, producidas por las nuevas industrias que creaban gran cantidad de información.

En la década del 50, surgen las cintas magnéticas, sirvieron para suplir las necesidades de información de las nuevas industrias. Por medio de este mecanismo, se empezó a automatizar la información de las nóminas, como por ejemplo el aumento de salario. Consistía en leer una cinta o más y pasar los datos a otra. La nueva cinta a la que se transfiere la información pasa a ser una cinta maestra. Estas solo se podían leer secuencial y ordenadamente.

Posteriormente se crearon los discos, que fueron un adelanto muy efectivo, ya que por medio de estos se podía acceder a la información directamente, esto ayudó a ahorrar tiempo. Los discos dieron inicio a las bases de datos de red y jerárquicas, pues era posible guardar estructuras de datos como listas y árboles.

En años posteriores Edgar Frank Codd, definió el modelo relacional, naciendo así, las bases de datos relacionales. Inicialmente no se usó este tipo de modelo debido a que tenía inconvenientes por el rendimiento, ya que no podían ser competitivas con las bases de datos jerárquicas y de red.

[1] Para realizar este epígrafe se consultó: Historia de las bases de datos en Ciencia de la Información, en http://recursostic.javeriana.edu.co/wiki/index.php/Historia_de_las_bases_de_datos_en_Ciencia_de_la_Informaci%C3%B3n#Historia__de_las_bases_de_datos_en_Ciencia_de_la_Informaci.C3.B3n, junio 2007.

Ya en los años 80 las bases de datos relacionales, con su sistema de tablas, filas y columnas, pudieron competir con las bases de datos jerárquicas y de red, ya que su nivel de programación era bajo y su uso muy sencillo. En esta década, se iniciaron grandes investigaciones paralelas y distribuidas, como las bases de datos orientadas a objetos.

Más adelante se crea el lenguaje estructurado SQL de consulta que analiza grandes cantidades de información permitiendo la especificación de diversos tipos de operaciones frente a la misma información.

En la actualidad, existe gran cantidad de alternativas en línea que permiten hacer búsquedas orientadas a necesidades específicas de los usuarios. Una de las tendencias más amplias son las bases de datos que cumplan con el protocolo Open Archives Initiative – Protocol for Metadata Harvesting (OAI-PMH) permitiendo el almacenamiento de gran cantidad de artículos, lo que permite una mayor visibilidad y acceso en el ámbito científico y general.

1.3 Sistema de información hospitalaria

Los sistemas han existido desde hace mucho tiempo, aunque no eran automatizados. Después del surgimiento de las computadoras, empezaron a crearse sistemas sencillos de información, con diferentes fines. En la década de los setenta en el sector de la salud surgen los primeros sistemas de información médica que dieron lugar a los SIH. Su impacto en estas instituciones es fuerte, ya que busca elevar la calidad de la atención del paciente, de los servicios brindados y la productividad (CERRITOS 2003).

Los SIH son sistemas que realizan la recolección, almacenamiento, procesamiento, recuperación y comunicación de información de atención al paciente para todas las actividades relacionadas con el Hospital.

La meta es construir un SIH estandarizado donde el paciente sea el más beneficiado, y los profesionales de la salud encuentren un recurso idóneo, amigable y flexible que responda a las necesidades de información de la institución hospitalaria o de salud.

1.3.1 Historia de los sistemas de información hospitalaria.

Las tecnologías de la información llegan a los hospitales a mediados de la década de los ochenta, utilizando computadoras personales y redes de área loca. En esta época la informatización apuntaba solamente a la gestión administrativa y su uso inicial tendía a facilitar pagos y a automatizar el reporte de resultados (ASSETTA 2004).

Posteriormente su uso se extiende al área clínica con el objetivo de mejorar la calidad de la atención a la población. Actualmente, un SIH debería incluir: registro del paciente con expediente digital, generación de recetas y otras órdenes o formularios de trabajo, control de laboratorios y quirófanos desde una sala de mando, generación de estadísticas y reportes, videoconferencias, acceso a bibliotecas y base de datos, telemedicina, etc.

1.3.2 Ejemplo de algunos de los sistemas de información hospitalaria

Actualmente se están haciendo grandes esfuerzos por desarrollar aplicaciones con el objetivo de hacer el trabajo más fácil y productivo para los médicos y trabajadores de la salud.

Existen varios SIH, uno de los más conocidos es el Galen que está diseñado para ser utilizado por todo el personal perteneciente a un centro hospitalario que necesitan del sistema para optimizar su trabajo y elevar su eficiencia. Está desarrollado para una plataforma Windows de 32 bits con una configuración cliente/servidor y usa como gestor de base de datos SQLServer 2000.

También está el Care2x desarrollado desde el 2002, basado en estándares OpenSource. El sistema posee varios módulos: admisión, turnos, historia clínica básica, administración de recursos humanos, quirófanos, farmacia, laboratorio, internación, entre otros. Este sistema integra datos, funciones y flujo de tareas en un entorno de cuidados de la salud. Está basado en php+MySQL+Apache web server, escalable, modular y que soporta plugins (LATORILLA 2004).

A pesar de las grandes ventajas que posee el Care2x, tiene funcionalidades que el sistema hospitalario cubano no realiza y su precio es muy alto.

También está, el GalenHos ha sido diseñado con el propósito de apoyar a los establecimientos de salud en el correcto registro de información, clínica o administrativa, y la generación de información gerencial. Posee como gestor de base de datos el SQL Server 2000 (CASTRO 2005).

Otro de los sistemas es el Medfile que es un sistema diseñado para satisfacer las necesidades de un consultorio médico en general, y en particular para archivar historias clínicas de pacientes en una base de datos, asignar turnos para la consulta y emitir prescripciones y órdenes médicas en forma altamente personalizable y configurable por el usuario.

Este sistema utiliza SQL Server 2000 como gestor de base de datos que es un software propietario cuya licencia posee altos costos.

El sistema informativo hospitalario Medisys en su versión cliente/servidor**,** permite registrar, controlar y procesar la información necesaria del paciente para la realización de los servicios de salud que requiera, así como la información de la planificación, ejecución y supervisión de los servicios de salud prestados por la institución para facilitar la toma de decisiones con vistas a mejorar la calidad y eficiencia de los mismos (COPYRIGHT, CEDISAP 1998).

El sistema consta de un grupo de módulos que tienen una misión bien definida en el universo de actividades del hospital y que se interrelacionan para dar una respuesta integral a las necesidades informativas de la institución. Este sistema utiliza como gestor de base de datos para plataforma cliente/servidor el SQL Server 6, 5.

Algunos de los módulos que este sistema posee son:

1. Urgencias
2. Laboratorio clínico
3. Salón de Operaciones, etc.

Otro de los sistemas es el Estadillo de Urgencias, que es el sistema de control de pacientes que se implantará en el servicio de urgencias. Proporciona la misma información en cualquier punto del servicio, manteniendo actualizado en todo momento el estado de los pacientes actualmente en urgencias, dando información de las pruebas solicitadas y el estado en el que se encuentran(*Estadillo de Urgencias.*).

1.4 Herramientas utilizadas para el desarrollo de la solución propuesta.

Las tecnologías están presentes en cualquier actividad o tarea que se proponga realizar, es por ello que para el desarrollo de la base de datos del módulo de cuerpo de guardia, se realizó un estudio exhaustivo de las herramientas a utilizar, tanto para el diseño como para la confección en general de la misma.

1.4.1 Propuesta de los principales gestores de base de datos para el desarrollo de la solución propuesta

Se realizó el estudio de cuatro gestores de base de datos entre los múltiples que existen pues son los más utilizados para desarrollar una base de datos en la actualidad:

Microsoft SQL Server 2000 es un gran gestor de bases de datos, permite almacenar y administrar grandes bases de datos y es muy organizado. Se considera que es el más completo, aunque consume muchos recursos y posee un gran soporte para administrar bases de datos distribuidas (GMBH. 2007).

Ventajas:

1. Soporta la configuración automática y la auto-optimización.
2. Administración multiservidor para un gran número de servidores.

Desventajas:

1. Licencias con costos altos
2. Plataformas Windows

MySql

Es una idea originaria de la empresa open source MySQL AB. Esta lo desarrolla como software libre en un esquema de licenciamiento dual. Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso.

La GNU GPL (*General Public License* o licencia pública general) es una licencia creada por la Free Software Foundation a mediados de los 80. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Posee algunas ventajas:

1. Diseñado en vistas a la velocidad.
2. Consume muy pocos recursos de CPU y memoria
3. Muy buen rendimiento.
4. Tamaño del registro sin límite.
5. Buena integración con PHP.
6. Utilidades de administración (phpMyAdmin).
7. Buen control de acceso usuarios-tablas-permisos.

Desventajas:

1. No soporta subconsultas.
2. No soporta transacciones.

Oracle

Es un sistema de gestión de base de datos fabricado por Oracle Corporation. Se considera como uno de los sistemas de bases de datos más completos, destacando su:

1. -Soporte de transacciones.
2. -Estabilidad.
3. -Escalabilidad.
4. -Es multiplataforma.

Su mayor defecto es su enorme precio.

Gestor utilizado en el desarrollo de la solución propuesta.

Para desarrollar la base de datos del proyecto se seleccionó PostgreSQL 8.2 porque es un motor de base de datos, liberado bajo la licencia Berkeley Software Distribution (BSD), licencia otorgada

a los a los sistemas BSD y pertenece al grupo de licencia de software libre. Una vez obtenido este tipo de software puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente, pero no hay que asociar software libre a software gratuito, pues aunque conserve su carácter de libre, puede ser vendido comercialmente (GROUP 2007).

PostgreSQL ofrece muchas ventajas para el proyecto respecto a otros sistemas de bases de datos:

1. Instalación ilimitada: con PostgreSQL nadie puede demandarlo por violar acuerdos de licencia del software.
2. Confiabilidad: se puede trabajar con total confianza, sin temor a que sufra algún daño o caída, puesto que esto nunca ha sucedido en largos años de alta actividad.
3. Software Libre: se puede redistribuir libremente sin costo alguno.
4. Alta concurrencia: PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

Se utiliza específicamente la versión 8.2 pues en su paquete de instalación tiene la herramienta de implicación Slony que permite ratificar datos del área local donde se aplique el sistema.

