Universidad de las Ciencias Informáticas

Facultad 7

Título: Sistema de almacenamiento de información del módulo Archivo de Historia Clínica del Sistema de Información Hospitalaria.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Jenlys Ramírez Nepomuceno.

Tutor: Ing. Paúl Pérez Zurita.

Ciudad de La Habana, junio de 2007.

Ahora es nuestro tiempo de brillar.

El tiempo en que alcanzamos nuestros sueños y las posibilidades son inmensas.

Ahora es tiempo para todos nosotros de convertirnos en gente que siempre soñamos ser.

DECLARACIÓN DE AUTORÍA

Declaro ser autor 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 5 días del mes de junio de 2007.

Jenlys Ramírez Nepomuceno

_____

Ing. Paúl Pérez Zurita.

_____

Agradecimientos

A mis padres por ser excelentes, enseñarme siempre a luchar y nunca dejarme vencer ante las dificultades. Por apoyarme en todo, mostrándome siempre el camino correcto a seguir y brindándome su inmenso cariño.

A mis abuelos y toda mi familia en general, por su comprensión y su cariño y enseñarme siempre a luchar por todas aquellas cosas que son importantes en la vida y a no dejarme vencer ante las dificultades.

A todos mis compañeros, que estuvieron conmigo en todo momento y siempre me ayudaron.

A las personas que han colaborado con mi formación profesional.

A mi tutor, por ayudarme en el desarrollo de mi tesis.

De corazón, gracias.

Dedicatoria

A mi familia y en especial a mis padres…

RESUMEN

Dentro del proceso de informatización de la sociedad cubana, se ha priorizado el sector de la salud, para lo cual se llevan a cabo una serie de tareas con el objetivo de brindar soluciones que permitan mejoras en la calidad y gestión de la información en los hospitales y centros médicos del país en general.

Con el desarrollo de este trabajo, se persigue el objetivo de diseñar una Base de Datos (BD) que facilite el acceso a la información de los pacientes, mediante la gestión de las Historias Clínicas en el departamento Archivo de Historias Clínicas en los hospitales del país.

Para estructurar y diseñar un modelo de la BD que se ajuste y cumpla con lo establecido se utilizan diferentes herramientas de desarrollo: CASE Studio para el modelado de la misma, EMS SQL Manager for PostgreSQL para la administración y desarrollo del Sistema Gestor de Base de Datos utilizado, PostgreSQL 8.2.

Con el desarrollo de la BD, se espera lograr el incremento de la capacidad organizativa de la información gestionada en el Archivo de Historias Clínicas en los hospitales cubanos, así como el aumento de la calidad de todos los servicios que se gestionan.

TABLA DE CONTENIDOS.

INTRODUCCIÓN

El proceso de informatización del Sistema Nacional de Salud (SNS) consiste en un conjunto de métodos, técnicas, procedimientos y actividades gerenciales dirigidas al manejo de la información en salud, comprende el estado de salud de la población, la información sobre el conocimiento de las ciencias de la salud y en general para la toma de decisiones, clínico-epidemiológicas, operativas y estratégicas.

El presente impone la implementación de un proceso similar orientado a la superación profesional, a la automatización de los servicios, a la investigación y al apoyo en la toma de decisiones en toda la actividad de salud.

Actualmente el Ministerio de Salud Pública (MINSAP) ha definido a la informatización como una de sus prioridades y ha convocando para ello a un grupo de instituciones propias de sector, del Ministerio de Informática y Comunicaciones y de otros organismos de la administración central del estado, para definir de conjunto la estrategia a desarrollar. En algunos casos, se ha tomado como punto de partida sistemas ya desarrollados en el país en el marco de aquella primera estrategia de desarrollo.

La Universidad de las Ciencias Informáticas (UCI) utiliza una fuerte infraestructura tecnológica, informática y telemática para la formación de profesionales altamente calificados y la producción de aplicaciones informáticas. El uso eficiente de esta infraestructura tecnológica constituye un reto para la Universidad y por esta razón una de las tareas fundamentales que se llevan a cabo es la de informatizar los procesos que forman parte de la sociedad.