Las características avanzadas que se ofrecen con PostgreSQL 8.2 antes que ningún otro sistema de gestor de bases de datos incluyen:

1. Los índices invertidos generalizados son una forma más escalable y programable de indexar datos semi-estructurados y texto (CUSCO 2004-2007).

2. PostgreSQL ha sido instrumentado para permitir la trazabilidad a través de un marco genérico de monitoreo, usando DTrace en Solaris y otras herramientas avanzadas de traza (CUSCO 2004-2007).

1.4.2 Principales herramientas de diseño de base de datos.

En la actualidad existen gran variedad de herramientas que permiten el diseño de una base de datos entre ellos están el Toad data modeler, el ERStudio y el CaseStudio

Toad Data Modeler.

Toad Data Modeler es una aplicación que no sólo permite diseñar esquemas de base de datos, sino también generar el código SQL necesario para producirlas.

Con el se puede desarrollar diagramas para la mayor parte de sistemas gestores de bases de datos existentes: Access, Firebird, InterBase, MySQL, Oracle, Paradox, Postgre, Sybase y muchos más.

La aplicación resulta muy útil a la hora de crear diagramas de entidad-relación, definir reglas de integridad referencial, generar scripts SQL que construyan la base de datos.

Además, posee una herramienta denominada 'Model Explorer' que permite navegar por todos los atributos del modelo que se cree. Esta herramienta posee una gran desventaja ya que con ella no se pueden diseñar grandes base de datos, ya que no soporta un diseño que posea más de 24 tablas.

ER/Studio.

Es una herramienta de modelado de datos, fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen las aplicaciones de bases de datos, grandes y complejas.

ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los

diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes.

ER/Studio ofrece las siguientes funcionalidades:

1. Capacidad fuerte en el diseño lógico.
2. Sincronización bidireccional de los diseños lógico y físico.
3. Construcción automática de base de datos.
4. Reingeniería inversa de base de datos.
5. Documentación basada en HTML.
6. Un repositorio para el modelado

Herramienta utilizada para el diseño de la base de datos

Para el desarrollo de la base de datos, se consideró que la herramienta más eficiente es el CaseStudio.

Case Studio en su versión 2.22

Para el desarrollo del modelado de la base de datos del módulo de cuerpo de guardia se escogió el Case Studio 2.22. Es una herramienta profesional con la que se puede diseñar bases de datos propias, facilitando herramientas para la creación de diagramas de relación, modelado de datos y gestión de estructuras.

Tiene soporte para trabajar con una amplia variedad de formatos de base de datos (Oracle, SQL, MySQL, PostgreSQL, Access, etc.) y permite además generar scripts SQL, aplicar procesos de retroingeniería a las bases de datos, usar plantillas de diseño personalizables y crear detallados informes en HTML y RTF. Su principal características, es su potente sistema de ingeniería

inversa, que permite identificar y estructurar bases de datos ya existentes para poder trabajar con ellas sin problemas.

Con el estudio realizado en este capítulo, se llegó a la conclusión de que existen varios SIH a nivel mundial pero ninguno soluciona todos los problemas que coexisten en los diferentes servicios que se prestan en un hospital.

También se concluyó, que las herramientas mas óptimas para el desarrollo de la base de datos fueron el CASE Studio 2.22 y el PostgreSql 8.2, por la compatibilidad que existe entre ellos y las ventajas que proporciona, que ningún otro software brinda.

CAPÍTULO 2. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

En este capítulo se precisarán los requisitos funcionales y no funcionales tenidos en cuenta para desarrollar la base de datos propuesta. Se especificará la arquitectura establecida, para el desarrollo de la misma. Así como, su estrategia de integración con otros módulos o sistemas. También se analizarán el diseño de la base de datos y como lograr una mejor optimización de consultas para obtener un resultado superior.

2.1 Estrategia de integración de la solución con otros módulos o sistemas

La integración de solución con otros módulos o sistemas es realizada a nivel del negocio donde todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa. La comunicación entre diferentes componentes se realiza mediante llamadas a servicios web o de forma directa a nivel de negocio, en caso de utilizarse servicios web, la información que es transmitida debe cumplir con los estándares internacionales que hay establecidos para facilitar la integración entre nuevos componentes y otros sistemas hospitalarios. La base de datos es accesada de forma directa mediante controladoras y los componentes rehusados son integrados mediante interfaces sencillas.

2.2 Tipos de arquitecturas de base de datos que existen

La arquitectura de un sistema de base de datos está influenciada en gran medida por el sistema informático en el que se ejecuta el sistema de base de datos. En la arquitectura del mismo se reflejan aspectos como la conexión en red, el paralelismo y la distribución.

Entre los diferentes tipos de arquitecturas de base de datos que existen se pueden mencionar los siguientes.

2.2.1 Arquitectura ANSI/SPARC[2]

El objetivo principal de la arquitectura ANSI/SPARC es definir un SGBD con el máximo grado de independencia, separando las aplicaciones de usuario y la base de datos física. Para ello, se utilizan tres niveles de abstracción conocidos como interno, conceptual y externo.

1. El nivel interno es el más cercano a la máquina. Es una representación a bajo nivel de la base de datos (BD) en la que se define la forma en la que los datos se almacenan físicamente en la máquina.

2. El nivel conceptual tiene un esquema conceptual, que describe la estructura de los datos que van a ser almacenados en la base de datos.

3. El nivel externo incluye varios esquemas externos. Cada esquema externo describe la parte de la base de datos en la que está interesado un grupo de usuarios en particular y esconde el resto de la base de datos para esos usuarios.

2.2.2 Características de un sistema cliente/servidor[3]

Un sistema cliente/servidor es aquel en el que uno o más clientes y uno o más servidores, conjuntamente con un sistema operativo y un sistema de comunicación entre procesos, forma un sistema compuesto que permite el análisis y presentación de los datos. Si existen múltiples servidores de base de datos, cada uno de ellos deberá procesar una base de datos distinta, para que el sistema sea apreciado como cliente/servidor. Cuando dos servidores procesan la misma

[2] y [3] Para realizar estos epígrafes se consultó de Óscar González Martín (1999/2000): ARQUITECTURAS DE SISTEMAS DE BASES DE DATOS, en http: //alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T9900_OGonzalez.pdf, junio 2007.

base de datos, el sistema ya no se llama un sistema cliente/servidor, sino que se trata de un sistema de base de datos distribuido.

Las principales características de la arquitectura cliente/servidor son:

1. El servidor presenta a todos sus clientes una interfaz única y bien definida.

2. El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.

3. El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.

4. Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Tipos de arquitecturas cliente/servidor

Arquitectura de 2 capas

La arquitectura cliente/servidor tradicional es una solución de 2 capas. Esta consta de tres componentes distribuidos en dos capas: cliente, que es el que solicita los servicios y servidor, que es el que provee los servicios. Los tres componentes son:

1. Interfaz de usuario.
2. Gestión del procesamiento.
3. Gestión de la base de datos.

Limitaciones que presenta

1. No hay independencia entre la interfaz de usuario y los tratamientos, lo que hace difícil la evolución de las aplicaciones.

2. Dificultades en la relocalización de las capas de tratamiento consumidoras de cálculo.

3. Reutilización delicada del programa desarrollado bajo esta arquitectura.

Arquitectura de 3 capas

La arquitectura de 3 capas surgió para superar las limitaciones de la arquitectura de 2 capas. La tercera capa está entre el interfaz de usuario y el gestor de datos. La capa intermedia proporciona gestión del procesamiento y en ella se ejecutan las reglas y lógica del mismo.

Permite cientos de usuarios y es usada cuando se necesita un diseño cliente/servidor que proporcione incrementar el rendimiento, flexibilidad, mantenibilidad, reusabilidad y escalabilidad mientras se esconde la complejidad del procesamiento distribuido al usuario.

Limitaciones que presenta

1. Construir una arquitectura de 3 capas es una tarea complicada. Las herramientas de programación que soportan el diseño de arquitecturas de 3 capas no proporcionan todos los servicios deseados que se necesitan para soportar un ambiente de computación distribuida.

2. Un problema potencial en el diseño de arquitecturas de 3 capas es que la separación de la interfaz gráfica de usuario, la lógica de gestión de procesamiento y la lógica de datos no es siempre obvia. Algunas lógicas de procesamiento de transacciones pueden aparecer en las 3 capas.

2.2.3 Arquitectura de base de datos distribuida

En un sistema de base de datos distribuida, los datos se almacenan en varios computadores. Los computadores de un sistema distribuido se comunican entre sí a través de diversos medios de comunicación, tales como cables de alta velocidad. No comparten la memoria principal ni el reloj.

Un sistema distribuido de bases de datos consiste en un conjunto de localidades, cada uno de las cuales puede participar en la ejecución de transacciones que accedan a datos de una o varias localidades. La diferencia principal entre los sistemas de base de datos centralizados y distribuidos es que, en los primeros, los datos residen en una sola localidad, mientras que, en los últimos, se encuentran en varias localidades (COPYRIGHT, ORANGE 2007).

Existen varias razones para construir sistemas distribuidos de bases de datos que incluyen compartir la información, fiabilidad, disponibilidad y agilizar el procesamiento de las consultas. Pero también tiene sus desventajas como desarrollos de software más costosos y mayor posibilidad de errores.

2.2.4 Arquitectura de la base de datos del módulo de cuerpo de guardia

La arquitectura del módulo de cuerpo de guardia es centralizada, pues el Sistema Gestor de Base de Datos (SGBD) esta implantado en una sola plataforma u ordenador desde donde se gestiona directamente, de modo centralizado, la totalidad de los recursos.

Los sistemas gestores de base de datos centralizados, comprenden el rango desde los sistemas de bases de datos monousuario ejecutándose en computadoras personales hasta los sistemas de bases de datos de alto rendimiento ejecutándose en grandes sistemas.