La UCI forma parte esencial de las instituciones que participan en la informatización de la sociedad, que no puede excluir al sistema de salud cubano, por lo que se hace necesario la informatización de los procesos de gestiones hospitalarias y dentro de estas la gestión de las Historias Clínicas (HC).

El sistema informático hospitalario, para el manejo de los datos primarios arma sus campos, partiendo de las normativas para la confección de la HC del paciente. Este conjunto de datos básicos, forma parte del conocimiento necesario para poner en funcionamiento el sistema institucional de atención y cuidado al

paciente y familia. Después en el mismo, esta información estructura la base de los Sistemas de Información Hospitalaria (SIH), y fundamenta el procesamiento de todo otro dato e información posterior.

Durante las décadas de los 80´s y los 90´s se realizaron múltiples esfuerzos para perfeccionar los SIH, específicamente en los procesos de recolección, distribución, interpretación y almacenamiento de los datos. Aunque se han obtenido grandes progresos en esta empresa, los SIH actuales distan de ser perfectos y la realidad es que se desarrollan numerosas versiones para mejorar continuamente los ya implementados.

El SIH está compuesto por varios sistemas, dentro de los cuales resalta el clínico, compuesto a su vez por los subsistemas clínicos departamentales y clínicos de apoyo, entre los cuales se encuentra el sistema de archivos que proporciona el cuidado y control de la HC del centro hospitalario.

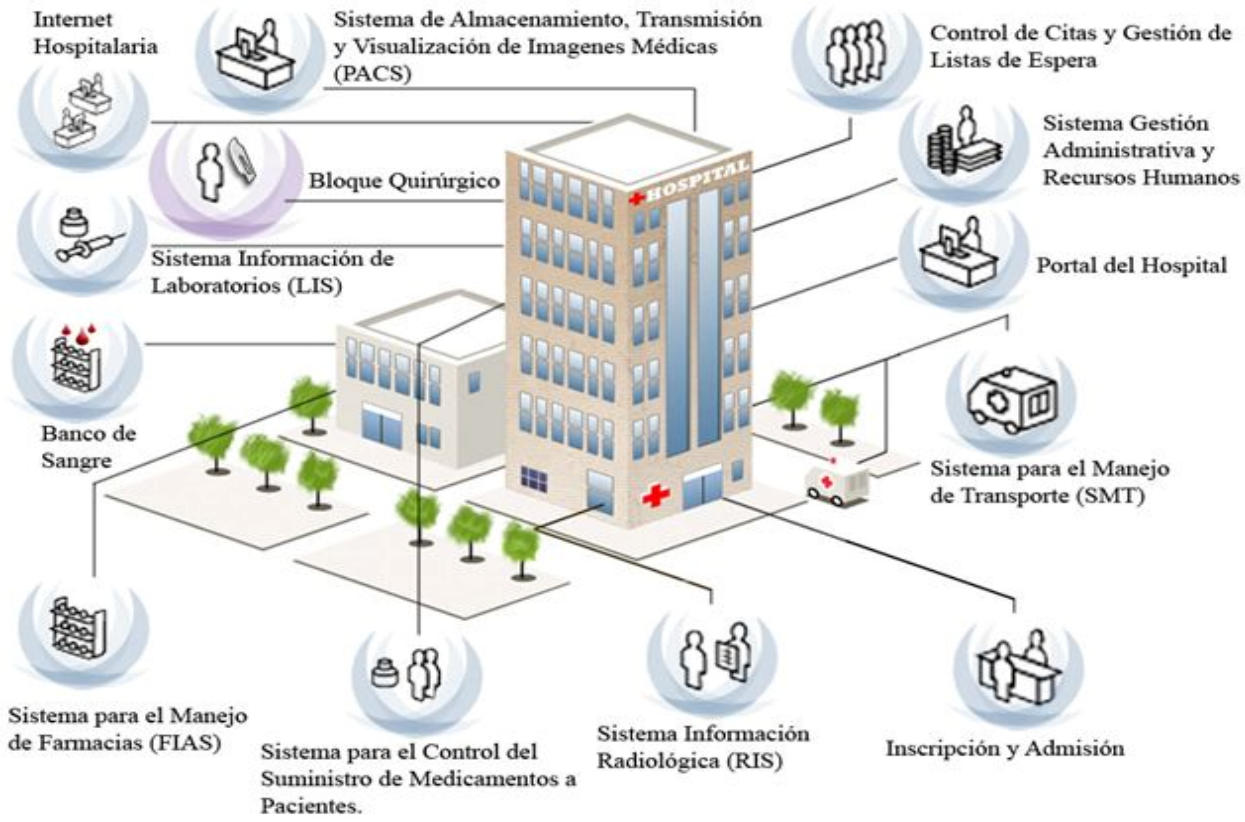


Fig. 1: Infraestructura de un SIH.

Actualmente, la sociedad se encuentra inmersa en un proceso progresivo y sostenido, donde el uso de las nuevas tecnologías en el procesado de la información conduce a una transición, de la HC tradicional en el papel, a su sustitución o coexistencia con otros soportes magnéticos o digitales que permitan almacenar y procesar gran cantidad de datos.

La irrupción de la informática en la actividad diaria, está permitiendo en buena medida al médico gestionar gran volumen de información de forma rápida, facilitando el acceso a datos concretos; la conexión con otras instituciones y bancos de datos. Permitiendo la continuidad en la atención, con independencia del lugar donde se encuentre el paciente; esto proporciona ahorro económico y mejora de las condiciones de trabajo y, por último, una reorganización más efectiva de las actividades.

En el país actualmente, existen programas de aplicación (*software*) que regulan el manejo de la información de las HC en los hospitales cubanos. A su vez en el mundo se encargan del trabajo en los departamentos de Archivo/HC, *software* de engorrosa manipulación y no adaptables al entorno de salud, además de estar desarrollados bajo la política de *software* propietario entre los cuales se pueden mencionar el Care2x y el Open HIS.

Teniendo en cuenta lo anteriormente expuesto ***el problema radica en****:* ¿Cómo modelar un sistema de bases de datos que facilite el acceso a la información de los pacientes mediante la gestión de las HC en el departamento de Archivo/HC de los centros hospitalarios de Cuba?

Para esto, es necesario un estudio profundo y detallado acerca de cuales son los aspectos con los que debe contar el sistema de archivo en un SIH. Además, de la tecnología para el desarrollo de aplicaciones y el análisis del sistema gestor de bases de datos a utilizar. Es imprescindible realizar un estudio de los diferentes artefactos de análisis y diseño en términos de diseño de bases de datos con el fin de lograr un diseño robusto y capaz de enfrentar los fallos del sistema y que solucione los problemas actuales.

En correspondencia con el problema, el ***objeto de estudio*** lo constituyen: *los procesos de gestión de la información en hospitales de Cuba.*

El ***campo de acción*** está centrado en los sistemas de registros médicos del paciente (Archivo/HC) automatizados en los centros hospitalarios del país.

El ***objetivo de la investigación*** es Diseñar una Base de Datos que facilite el acceso a la información de los pacientes mediante la gestión de las Historias Clínicas en el departamento de Archivo/HC de los centros hospitalarios del país.

Para dar cumplimiento al objetivo anteriormente planteado, se definen las **siguientes tareas:**

- Estudiar acerca de los SIH que se utilizan en otros países y que conllevan al mejor funcionamiento del módulo de Archivo de HC en los centros hospitalarios en los que se encuentran implantados los mismos.
- Estructurar y diseñar un modelo conceptual de la base de datos que se ajuste y cumpla con lo establecido.
- Implementar las funciones (procedimientos almacenados), vistas y dominios del módulo Registros Médicos del Paciente.
- Hacer una validación funcional del diseño de la base de datos propuesto.