2.3 Análisis y optimización de consultas[4]

La reescritura de una consulta para que se ejecute más rápidamente es una labor que solo puede traer buenos efectos y el precio a pagar es solamente la inversión de tiempo del optimizador en reescribirla.

Antes de decidir si una consulta debe ser elaborada nuevamente para un mejor desempeño es necesario analizar si se está ejecutando muy lentamente.

Para ello existen dos indicadores:

1. La consulta accede demasiado al disco, por ejemplo, siendo una consulta puntual recorre toda la tabla.

2. Al revisar el plan de ejecución se observa que hay índices relevantes que no se utilizan.

Existen varias reglas que son de gran utilidad para mejorar el rendimiento de una consulta:

1. La cláusula distint es costosa de ejecutar porque generalmente involucra un ordenamiento de las tuplas resultantes para eliminar duplicados. Es necesario analizar si realmente hace falta usar esa cláusula.

2. En algunas consultas el usuario almacena resultados intermedios explícitamente en tablas temporales. Estas tablas pueden bajar el rendimiento de la consulta porque fuerza un orden de ejecución que quizás no sea ideal.

[4] Para realizar este epígrafe se consultó de Soraya Abad Mota (2005): Entonación de Bases de Datos, en http://www.bd.cesma.usb.ve/ci5851/apuntes3.pdf, junio 2007

3. Es preferible no usar la cláusula having si la condición deseada se puede expresar en la cláusula where.

4. Otra particularidad importante que es bueno conocer es si el orden de las tablas en la cláusula from puede afectar la implementación del join utilizado, posiblemente esto es relevante para join que involucran 5 tablas o más.

Optimizando sentencias INSERT

- Si se insertan muchas filas desde el mismo cliente al mismo tiempo, es mucho más rápido usar una sentencia INSERT con múltiples listas de valores para insertar varias filas a la vez que usar varias sentencias INSERT de manera separada. Por ejemplo, la siguiente consulta:

INSERT INTO nombreTabla **VALUES** (registro1), (registro2),... (registroN);

Es mucho más rápida que esta alternativa:

INSERT INTO nombreTabla **VALUES** (registro1);
INSERT INTO nombreTabla **VALUES** (registro2);
...
INSERT INTO nombreTabla **VALUES** (registroN);

En el caso de las consultas INSERT de la base de datos del módulo de cuerpo de guardia se realizaron las sentencias INSERT utilizando la herramienta Abstract Factory, que permite entrar un registro a la vez. Un ejemplo de ellas es en la tabla cfg_clasificacion:

INSERT INTO cfg_clasificacion
(

```
        id_codigo,
        codigo,
        prioridad
)
values
(
        pid_codigo,
        pcodigo,
        pprioridad
);
```

Optimizando sentencias UPDATE

Las sentencias UPDATE son optimizadas de manera similar a las sentencias SELECT con la sobrecarga adicional de la escritura. Por ejemplo:

UPDATE nombreCampo **FROM** nombreTabla **WHERE** algunaCondicion

Es el mismo que este:
SELECT nombreCampo **FROM** nombreTabla **WHERE** algunaCondicion

La base de datos del módulo de cuerpo de guardia, necesita de sentencias select y update para seleccionar y actualizar los datos. Estas se ejecutaron de forma óptima para lograr una mayor velocidad de respuesta.

A continuación se mencionaran varios ejemplos de sentencias select y update de la base de datos del módulo de cuerpo de guardia.

Sentencia SELECT

```
SELECT * FROM cfg_clasificacion
    WHERE
        (id_codigo = pid_codigo OR pid_codigo is null) and
        (codigo = pcodigo OR pcodigo is null) and
        (prioridad = pprioridad OR pprioridad is null)
```

Se seleccionan los campos de la tabla cfg_clasificacion dada la condición.

Sentencia UPDATE

```
    UPDATE cg_examen_fisico
    SET
        id_consulta = pid_consulta,
        fecha_hora = pfecha_hora,
        peso = ppeso,
        temperatura = ptemperatura,
        frecencia_cardiaca = pfrecencia_cardiaca,
        frecencia_respiratoria = pfrecencia_respiratoria,
        frecencia_alterial_minima = pfrecencia_alterial_minima
    WHERE
    (
        id_consulta = pid_consulta
    );
```

Se actualizan los datos de la tabla cg_examen_fisico dada la condición exigida.

2.4 Selección y argumentación de los requisitos funcionales y no funcionales del sistema propuesto

En los sistemas informáticos se necesita determinar los requisitos funcionales y no funcionales, ya que en gran medida de ellos depende el buen funcionamiento de los mismos.

2.4.1 Requisitos funcionales del módulo de cuerpo de guardia.

Los requisitos funcionales son todos aquellos que se refieren a las actividades para las cuales se ha desarrollado el sistema. En el módulo de cuerpo de guardia en específico se determinaron cinco requisitos funcionales.

RF1. Recepción del paciente.

Cuando un paciente llega a la institución hospitalaria es atendido por el recepcionista. Este le recoge los datos al paciente, busca si esta inscrito, en caso de que no lo este, el paciente debe dirigirse a inscripción y admisión para crearse su historia clínica.

Una vez creada esta, el recepcionista clasificará al paciente dependiendo de la gravedad de los síntomas que este presente. Luego lo envía a consulta y en caso de que se necesite algún examen físico se le solicitarán los servicios a enfermería.

RF2. Consultar paciente del cuerpo de guardia.

Una vez que el paciente llega a la consulta se actualiza su historia clínica agregándole la última consulta con todos los datos que esta conlleva.

Dependiendo de las dolencias del paciente puede que el médico lo remita a otra especialidad o solicite los servicios de laboratorio para la realización de los análisis pertinentes. De igual manera se pueden solicitar los servicios de enfermería.

RF3. Consultar paciente urgencia.
Cuando un paciente llega de urgencia al cuerpo de guardia, se le crea una mini historia clínica en la cual se recogen los datos de la consulta.

Dependiendo de la gravedad del paciente, este puede ser remitido a otra especialidad o a una sala de cuidados intensivos (UCIE).

RF4. Consultar paciente especializada.
Cuando un paciente es remitido a otra especialidad se le agrega la consulta con todos los datos pertinentes. Puede que el paciente necesite los servicios de laboratorio o enfermería, o sea remitidos a una sala de observación u otra especialidad.

El sistema da la opción de eliminar estas consultas, aunque estas se guardarán indefinidamente en la base de datos, para en caso de que se necesite esta información este siempre disponible.

RF5. Informes.
Al finalizar cada guardia, se confecciona un reporte con toda la información relacionada con los pacientes atendidos en el turno de guardia.

El sistema da la opción de eliminar estos reporte, aunque estos se guardarán indefinidamente en la base de datos, para en caso de que se necesite esta información este disponible.

2.4.2 Requisitos no funcionales del sistema propuesto.

Los requisitos no funcionales son aquellas características físicas que debe tener el sistema para su buen funcionamiento. En el caso de la base de datos del módulo de cuerpo de guardia se determinaron los siguientes.

R.N.F1 Requerimientos de rendimiento

El tiempo de respuesta de una petición al servidor debe ser rápido para la toma de decisiones.

R.N.F2 Requerimientos de soporte

Se le debe dar mantenimiento periódicamente a los servidores de bases de datos controlando la integridad de la información.

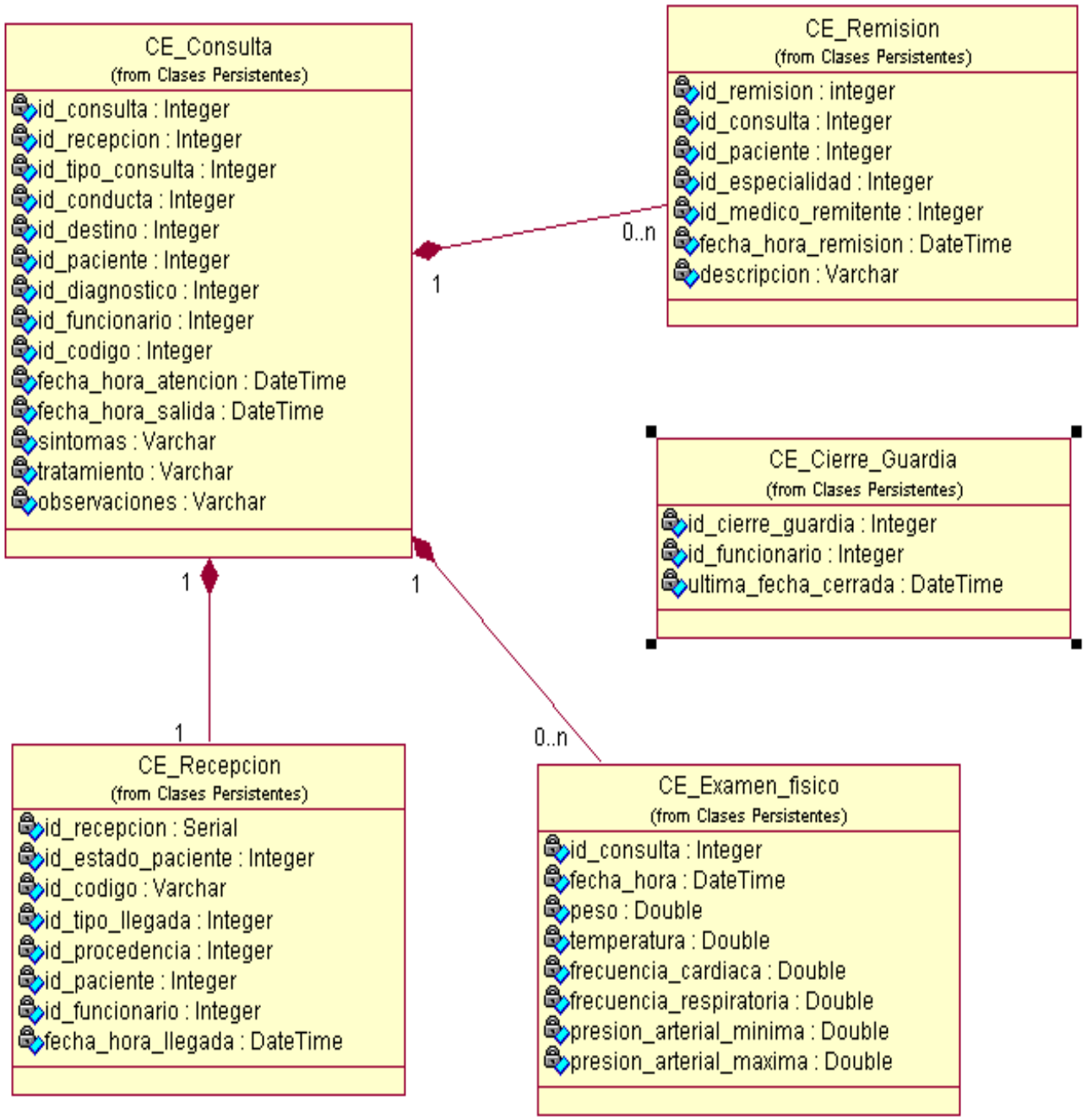
R.N.F3 Requerimientos de seguridad y privacidad

La información debe transmitirse de manera segura, se debe garantizar la seguridad a todos los niveles (Interfaz, negocio y Acceso a datos) restringiendo las funcionalidades mediante roles de usuarios garantizando que la información sea accesible al usuario autorizado.