El contenido de este trabajo se encuentra estructurado en tres capítulos:

El Capítulo 1. Fundamentación teórica: Incluye un estado del arte del tema bases de datos a nivel internacional, nacional y en la UCI. Las tendencias, técnicas, tecnologías, metodologías relacionadas con este tema en la actualidad y en las que se apoya para la solución del problema que se enfrenta, aspectos estos en que es necesario profundizar.

El Capítulo 2. Descripción y análisis de la solución propuesta: Incluye una selección y argumentación de los requisitos funcionales y no funcionales del sistema propuesto. Además se describe y fundamenta la arquitectura utilizada para la base de datos. Se detalla el diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño.

El Capítulo 3. Validación del diseño realizado: Incluye una validación teórica y funcional del diseño de la base de datos, donde se tratarán aspectos como la integridad, normalización de la base de datos, análisis de redundancia de información, seguridad y la trazabilidad de las acciones.

Cada capítulo es iniciado por una breve introducción donde se dan a conocer los temas que se desarrollarán durante el mismo. Finaliza con las conclusiones, en las que se plantean los resultados obtenidos.

CAPITULO 1. FUNDAMENTACIÓN TEÓRICA.

En este capitulo se hará un estudio detallado de los principales conceptos y tecnologías utilizadas en el desarrollo del Módulo de Archivo/HC del SIH. Se abordan temas relacionados con las Bases de Datos (BD), el uso de Sistemas de Gestión de Bases de Datos (SGBD), etc. Persigue el propósito de dar a conocer el estado del arte en los SGBD y la elección de las tecnologías utilizar en el desarrollo de este trabajo.

1.1 Sistema de Gestión de Archivos.

Un sistema de gestión de archivos es aquel sistema informático que brinda servicios a los usuarios en el uso de archivos. Haciendo cumplir una serie de objetivos como garantizar que el dato en el archivo es válido, optimizar el rendimiento, proveer un soporte de entrada y salida y así tener control de los datos archivados y minimizar o eliminar la pérdida o destrucción de datos.

1.2 Sistema de Gestión de Archivos Hospitalario.

Un sistema de gestión de archivo hospitalario no equidista de los objetivos principales de la gestión de un archivo pero como bien lo dice está guiado sobre el ambiente de los hospitales donde el principal objeto a organizar, almacenar y adquirir adecuadamente son las HC siendo estas “el elemento esencial de acreditación por parte del médico de su conducta con el paciente en todo momento, al reflejar toda la información relacionada con la asistencia dispensada al propio paciente”**.** [1]

La Historia Clínica presenta una gran importancia para la práctica médica donde la tecnología de la información ofrece una opción para hacer de esta, un documento realmente eficiente y por tanto útil, lo que proporciona la ventaja de resolver los principales problemas existentes en los archivos de HC:

a) Mal trabajo con la información médica que contienen las HC propiciando así la pérdida de ésta.
b) Poco control de la información dentro del archivo.

1.3 Bases de Datos.

En los inicios de las BD, las estructuras de datos eran muy simples y el costo para procesar y acceder a la información era muy alto. A la par, con el desarrollo de la tecnología han evolucionado los procedimientos y estructuras para almacenar y procesar la información.

En las nuevas investigaciones sobre la tecnología de BD, surgió la necesidad de independizar la estructura de la información de los procedimientos, dando lugar al concepto de base de datos, conjunto de datos persistentes que es utilizado por los sistemas de aplicación, de alguna empresa dada. [2] En este contexto, el término empresa se refiere a cualquier organización, lo cual incluye la posibilidad de que sea un solo individuo que es usuario de la BD.

La tecnología de las modernas bases de datos provee los medios para almacenar, administrar y acceder a semejantes cantidades de información. Sin embargo, las bases de datos no son un fenómeno nuevo. De hecho, Herman Hollerith mecanizó el almacenamiento del censo de EE.UU. de 1890 en lo que de alguna forma se considera la primera base de datos significativa "computarizada". [3]

Desde hace mucho tiempo, la mayoría de la información de todo el mundo está contenida en bases de datos, de aquí, su merecida importancia y desarrollo vertiginoso para poder estar a la par con las necesidades de administración y transacciones diarias. Por estas razones los administradores y diseñadores de bases de datos enfrentan numerosos retos para expresar y mantener el complejo ambiente de los datos, sus relaciones, así como la seguridad, integridad y recuperación rápida y fiable de estos.

1.3.1 Rendimiento y escalabilidad.

Cuando se va a realizar una aplicación de BD, se tiene que pensar en todas las partes que conforman la misma, desde el diseño lógico de los datos, hasta el gestor de BD más adecuado y la arquitectura del servidor que se dispone.

En este tipo de aplicaciones, son muy importantes el rendimiento y la escalabilidad. El rendimiento se tiene que tener en cuenta desde el principio, influye mucho en un buen diseño de la BD. El rendimiento es una medida del número de transacciones simultáneas que puede controlar la base de datos correctamente. Para que una BD sea escalable, debe admitir el rendimiento requerido para el número previsto de usuarios simultáneos.

Ambas propiedades pueden mejorar de muchas maneras, según las características del sistema, y después de realizar un profundo estudio de muchos puntos tanto del diseño de la BD, como de *software* y *hardware.* Por lo que los sistemas para mejorar la escalabilidad pueden crecer de dos formas fundamentales:

- **Escalado vertical:** se logra agregando más *hardware* a un único nodo o actualizando a un nodo más grande.
- **Escalado horizontal:** se logra agregando más nodos y distribuyendo los datos para repartir la carga de trabajo entre ellos.

El rendimiento y la escalabilidad están entre las principales preocupaciones de los diseñadores de sistemas que manejan gran volumen de información. A la par, que las empresas crecen y concentran más información se hace necesario buscar una solución para los altos niveles de transacción diarias. Con cada aumento se presentan nuevos retos de rendimiento y escalabilidad.

1.3.2 Tendencias actuales.

Las BD y su tecnología han tenido y están teniendo un impacto considerable y decisivo en el mundo de la informática, siendo una necesidad manejar grandes volúmenes de información de forma rápida, ágil y segura.

Los datos encaminados cada vez más al enfoque comercial y de reglas de negocios, están cambiando su forma de almacenamiento y el uso que se hace de ellos, siendo de especial interés la extracción de información implícita en la BD, surgiendo la necesidad de plantear nuevos modelos y sistemas de BD que aporten un valor añadido a las BD relacionales.

Esto abre nuevos campos dentro de la tecnología de BD, como son los Almacenes de Datos (*Data Warehouse*), Exploración de los Datos (*Data Mining*), y OLAP (*Online Analytical Processing*) para la recuperación de grandes volúmenes de información.

La información está conformada por los datos y los significados de los mismos, el dato por sí solo no es información y sobre esta base trabajan los Almacenes de Datos (AD) que constituyen la base del proceso analítico (OLAP), para brindar la información necesaria en el momento pedido. A diferencia de las BD, los AD, contienen datos consolidados, rico contenido histórico, estructura de datos comprensibles y son eficientes en las consultas. Los datos contenidos en un AD se encuentran organizados para permitir el análisis más que para procesar transacciones en tiempo real como ocurre en los sistemas de proceso de transacciones en línea (OLTP, *Online Transaction Processing*).

1.4 Análisis de las herramientas de desarrollo a utilizar.

1.4.1 Sistemas Gestores de BD (SGBD).

Los Sistemas Gestores de Bases de Datos (SGBD) son una parte importante dentro del mundo de la información, contienen todas las rutinas necesarias para el manejo de los datos; se puede definir un SGBD como: conjunto de herramientas que suministra a todos (administrador, analistas, programadores, usuarios) los medios necesarios para describir, recuperar y manipular los datos almacenados en la BD, manteniendo la seguridad, integridad y confidencialidad de los mismos.