R.N.F4 Requerimientos de confiabilidad

La información debe transmitir a través de canales seguros. Se debe chequear la integridad de los datos.

2.5 Modelo de clases persistentes del módulo de cuerpo de guardia



2.5.1 Descripción de las clases del modelo de clases persistentes

TABLA CE_Consulta

Nombre: CE_Consulta	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_consulta	integer
Id_recepcion	integer
Id_tipo_consulta_cg	integer
Id_conducta	integer
Id_destino	integer
Id_paciente	integer
Id_diagnostico	integer
Id_funcionario	integer
Id_codigo	integer
Fecha_Hora_atencion	Data time
Fecha_Hora_salida	Data time
sintomas	varchar
tratamiento	varchar
observaciones	varchar

TABLA CE_Remision

Nombre: CE_Remision	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_remision	integer
Id_consulta	integer
Id_paciente	integer
Id_especialidad	integer
Id_medico_remitente	integer
Fecha_hora_remision	Data time
descripcion	varchar

TABLA CE_Recepcion

Nombre: CE_Recepcion	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_recepcion	integer
Id_estado_paciente	integer
Id_codigo	integer
Id_tipo_llegada	integer
Id_procedencia	integer
Id_paciente	integer
Id_funcionario	integer

Fecha_hora_llegada	Data time

TABLA CE_Examen_fisico

Nombre: CE_Examen_fisico	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_consulta	integer
Fecha_hora	Data time
peso	Double precision
temperatura	Double precision
Frecuencia_cardiaca	Double precision
Frecuencia_respiratoria	Double precision
tension_alterial_minima	Double precision
tension_alterial_maxima	Double precision

TABLA CE_Cierre_guardia

Nombre: CE_Cierre_guardia	
Tipo de clase: entidad	
Atributo	**Tipo**
Id_cierre_guardia	integer
Id_funcionario	integer
Ultima_fecha_cerrada	Data time

2.6 Tipos de modelo de datos existentes

El modelado de la base de datos es un paso importante para su desarrollo, constituye el inicio de su creación. Como existen varios tipos de modelos de base de datos, se realizó un estudio para determinar cual sería el más óptimo.

2.6.1 Modelo de datos jerárquico

Se puede definir un modelo jerárquico como un conjunto de entidades que se relacionan entre sí proponiendo un conjunto de restricciones inherentes que provienen de la estructura jerárquica.
Un modelo de datos jerárquico almacena la información en una estructura jerárquica que enlaza los registros en forma de estructura de árbol en donde un nodo padre de información puede tener varios nodos hijo (WIKIPEDIA 2007a).

A diferencia del modelo relacional, el modelo jerárquico; no diferencia una vista lógica de una vista física de la base de datos. De manera que las relaciones entre los datos se establecen siempre a nivel físico, es decir, mediante referencia a direcciones físicas del medio de almacenamiento (WIKIPEDIA 2007a).

En el modelo jerárquico, el almacenamiento de los datos se realiza en forma de registros; el equivalente a las filas del modelo relacional. Cada uno consta de un conjunto de campos, el equivalente a las columnas del modelo relacional. Un conjunto de registros con los mismos campos se denomina fichero, el equivalente a las tablas del modelo relacional.

A continuación se mencionan los problemas típicos de las bases de datos jerárquicas y que no existen en las bases de datos relacionales:

1. Duplicidad de registros, pues no se garantiza que dos registros cualesquiera tengan valores diferentes en un subconjunto de campos.

2. Integridad referencial, donde no existe garantía de que un registro hijo esté relacionado con un registro padre válido. Por ejemplo, es posible borrar un nodo padre sin eliminar antes los nodos hijo, de manera que éstos últimos están relacionados con un registro inválido o inexistente.

3. Desnormalización, a diferencia del modelo relacional, las bases de datos jerárquicas no tienen controles que impidan la desnormalización de una base de datos.

2.6.2 Modelo de datos de red

El modelo de datos de red surge como respuesta a limitaciones del modelo jerárquico en cuanto a representación de relaciones más complejas.

El modelo de datos en red representa las entidades en forma de nodos de un grafo, y las interrelaciones entre estas mediante arcos que unen dichos nodos. En principio esta representación no impone restricción alguna acerca del tipo y el número de arcos que puede haber, con lo que se pueden modelar estructuras de datos tan complejas como sea necesario(GUZMÁN 2000-2001).

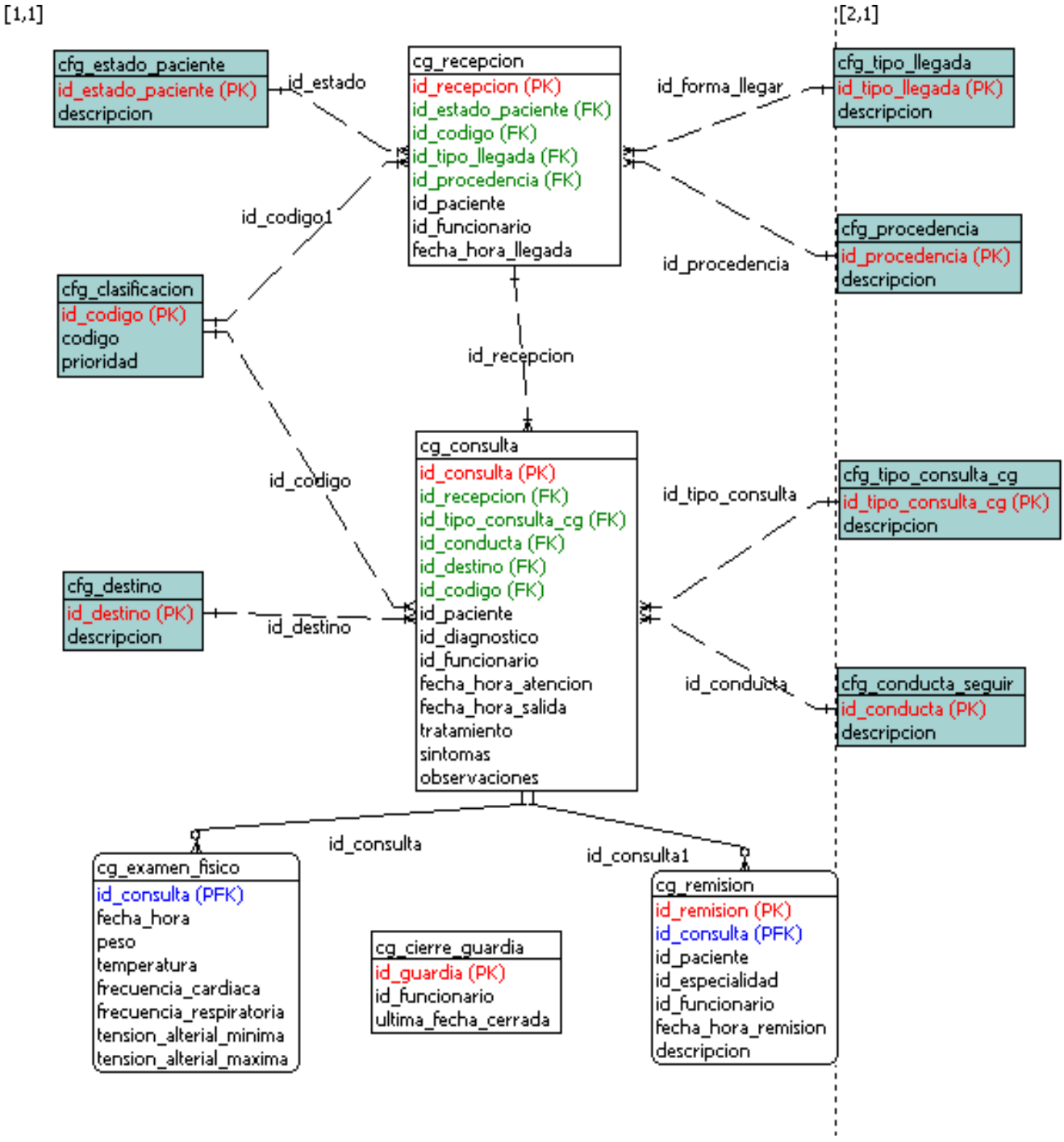
2.6.3 Modelo de datos relacional

El tipo de modelo utilizado para el diseño de la base de datos del módulo de cuerpo de guardia es el modelo de datos relacional.

Este es el más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de relaciones. Estas podrían considerarse en forma lógica como conjuntos de datos llamados tuplas.

Un modelo de datos relacional es un conjunto de dos o mas tablas estructuradas en registros y campos, que se vinculan entre sí por un campo en común, en ambos casos posee las mismas características como por ejemplo el nombre de campo, a este campo, generalmente, se le denomina ID, identificador o clave. En el caso de la base de datos del módulo de cuerpo de guardia, esto se puede observar ya que una tabla hereda la llave de otra dependiendo de la relación que exista entre ellas.