Existen actualmente numerosos SGBD, como por ejemplo: Access, Oracle, SQL, PostgreSQL, MySQL, etc. Cada sistema presenta características propias, la elección de uno u otro sistema para gestionar los datos vendrá definida por las necesidades.

Objetivos de los SGBD.

Los SGBD permiten un control total de la información, los principales objetivos son:

- Evitar la redundancia de los datos, eliminando así la inconsistencia de los mismos.

- Mejorar los mecanismos de seguridad de los datos y la privacidad.
- Asegurar la independencia de los programas y los datos, es decir, la posibilidad de modificar la estructura de la base de datos (esquema) sin necesidad de modificar los programas de las aplicaciones que manejan esos datos.
- Mantener la integridad de los datos realizando las validaciones necesarias cuando se realicen modificaciones en la base de datos.
- Mejorar la eficacia de acceso a los datos, en especial en el caso de consultas imprevistas.

Funciones de los SGBD.

Las principales funciones que debe realizar un SGBD son:

- La definición de los datos.
- La manipulación de los datos.
- Garantizar la seguridad e integridad de los datos.
- La gestión de las transacciones y el acceso concurrente.

Actualmente existen SGBD líderes del mercado mundial, atendiendo a su clasificación entre *software* libre y *software* propietario los más usados son: SQL Server, Oracle, My SQL y PostgreSQL, se analizan algunas características de ellos.

SQL Server

Microsoft SQL Server 2000 es uno de los mejores SGBD para Windows, es el RDBMS (Relational Data Base Management System) de elección para una amplia gama de clientes corporativos y proveedores independientes de *software* (ISVs) que construyen aplicaciones de negocios. Las necesidades y requerimientos de los clientes han llevado a la creación de innovaciones de producto significativas para facilitar la utilización, escalabilidad, confiabilidad y almacenamiento de datos.

Ventajas:

- Soporta la configuración automática y la auto-optimización.
- Administración multiservidor para un gran número de servidores.
- Gran variedad de opciones de duplicación de cualquier base de datos.
- Acceso universal a los datos (Universal Data Access).
- Fácil de usar.
- Escalabilidad: Se adapta a las necesidades de la empresa, soportando desde unos pocos usuarios a varios miles.
- Potencia: Microsoft SQL Server es la mejor BD para Windows NT Server.
- Posee los mejores registros de los benchmarks independientes (TCP) tanto en transacciones totales como en coste por transacción.
- Gestión: Con una completa interfaz gráfica que reduce la complejidad innecesaria de las tareas de administración y gestión de la BD.

Inconvenientes:

- Licencias con costos altos.
- Plataformas Windows.[4]

MySQL

Es un SGBD basado en Open Source (código abierto) diseñado para los sistemas Unix, aunque existen versiones para Windows.

Ventajas:

- Diseñado en vistas a la velocidad.
- Consume muy pocos recursos de CPU y memoria. Muy buen rendimiento.
- Tamaño del registro sin límite.
- Buena integración con PHP.
- Utilidades de administración (phpMyAdmin).

- Buen control de acceso usuarios-tablas-permisos.

Inconvenientes:

- No soporta subconsultas.
- No soporta transacciones.
- No soporta triggers ni procedimientos en el servidor.
- No soporta claves ajenas. Ignora la integridad referencial.
- No soporta vistas.
- Se hace inestable cuando contiene gran cantidad de datos.

Oracle

Oracle es un SGBD fabricado por Oracle Corporation.Se considera a Oracle como uno de los sistemas de BD más completos.

Ventajas:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

Inconvenientes:

- Su enorme precio, que es de varios miles de euros (según versiones y licencias).
- La seguridad de la plataforma y las políticas de suministro de parches de seguridad, modificadas a comienzos del año 2005 y que incrementan el nivel de exposición de los usuarios ha sido criticado por algunos especialistas. [5]

PostgreSQL

Hoy en día existen muchas empresas y sitios Web que necesitan mantener de forma eficiente un gran volumen de datos. Muchos de ellos optan por soluciones comerciales, aunque muchas otras confían en el *software* libre optando por una solución como PostgreSQL o MySQL.