2.7 Diseño de la base de datos del módulo de cuerpo de guardia



2.7.1 Descripción de las tablas de la base de datos

TABLA cg_recepcion

Nombre: cg_recepcion		
Descripción: recoge los datos de los pacientes cuando llegan al hospital		
Atributo	**Tipo**	**Descripción**
Id_recepcion(PK)	serial	Identificador que se le asigna a la recepción del cuerpo de guardia.
Id_funcionario	integer	Identificador que se le asigna a cada funcionario de la institución.
Id_paciente	integer	Identificador que se le asigna a cada paciente.
Id_estado_paciente	Varchar	Identificador que se le asigna a cada estado del paciente.
Id_codigo	Varchar	Identificador que se le asigna a cada código.
Id_tipo_llegada	integer	Identificador que se le asigna a cada tipo de llegada.
Id_procedencia	integer	Identificador que se le asigna a cada procedencia.
Fecha_Hora_llegada	Timestamp with time zone	Refleja la fecha y hora de llegada de un paciente al cuerpo de guardia

TABLA cg_consulta

Nombre: cg_consulta		
Descripción: recoge todos los datos de la consulta a un paciente		
Atributo	**Tipo**	**Descripción**
Id_consulta(PK)	serial	Identificador que se le asigna a cada consulta
Id_conducta	integer	Identificador que se le asigna a cada conducta.
Id_recepcion	integer	Identificador que se le asigna a la recepción del cuerpo de guardia.
Id_tipo_consulta_cuerpo_guardia	integer	Identificador que se le asigna a cada tipo de consulta.
Id_paciente	integer	Identificador que se le asigna a cada paciente.
Id_destino	integer	Identificador que se le asigna a cada destino.
Id_funcionario	integer	Identificador que se le asigna a cada funcionario de la institución.
Id_codigo	varchar	Identificador que se le asigna a cada código.
Fecha_Hora_atencion	Timestamp with time zone	Refleja la fecha y hora de atención de la consulta de un paciente.

Fecha_Hora_salida	Timestamp with time zone	Refleja la fecha y hora de salida de la consulta de un paciente.
síntoma	varchar	Refleja los síntomas de un paciente.
diagnostico	varchar	Refleja el diagnóstico de un paciente.
tratamiento	varchar	Refleja el tratamiento de un paciente.
observación	varchar	Refleja las observaciones realizadas por el médico en la consulta.

TABLA cg_examen_fisico

Nombre: cg_examen_fisico		
Descripción: recoge los exámenes que prestan el servicio de enfermería		
Atributo	**Tipo**	**Descripción**
Id_consulta(PFK)	integer	Identificador que se le asigna a cada consulta.
Fecha_hora	Timestamp with time zone	Refleja la fecha y hora en que se realizaron los exámenes.
peso	Double presicion	Refleja el peso del paciente.
temperatura	Double presicion	Refleja la temperatura del paciente.

Frecuencia_cardiaca	Double presicion	Refleja la frecuencia cardiaca del paciente.
Frecuencia_respiratoria	Double presicion	Refleja la frecuencia respiratoria del paciente.
Tensión_arterial_minima	Double presicion	Refleja la tensión arterial mínima del paciente.
Tensión_arterial_maxima	Double presicion	Refleja la tensión arterial máxima de un paciente.

TABLA cg_remision

Nombre: cg_remision		
Descripción: Recoge las remisiones que puede tener un paciente en una consulta		
Atributo	**Tipo**	**Descripción**
Id_remision	serial	Identificador que se le asigna a cada tipo de consulta.
Id_consulta	integer	El tipo de consulta (urgencia, especializada, normal).
Id_paciente	integer	Identificador que se le asigna a cada paciente.
Id_especialidad	integer	Identificador que se le asigna a cada especialidad.
Id_funcionario	integer	Identificador que se le asigna a cada funcionario
Fecha_hora_remision	Timestamp with	Refleja la fecha y la hora en que se realizó

	time zone	la remisión.
descripción	varchar	Refleja las formas en que puede realizarse una remisión.

TABLA cg_cierre_guardia

Nombre: cg_cierre_guardia		
Descripción: recoge el tipo de consulta que se realizará el paciente		
Atributo	**Tipo**	**Descripción**
Id_cierre_guardia(PK)	serial	Identificador que se le asigna a cada guardia cerrada.
id_funcionario	integer	Identificador que se le asigna a cada funcionario de la institución.
ultima_fecha_cerrada	Timestamp with time zone	Refleja las consultas que fueron hechas en un día, las cuales no pueden ser cambiadas.

TABLA cfg_clasificacion

Nombre: cfg_clasificacion		
Descripción: recoge la prioridad de los pacientes en la cola de cuerpo de guardia		
Atributo	**Tipo**	**Descripción**
Id_codigo(PK)	Varchar	Identificador que se le asigna a cada código.
código	Varchar	Código que se le asigna a cada paciente

		en la recepción.
prioridad	integer	Prioridad que se le da a cada paciente según la gravedad de sus síntomas.

TABLA cfg_estado_paciente

Nombre: cfg_estado_paciente		
Descripción: recoge el estado del paciente en el cuerpo de guardia		
Atributo	**Tipo**	**Descripción**
id_estado_paciente(PK)	serial	Identificador que se le asigna a cada estado del paciente.
Tipo_estado	varchar	Refleja el estado en que se encuentra el paciente en el cuerpo de guardia.

TABLA cfg_tipo_llegada

Nombre: cfg_tipo_llegada		
Descripción: recoge las forma en que un paciente llega al hospital		
Atributo	**Tipo**	**Descripción**
id_tipo_llegada(PK)	serial	Identificador que se le asigna a cada tipo de llegada.
descripción	varchar	Refleja las formas en que un paciente puede llegar al cuerpo de guardia.

TABLA cfg_procedencia

Nombre: cfg_procedencia		
Descripción: recoge de donde viene el paciente cuando llega al hospital		
Atributo	**Tipo**	**Descripción**
id_procedencia(PK)	serial	Identificador que se le asigna a cada procedencia.
descripción	varchar	Refleja los lugares de donde puede llegar el paciente a la consulta.

TABLA cfg_destino

Nombre: cfg_destino		
Descripción: recoge el destino de los pacientes después de la consulta		
Atributo	**Tipo**	**Descripción**
id_destino(PK)	serial	Identificador que se le asigna a cada destino.
descripción	varchar	Refleja los destinos a donde se puede dirigir el paciente después de una consulta.

TABLA cfg_conducta_seguir

Nombre: cfg_conducta_seguir		
Descripción: recoge la conducta a seguir por el paciente después de la consulta		
Atributo	**Tipo**	**Descripción**

id_conducta(PK)	serial	Identificador que se le asigna a cada conducta.
descripción	varchar	Refleja las conductas a seguir por el paciente.

TABLA cfg_tipo_consulta_cg

Nombre: cfg_tipo_consulta_cg		
Descripción: recoge el tipo de consulta que se realizará el paciente		
Atributo	**Tipo**	**Descripción**
Id_tipo_consulta_cg(PK)	serial	Identificador que se le asigna a cada tipo de consulta.
descripción	varchar	El tipo de consulta (urgencia, especializada, normal).

2.8 Vista en la base de datos[5]

Una vista, es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla real, una vista consta de un conjunto de columnas y filas de datos con un nombre. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista.

Antes de crear una vista, deben considerarse las siguientes indicaciones:

[5] Para realizar este epígrafe se consultó de Microsoft Corporation (2007): Vistas (motor de base de datos), en http://technet.microsoft.com/es-es/library/ms190706(SQL.90).aspx, junio 2007

1. Sólo se puede crear vistas en la base de datos actual. Sin embargo, las tablas y las vistas a las que se haga referencia desde la nueva vista pueden encontrarse en otras bases de datos e incluso, en otros servidores, si la vista se define mediante consultas distribuidas.

2. Los nombres de las vistas deben seguir las reglas que se aplican a los identificadores y ser únicos para cada esquema. Además, el nombre debe ser distinto del de las tablas incluidas en ese esquema.

3. No se puede asociar con las vistas reglas ni definiciones default.

4. La consulta que define la vista no puede incluir la cláusula option que especifica una sugerencia de consulta.

5. No se pueden definir definiciones de índice de texto completo en las vistas.

6. No se pueden crear vistas temporales, ni vistas dentro de tablas temporales.

7. No se puede emitir consultas de texto completo en una vista, aunque una definición de vista puede incluir una consulta de texto completo si esta hace referencia a una tabla configurada para la indización de texto completo.

8. Se debe especificar el nombre de todas las columnas de la vista en el caso de que:
   - Alguna de las columnas de la vista derive de una expresión aritmética, una función integrada o una constante.
   - Dos o más columnas de la vista tuviesen, el mismo nombre (normalmente, debido a que la definición de la vista incluye una combinación y las columnas de dos o más tablas diferentes tienen el mismo nombre).

- Desee darle a una columna de la vista un nombre distinto del de la columna de la que deriva. También puede cambiar el nombre de las columnas en la vista. Una columna de una vista hereda los tipos de datos de la columna de la que deriva, aunque no cambie su nombre. Esta regla no se aplica cuando una vista se basa en una consulta que contiene una combinación externa, ya que las columnas pueden cambiar al permitir valores null.

2.8.1 Análisis de las vistas realizadas en la base de datos del módulo de cuerpo de guardia

Teniendo en cuenta, lo planteado en el epígrafe anterior se procedió a la elaboración de las vistas necesarias en la base de datos del módulo de cuerpo de guardia. Se realizaron dos vistas fundamentales por la necesidad que existe de mostrar los datos relacionados con varias tablas. Una de ellas es la vista cg_recepcion_view con el propósito de mostrar los datos que se encuentran en las tablas cg_recepcion, cfg_tipo_llegada, cfg_procedencia, cfg_estado_paciente, cfg_clasificacion[6]. Luego se procedió a realizar el procedimiento almacenado que permite seleccionar los datos de la vista[7].

Otra las vistas realizadas es cg_consulta_view. Esta muestra los datos relacionados con las tablas cg_consulta, cfg_clasificacion, cfg_tipo_consulta_cg, cfg_conducta_seguir, cfg_destino, cg_recepcion_view, cfg_estado_paciente[8]. Al igual que la vista anterior, esta requiere del uso de la sentencia SELECT, para seleccionar los datos que en ella se almacenan[9].

[6] Ver anexo 2
[7] Ver anexo 3
[8] Ver anexo 4
[9] Ver anexo 5

CAPITULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

En este capítulo, se tuvieron en cuenta diferentes factores para la creación de la solución propuesta, como los requisitos funcionales y no funcionales, su integración con otros módulos o sistemas y la arquitectura de la base de datos empleada para su buen funcionamiento. También se tuvo en cuenta el tipo de modelo de base da datos a utilizar para arribar así al diseño de la base de datos del módulo de cuerpo de guardia.

CAPÍTULO 3: VALIDACIÓN TEÓRICA DEL DISEÑO REALIZADO

El diseño de la base de datos, es fundamental para obtener cualidades como: la integridad de los datos, la seguridad, el tiempo de respuesta, la concurrencia. Estas cualidades deben ser mantenidas mediante la evaluación y el análisis, una vez que la base de datos entra en funcionamiento.

3.1 Integridad de una base de datos

El término integridad de datos se refiere a la corrección y completitud de los datos en una base de datos. Cuando los contenidos de una base de datos se modifican con sentencias insert, delete o update, la integridad de los datos almacenados puede perderse modificándose los datos tomando un valor incorrecto, como por ejemplo si se remite un paciente a una especialidad no existente y además los cambios en la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de energía.

Una de las funciones más importantes de un Sistema de Base de Datos (DBMS) relacional es preservar la integridad de la información almacenada en la mayor medida posible.

3.1.1 Integridad de la base de datos del módulo de cuerpo de guardia

En los procesos de salud es muy importante mantener la integridad y precisión de los datos ya que de acuerdo a ellos se actúa, ya sea en la toma de decisión médico - paciente, como en la toma de decisión de la dirección del sistema, en cuanto a los procesos involucrados en la salud y calidad de vida de la población.

En el caso de la base de datos del módulo de cuerpo de guardia, la integridad de los mismos se aseguró desde el diseño de la misma, cerciorándose de que cada tabla tuviera nombre único y

que sus llaves no tuviera un valor nulo, impidiendo algún tipo de error a la hora de implementar el sistema.

En el diseño se hace reseña a la integridad referencial, asegurando de que las llaves foráneas de una tabla, que son llaves primarias en otras, tengan el mismo valor.

También se aseguró la existencia de las tablas nomencladoras, que son las que poseen valores constantes, impidiendo así que se incorporen datos erróneos. Por ejemplo: la tabla cfg_clasificacion posee el código que va a tener el paciente en la recepción de acuerdo a la gravedad de sus síntomas, que puede ser verde (V), amarillo(A) o rojo(R), certificando así, que no se ingrese algún código no válido.

Por otra parte, el SQL Manager 2005 para PostgreSQL, da la opción de realizar y mantener la integridad de los datos a través de de la creación de dominios. Estos establecen un rango de valores a determinados atributos, impidiendo la entrada de valores no válidos. Por ejemplo:

CREATE DOMAIN "public"."cg_temperatura" **AS double precision NULL**;
ALTER DOMAIN "public"."cg_temperatura"
ADD CONSTRAINT "cg_temperatura_chk" **CHECK** ((VALUE >= (35)::**double precision**) **AND** (VALUE <= (42)::**double precision**));

Este dominio lo que plantea es que la temperatura de un paciente tiene que oscilar entre 35 y 42 grados. Así sucesivamente sucede con la frecuencia cardiaca, el peso, frecuencia respiratoria y la tensión arterial, cada uno de ellos con sus respectivos valores.

3.2 Normalización de la Base de datos

La normalización está compuesta de un conjunto de reglas que sirven para ayudar a los diseñadores a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede.

Una de las ventajas de la normalización de su base de datos es el consumo de espacio. Una base de datos normalizada puede ocupar menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia menor uso de espacio en disco.

Los propósitos de la normalización son:

1. Reducir o eliminar el almacenaje de datos duplicados.

2. Organizar los datos en una estructura eficiente y lógica.

El proceso de la normalización implica el determinarse de qué datos se deben almacenar en cada tabla de la base de datos, además de trabajar con pasos bien definidos, llamados las formas normales.

Básicamente se utilizan tres niveles de normalización: Primera Forma Normal (1NF), Segunda Forma Normal (2NF) y Tercera Forma Normal (3NF). Cada una de estas formas tiene sus propias reglas. Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización.

Primera forma normal

Una relación está en primera forma normal (1FN) si y solo si todos los dominios son atómicos, asegurando que sus elementos sean indivisibles. Es decir, no se tienen grupos de repetición o un conjunto de valores repetidos asociados a una misma tupla.

Segunda forma normal

Una relación está en segunda forma normal (2FN) si y sólo si está en 1FN y si no existe dependencia parcial.

Tercera forma normal

La tercera forma normal exige que todos los atributos no primos (que no pertenecen a ninguna clave) sean dependientes sólo de atributos pertenecientes a claves, y nunca de algún otro atributo no primo (SL 2000-2007).

Además de estas formas normales existen otras reglas para la normalización como Forma Normal Boyce-Codd, Cuarta Forma Normal, Quinta Forma Normal o Forma Normal de Proyección-Unión, Forma Normal de Proyección-Unión Fuerte, Forma Normal de Proyección-Unión Extra Fuerte y Forma Normal de Clave de Dominio

3.2.1 Normalización de la base de datos del módulo de cuerpo de guardia

Para determinar en que forma normal está la base de datos del módulo de cuerpo de guardia se realizó un estudio profundo de las tablas de su diseño.

1. Determinando si la base de datos está en primera forma normal. Para que la base de datos esté en primera forma normal ninguno de los atributos puede ser multivaluado.

Anteriormente en la tabla cg_examen_fisico se tenía el atributo tension_alterial, como se muestra en la figura 1, la cual puede ser máxima y mínima. Es por ello que se determinó poner a ambos como atributos de la tabla, quedando de la siguiente forma:

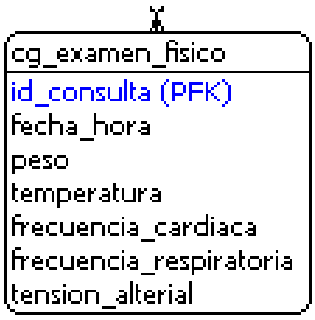


Figura 1

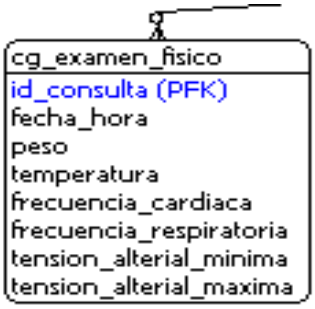


Figura 2

2. Determinando si la base de datos está en segunda forma normal.

Estando la base de datos en primera forma normal, se determina si está en segunda forma normal. Está se basa en el concepto de dependencia total y se aplica a las tablas que posean más de una llave primaria.

Remontándose al diseño de la base de datos expuesto en el capítulo 2, se puede apreciar que la tabla cg_remision posee 2 llaves (ver figura 3), una primaria y una foránea, pero esta última es como un atributo más de la entidad.

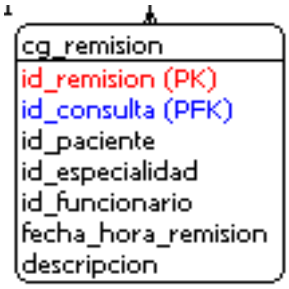


Figura 3

3. Determinando si la base de datos está en tercera forma normal.

Estando la base de datos normalizada en segunda forma normal, se procede al análisis para determinar si está en tercera forma normal, la cual se basa en el concepto de dependencia transitiva comprobando si existe dependencia entre los atributos de una misma tabla. Luego del estudio exhaustivo realizado se estipuló que no existía tal dependencia, determinando así, que la base de datos del módulo de cuerpo de guardia está en tercera forma normal.

Se considera que las primeras tres formas proveen suficiente nivel de normalización para cumplir con las necesidades de la base de datos. Hay que tener cuidado, pues normalizar demasiado puede conducir a tener una base de datos ineficiente y hacer a su esquema demasiado complejo para trabajar.

3.3 Análisis de la redundancia de información en el módulo de cuerpo de guardia

Para el desarrollo de una base de datos hay que tener en cuenta una serie de aspectos que son fundamentales para su buen funcionamiento. Uno de ellos es la redundancia de información, que

es la tendencia a almacenar información repetida en la base de datos ocupando espacio en memoria y haciendo el trabajo más difícil para los que laboren directamente con la aplicación.

Es por ello que el diseño juega un papel importante, pues ahí es donde se analiza de forma exhaustiva si existe o no redundancia de datos. En el caso específico del módulo de cuerpo de guardia toda la información que se maneja es entorno a los datos de los pacientes y a las consultas que se imparten, ya sean especializadas o no. Es por ello que en un principio se tenían varias tablas que almacenaban los mismos datos, como por ejemplo: cg_consulta especializada y la tabla cg_consulta, con la excepción de que en la tabla cg_consulta especializada se reflejaba también la especialidad en la cual se impartía la consulta.

Al percibir que esto podría traer consigo trabajo extra y muchas complicaciones innecesarias, se decidió crear una tabla cfg_tipo_consulta_cg que almacena el tipo de consulta, sean especializadas o no, sin necesidad de guardar nuevamente los datos de los pacientes y se acordó dejar solo la tabla cg_consulta (véase diseño de la base de datos).

3.4 Análisis de la seguridad de la base de datos

Al hablar de seguridad hay que centrarse en la información misma, aunque a menudo se hable de seguridad informática, de seguridad de los sistemas de información o de seguridad de las tecnologías de la información (S.A. 1997).

En cualquier caso hay tres aspectos principales.

1. La confidencialidad: se cumple cuando solo las personas autorizadas pueden conocer los datos o la información correspondiente.

2. La integridad: consiste en que sólo las personas autorizadas puedan cambiar los datos. Además deben quedar reportes para control posterior y para auditoria.