Dentro de los gestores de bases de datos, como uno de los más representativos, a PostgreSQL, un motor de base de datos que es servidor de base de datos relacional libre, liberado bajo la licencia BSD. Ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: Clases, Herencia, Tipos, Funciones.

Otras características aportan potencia y flexibilidad adicional:

- Restricciones (Constraints)
- Disparadores (triggers)
- Reglas (rules)
- Integridad transaccional

Las principales mejoras en PostgreSQL incluyen:

- Los bloqueos de tabla han sido sustituidos por el control de concurrencia multi-versión, el cual permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros, y permite copias de seguridad en caliente desde pg_dump mientras la base de datos permanece disponible para consultas.
- Se han implementado importantes características del motor de datos, incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (*constraints)* y disparadores (*triggers*).
- Se han añadido funcionalidades en línea con el estándar SQL92, incluyendo claves primarias, identificadores entrecomillados, forzado de tipos cadena literal, conversión de tipos y entrada de enteros binarios y hexadecimales.
- Los tipos internos han sido mejorados, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.
- Establecer condiciones o CHECKs para validar las entradas de datos

- Permitir transacciones, es decir, múltiples operaciones de tabla o registros de manera segura.
- Bloqueos de registros, útil en entornos hoy por hoy multiusuario.
- Administración de Grupos de usuarios, y soporte nativo SSL.
- Múltiples lenguajes para procedimientos almacenados (incluyendo el nativo PL/PgSQL, PL/PHP, PL/Perl y PL/Python)

- La velocidad del código del motor de datos ha sido incrementada aproximadamente en un 20-40%, y su tiempo de arranque ha bajado el 80% desde que la versión 6.0 fue lanzada.[6]

1.4.2 Herramientas CASE.

Las herramientas CASE (***C**omputer **A**ided **S**oftware **E**ngineering*, Ingeniería de *Software* Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y de dinero.

Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Aunque no es fácil y no existe una forma única de clasificarlas, las herramientas CASE se pueden clasificar en base a los parámetros siguientes:

- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.
- Su funcionalidad.

La siguiente clasificación es la más habitual basada en las fases del ciclo de desarrollo que cubren:

- *Upper CASE*, herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando, entre otros diagramas el Lenguaje Unificado de Modelado (UML).

- *Middle CASE*, herramientas para automatizar tareas en el análisis y diseño de la aplicación.
- *Lower CASE*, herramientas que semiautomatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas. Además automatizan la documentación completa de la aplicación. [7]

Por funcionalidad se pueden diferenciar algunas como:

- Herramientas de generación semiautomática de código.
- Editores UML.
- Herramientas de refactorización de código.
- Herramientas de mantenimiento como los sistemas de control de versiones.

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 aplicaciones de bases de datos grandes y complejos.

ER/Studio está equipado para crear y manejar diseños de BD 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:

- Capacidad fuerte en el diseño lógico.
- Sincronización bidireccional de los diseños lógico y físico.
- Construcción automática de BD.
- Reingeniería inversa de BD.
- Documentación basada en HTML.
- Un Repositorio para el modelado.

Las capacidades de diseño que contiene, ayudan a crear un diseño lógico que puede transformarse en cualquier número de diseños físicos. Como resultado, se puede mantener un diseño lógico normalizado mientras se des normalizan los diseños físicos para su desempeño.

ER/Studio mantiene ligaduras entre todos los niveles de su diseño por lo tanto puede mezclar cambios en cualquier dirección entre ellos. ER/Studio revisa la normalización y la compilación con la sintaxis de la plataforma de la base de datos.

Se pueden desplegar los modelos de datos usando la notación IDEF1X o IE. ER/Studio permite tomar por omisión las opciones para todos los diagramas así como realizar cambios al momento de la ejecución.