3. La disponibilidad: se cumple si las personas autorizadas pueden acceder a tiempo a la información.

Las aplicaciones de gestión hospitalaria tratan un conjunto considerable de datos sensibles. Mantener la seguridad y confidencialidad de estos cobra una vital importancia, sobre todo cuando se tienen de manera centralizada en un servidor de bases de datos.

Específicamente en el módulo de cuerpo de guardia, la seguridad de la base de datos se establece mediante el gestor utilizado, proporcionando diferentes opciones:

User Manager para crear usuarios, eliminarlos y editarlos. Estos usuarios son los que van a tener acceso a la base de datos.
Group Manager para administrar grupos de usuarios y el número de usuarios en cada grupo.
Grant Manager para conceder al usuario privilegios en las bases de datos seleccionados, tablas, campos, funciones y vistas etc.

En el caso del proyecto en el cual se desarrolla el módulo de cuerpo de guardia en conjunto con otros módulos, llamado gestión hospitalaria, se tiene la propuesta de un módulo de seguridad para el sistema, la cual debe comenzar desde la capa más baja (capa de datos) y no depender solamente de la seguridad brindada por el gestor de bases de datos.

Este módulo dará la posibilidad de autenticar al usuario, proceso mediante el cual el servidor de la base de datos se asegura de que la persona que está solicitando acceso a la base de datos es en realidad quien dice ser. También validará permisos necesarios para ejecutar la base de

datos, determinando los beneficiarios que tendrán acceso a la misma, así como los roles y autorizaciones de cada uno de ellos.

3.6 Trazabilidad de las acciones

La creación de una base de datos lleva consigo la toma de un conjunto de medidas de seguridad para lograr una mejor eficacia y validez de los datos que se almacenan en ella. Para una precisa observación del uso de los datos, se hace necesario tener el control de las acciones que realiza un determinado usuario en la base de datos.

De esto se encarga la trazabilidad que es la capacidad que tiene un sistema de base de datos para rastrear, reconstruir o establecer relaciones entre objetos monitoreados, para identificar y analizar situaciones específicas o generales en los mismos.

La trazabilidad de las acciones es un punto fundamental en el funcionamiento del sistema. En esta, las trazas juegan un papel importante pues ellas permiten visualizar las llamadas de funciones. Es un gran indicador para verificar que no se están realizando llamadas absurdas a la base de datos. También permite ver los tiempos de ejecución de las funciones.

La trazabilidad de las acciones de la base de datos del módulo de cuerpo de guardia es controlada por el gestor de base de datos utilizado, pues todas las actividades que realiza un determinado usuario son registradas en los ficheros “logs”, habilitados en los ficheros de configuración “postgres.conf.”. Hay que aclarar que solo se registrarán las acciones realizadas por los usuarios creados a nivel de base de datos.

Por otra parte el módulo de seguridad también estará a cargo de la trazabilidad de las acciones del proyecto de Gestión Hospitalaria, el cual es el responsable de controlar todas las acciones

que se realizan en la base de datos. Este módulo posee las funcionalidades para el control de los usuarios que acceden a la misma, la hora y la fecha en que accedió y las actividades o cambios que realizó.

En este capitulo se tuvieron en cuenta aspectos fundamentales en la confección de la base de datos del módulo de cuerpo de guardia, como la integridad y seguridad de los datos. También se consideró importante eliminar la redundancia de información, además de normalizar la base de datos a un nivel factible para su desarrollador. Otro punto importante que se ha tratado fue la trazabilidad de las acciones, la cual permite establecer y controlar cuales son los usuarios que pueden acceder a la base de datos y las acciones que realiza en la misma.

CONCLUSIONES

Con este trabajo se llegó a las siguientes conclusiones:

1. Se realizó un estudio de los procesos vinculados al módulo de cuerpo de guardia en los hospitales cubanos para conocer los problemas existentes en ellos.
2. Se realizó el levantamiento de requisitos necesarios para lograr un diseño óptimo de la base de datos.
3. Se hizo un estudio de las herramientas a utilizar para el desarrollo de la base de datos, lo que permitió obtener un conocimiento pleno de las mismas, facilitando el trabajo del desarrollador.
4. Se realizaron las consultas necesarias para el buen funcionamiento de la base de datos

De esta forma, se da cumplimiento al objetivo general logrando el diseño e implementación de una base de datos que almacene y organice de forma segura la información relacionada con los procesos que se realizan en el módulo de cuerpo de guardia.

RECOMENDACIONES

- Seguir perfeccionando el sistema para un mejor desempeño del mismo.

- La arquitectura de la base de datos, se recomienda que en un futuro sea distribuida, por los beneficios que tiene la misma para el manejo de los datos y la seguridad de los mismos.

BIBLIOGRAFÍA

ASSETTA, A., FERNÁNDEZ ROMERO, D. , ROSELL, S., SALDAÑA, A., STAVISKY DE FELDMAN, L., URE, J. *SISTEMAS DE INFORMACION HOSPITALARIA (SIH)*, 2004. [Disponible en: http://www.medicos-municipales.org.ar/bc1104.htm

CASTRO, W. *GalenHos, Sistema Básico de Gestion Hospitalaria, Manual de Sistema* 2005. [Disponible en: http://www.praes.org/docs-pdf/publica/4f.pdf

CERRITOS, M. F. J. F. P., MTRA. FLORINA GATICA LARA. *Manual de Introducción a la Informática*

Médica 2003. [Disponible en: http://educacion.salud.gob.mx/cursos/informatica/HIS/his.pdf

COPYRIGHT, C. *MEDISYS. Versión Cliente-Servidor*, 1998. [Disponible en: http://www.sld.cu/instituciones/cedisap/Medclien.htm

COPYRIGHT, O. *Base de datos distribuidas*, 2007. [Disponible en: http://html.rincondelvago.com/bases-de-datos-distribuidas_1.html

CUSCO. *Postgresql 8.2*, 2004-2007. [Disponible en: http://www.buayacorp.com/archivos/postgresql-82/

Estadillo de Urgencias. Disponible en: http://www.chospab.es/ykonos/estadillo_urgencias.htm

GMBH., C. *Microsoft SQL server 2000*, 2007. [Disponible en: http://www.ciao.es/Microsoft_SQL_server_2000__139445

GROUP, P. G. D. *PostgreSQL*, 2007. [Disponible en: http://es.wikipedia.org/wiki/PostgreSQL

GUZMÁN, I. G. R. D. *BASES DE DATOS, MODELO EN RED GENERAL*, 2000-2001. [Disponible en: http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001_IGarcia.pdf

LATORILLA, E. *Care2x*, 2004. [Disponible en: http://care2x.ourproject.org/

S.A., M. C. *SEGURIDAD DE LAS BASES DE DATOS*, 1997. [Disponible en: http://www.monografias.com/trabajos26/seguridad-base-datos/seguridad-base-datos2.shtml#segur

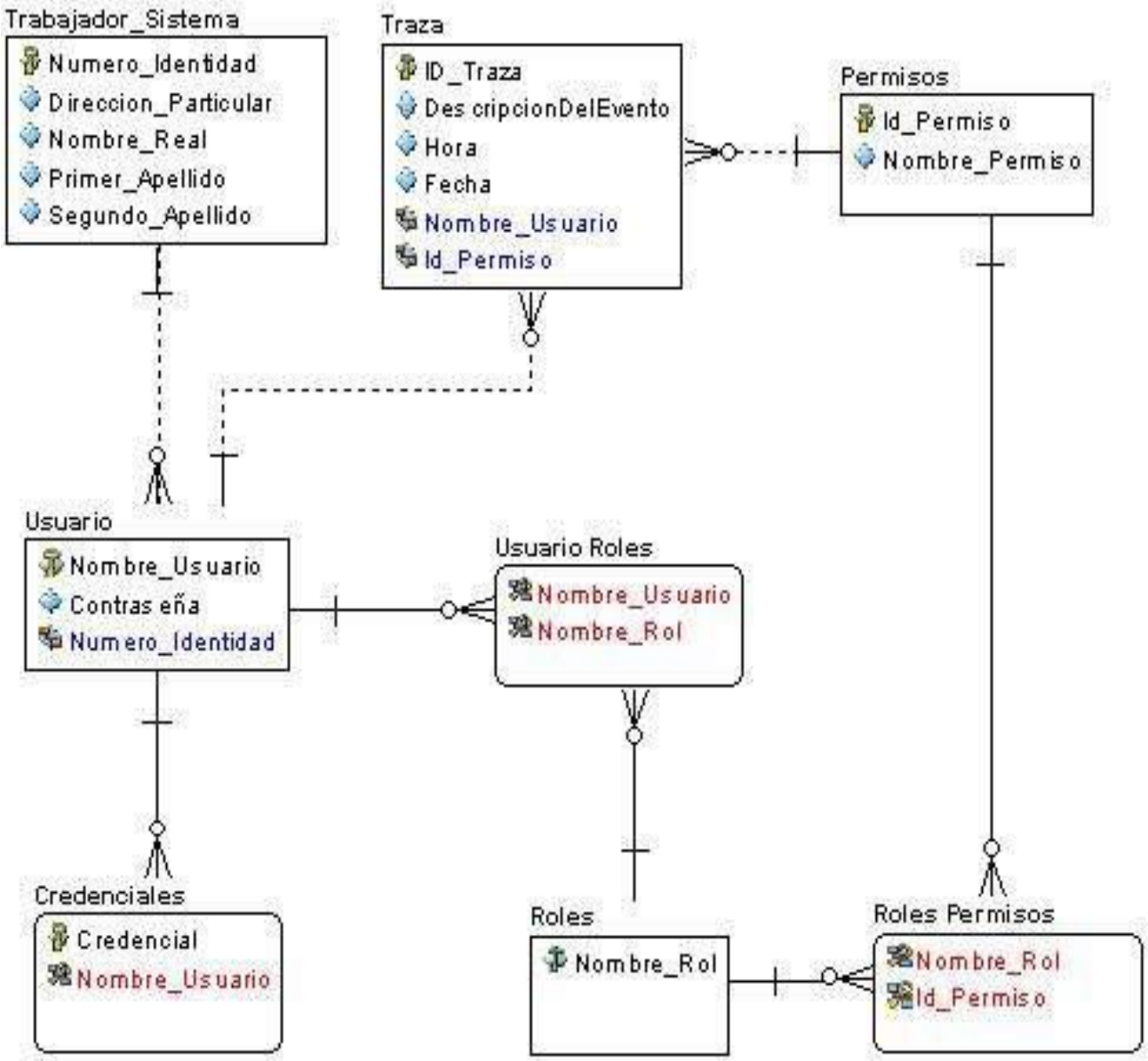
SL, E. W. *Normalización de Bases de Datos.*, 2000-2007. [Disponible en: http://www.trucostecnicos.com/trucos/ver.php?id_art=278

WIKIPEDIA. *Base de datos jerárquica*, 2007a. [Disponible en: http://es.wikipedia.org/wiki/Base_de_datos_jer%C3%A1rquica#Limitaciones_del_modelo_jer.C3.A1rquico

---. *Sistema de gestión de base de datos*, 2007b. [Disponible en: http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos

ANEXOS

Anexo 1. Diseño del módulo de seguridad del proyecto de gestión hospitalaria.



Anexo 2. Vista cg_recepcion_view

```
CREATE OR REPLACE VIEW "public"."cg_recepcion_view" (
    id_recepcion,
    id_tipo_llegada,
    forma_llegada,
    id_procedencia,
    procedencia,
    id_estado_paciente,
    estado_paciente,
    id_codigo,
    clasificacion,
    id_funcionario,
    id_persona,
    fecha_hora_llegada,
    hora_llegada,
    tiempo_espera)
AS
SELECT cg_recepcion.id_recepcion, cg_recepcion.id_tipo_llegada,
    cfg_tipo_llegada.descripcion AS forma_llegada, cg_recepcion.id_procedencia,
    cfg_procedencia.descripcion AS procedencia,
    cg_recepcion.id_estado_paciente, cfg_estado_paciente.descripcion AS
    estado_paciente, cg_recepcion.id_codigo, cfg_clasificacion.codigo AS
    clasificacion, cg_recepcion.id_funcionario, cg_recepcion.id_persona,
```

```
    cg_recepcion.fecha_hora_llegada,
    "substring"((cg_recepcion.fecha_hora_llegada)::text, 12, 8) AS
    hora_llegada, age(now(), (cg_recepcion.fecha_hora_llegada)::timestamp with
    time zone) AS tiempo_espera
FROM ((((cg_recepcion JOIN cfg_tipo_llegada ON ((cg_recepcion.id_tipo_llegada =
    cfg_tipo_llegada.id_tipo_llegada))) JOIN cfg_procedencia ON
    ((cg_recepcion.id_procedencia = cfg_procedencia.id_procedencia))) JOIN
    cfg_estado_paciente ON ((cg_recepcion.id_estado_paciente =
    cfg_estado_paciente.id_estado_paciente))) JOIN cfg_clasificacion ON
    ((cg_recepcion.id_codigo = cfg_clasificacion.id_codigo)))
ORDER BY cfg_clasificacion.prioridad, cg_recepcion.fecha_hora_llegada;
```

Anexo 3. Procedimiento almacenado para la vista cg_recepcion_view

```
SELECT * FROM  cg_recepcion_view
        WHERE
                ( id_recepcion = pid_recepcion OR pid_recepcion is null ) and
                ( id_tipo_llegada = pid_tipo_llegada OR pid_tipo_llegada is null ) and
                ( forma_llegada = pforma_llegada OR pforma_llegada is null ) and
                ( id_procedencia = pid_procedencia OR pid_procedencia is null ) and
                ( procedencia = pprocedencia OR pprocedencia is null ) and
                ( id_estado_paciente = pid_estado_paciente OR pid_estado_paciente is null ) and
                ( estado_paciente = pestado_paciente OR pestado_paciente is null ) and
                ( id_codigo = pid_codigo OR pid_codigo is null ) and
                ( clasificacion = pclasificacion OR pclasificacion is null ) and
                ( id_funcionario = pid_funcionario OR pid_funcionario is null ) and
```

(id_persona = pid_persona **OR** pid_persona **is null**) **and**
(fecha_hora_llegada = pfecha_hora_llegada **OR** pfecha_hora_llegada **is null**) **and**
(hora_llegada = phora_llegada **OR** phora_llegada **is null**) **and**
(tiempo_espera = ptiempo_espera **OR** ptiempo_espera **is null**)**and**
(fecha_hora_llegada >= pfecha1 **OR** pfecha1 **is null**) **and**
(fecha_hora_llegada <= pfecha2 **OR** pfecha2 **is null**)

Anexo 4. Vista cg_consulta_view

CREATE OR REPLACE VIEW "public"."cg_consulta_view" (
id_consulta,
id_tipo_consulta_cg,
tipo_consulta,
id_conducta_seguir,
conducta,
id_destino,
destino,
id_codigo,
clasificacion,
id_diagnostico,
id_persona,
id_estado_paciente,
estado,
id_funcionario,
sintoma,
id_recepcion,

```
    tratamiento,
    observacion,
    fecha_hora_atencion,
    fecha_hora_salida,
    id_procedencia,
    procedencia,
    id_tipo_llegada,
    forma_llegada,
    fecha_hora_llegada)
AS
SELECT cg_consulta.id_consulta, cg_consulta.id_tipo_consulta_cg,
    cfg_tipo_consulta_cg.descripcion AS tipo_consulta,
    cg_consulta.id_conducta_seguir, cfg_conducta_seguir.descripcion AS
    conducta, cg_consulta.id_destino, cfg_destino.descripcion AS destino,
    cg_consulta.id_codigo, cfg_clasificacion.codigo AS clasificacion,
    cg_consulta. id_diagnostico, cg_consulta.id_persona,
    cg_consulta.id_estado_paciente, cfg_estado_paciente.descripcion AS estado,
    cg_consulta.id_funcionario, cg_consulta.sintoma, cg_consulta.id_recepcion,
    cg_consulta.tratamiento, cg_consulta.observacion,
    cg_consulta.fecha_hora_atencion, cg_consulta.fecha_hora_salida,
    cg_recepcion_view.id_procedencia, cg_recepcion_view.procedencia,
    cg_recepcion_view.id_tipo_llegada, cg_recepcion_view.forma_llegada,
    cg_recepcion_view.fecha_hora_llegada
FROM ((((((cg_consulta JOIN cfg_clasificacion ON ((cg_consulta.id_codigo =
    cfg_clasificacion.id_codigo))) JOIN cfg_tipo_consulta_cg ON
    ((cg_consulta.id_tipo_consulta_cg =
    cfg_tipo_consulta_cg.id_tipo_consulta_cg))) JOIN cfg_conducta_seguir ON
```

```
   ((cg_consulta.id_conducta_seguir =
   cfg_conducta_seguir.id_conducta_seguir))) JOIN cfg_destino ON
   ((cg_consulta.id_destino = cfg_destino.id_destino))) JOIN cg_recepcion_view
   ON ((cg_consulta.id_recepcion = cg_recepcion_view.id_recepcion))) JOIN
   cfg_estado_paciente ON ((cfg_estado_paciente.id_estado_paciente =
   cg_consulta.id_estado_paciente)))
ORDER BY cg_consulta.id_estado_paciente;
```

Anexo 5. Procedimiento almacenado para la vista cg_consulta_view

```
SELECT * FROM  cg_consulta_view
        WHERE
                ( id_consulta = pid_consulta OR pid_consulta is null ) and
                ( id_tipo_consulta_cg = pid_tipo_consulta_cg OR pid_tipo_consulta_cg is null )
and
                ( tipo_consulta = ptipo_consulta OR ptipo_consulta is null ) and
                ( id_conducta_seguir = pid_conducta_seguir OR pid_conducta_seguir is null ) and
                ( conducta = pconducta OR pconducta is null ) and
                ( id_destino = pid_destino OR pid_destino is null ) and
                ( destino = pdestino OR pdestino is null ) and
                ( id_codigo = pid_codigo OR pid_codigo is null ) and
                ( clasificacion = pclasificacion OR pclasificacion is null ) and
                ( id_diagnostico = pid_diagnostico OR pid_diagnostico is null ) and
                ( id_persona = pid_persona OR pid_persona is null ) and
                ( id_estado_paciente > pid_estado_paciente OR pid_estado_paciente is null ) and
                ( estado = pestado OR pestado is null ) and
                ( id_funcionario = pid_funcionario OR pid_funcionario is null ) and
```

(sintoma = psintoma **OR** psintoma **is null**) **and**
(id_recepcion = pid_recepcion **OR** pid_recepcion **is null**) **and**
(tratamiento = ptratamiento **OR** ptratamiento **is null**) **and**
(observacion = pobservacion **OR** pobservacion **is null**) **and**
(fecha_hora_atencion = pfecha_hora_atencion **OR** pfecha_hora_atencion **is null**) **and**
(fecha_hora_salida = pfecha_hora_salida **OR** pfecha_hora_salida **is null**) **and**
(id_procedencia = pid_procedencia **OR** pid_procedencia **is null**) **and**
(procedencia = pprocedencia **OR** pprocedencia **is null**) **and**
(id_tipo_llegada = pid_tipo_llegada **OR** pid_tipo_llegada **is null**) **and**
(forma_llegada = pforma_llegada **OR** pforma_llegada **is null**) **and**
(fecha_hora_llegada = pfecha_hora_llegada **OR** pfecha_hora_llegada **is null**) **and**
(fecha_hora_atencion >= pfecha_atencion_buscar1 **OR** pfecha_atencion_buscar1 **is null**)
and (fecha_hora_atencion < pfecha_atencion_buscar2 **OR** pfecha_atencion_buscar2 **is null**)
and
(fecha_hora_salida >= pfecha_salida_buscar1 **OR** pfecha_salida_buscar1 **is NULL**) **and**
(fecha_hora_salida < pfecha_salida_buscar2 **OR** pfecha_salida_buscar2 **is NULL**)

1
2
3
4