Genera otros objetos de BD: vistas, procedimientos almacenados, defaults, reglas, y tipos de datos de usuario, lo cual ayuda a la auto ordenación de tipos de objetos para eliminar errores de dependencia al construir la base de datos. Tiene una opción para generar código fuente o para construir BD. Soporte para crear BD para Servidores SQL; y otra, para incluir código SQL y verificar la creación de objetos. Además de la opción para incluir encabezados de comentarios.

Una vez que se ha diseñado la BD, se pude construir o generar código fuente para todo o para parte de los diseños de la BD. Propiamente hace la secuencia de la creación de tipos de objetos diferentes para asegurar eficiencia, y construir bases de datos libres de errores.

Actualiza una BD del diagrama. ER/Studio permite aplicar cambios de diseño del modelo de datos directamente a la BD. Cuando se comparan las diferencias entre los dos, formula una estrategia de alteración inteligente que implementa el diseño de las modificaciones mientras se preserva la tabla con los datos existentes, privilegios de objetos, y dependencias en la BD. [8]

Toad

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

Con él, puedes desarrollar diagramas para la mayor parte de SGBD 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 BD o detallados informes en HTML y RTF.

Además, posee una herramienta denominada 'Model Explorer' que permite navegar por todos los atributos del modelo que se está creando. Permite desarrollar aplicaciones con mayor rapidez y facilidad, simplificando las tareas de administración de bases de datos.

Un navegador interno te permite además visualizar tu BD y gestionar sus objetos con facilidad. El programa es capaz además de conectarse a varias BD simultáneamente. [9]

Rational Rose

Este paquete permite documentar un producto *software* brindando una serie de facilidades para la elaboración de los artefactos que propone el *Rational Unified Process* (RUP). Está compuesto a su vez por un grupo de herramientas que se especializan en un aspecto en específico:

- **Rational Rose Enterprise Edition:** Esta herramienta permite modelar visualmente un grupo de artefactos que se generan como parte de la metodología. Dividido en cuatro vistas fundamentales: Vista de Casos de Uso, Vista Lógica, Vista de Componentes y Vista de Despliegue; propone un grupo de artefactos predefinidos que facilitan el desarrollo de *software* con calidad.
- **Rational Requisite Pro:** Esta herramienta gestiona los Requerimientos Funcionales y No funcionales del sistema a elaborar.
- **Rational Clear Case:** Rational ClearCase proporciona una gestión del ciclo de vida y control de los activos de desarrollo de *software*.
- **Rational ClearQuest:** proporciona un seguimiento flexible de defectos y cambios en toda la empresa. Soporte completo para consultas con generación de múltiples informes y gráficos.

- **Rational Soda:** Automatiza la documentación del proyecto de *software* a lo largo de todo el ciclo de vida. Genera documentos mediante la extracción de datos solicitados directamente de los repositorios de datos de herramientas.

Existen dentro del paquete otras herramientas con otras funcionalidades que gestionan y satisfacen todo el proceso de desarrollo de *software*. [10]

CASE Studio

CASE Studio es una herramienta profesional con la que se puede diseñar bases de datos, 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 BD (Oracle, SQL, MySQL, MS SQL, MaxDB, PostgreSQL, Access, etc.), permite además generar scripts SQL, usar plantillas de diseño personalizables y crear detallados informes en HTML y RTF.

A través de los diagramas de relación que se realizan en el Case Studio se tiene una visión más clara del contenido y estructura de tu base de datos, facilitando la gestión y mantenimiento de la misma.

Consta de un su potente sistema de ingeniería inversa, que permite identificar y estructurar bases de datos ya existentes para poder trabajar con ellas sin problemas. [11]

1.4.3 Herramienta para la administración y desarrollo de PostgreSQL Database Server.

EMS SQL Manager for PostgreSQL

Es una aplicación de alto desempeño para la administración y desarrollo de PostgreSQL Database Server. El programa trabaja con cualquier versión de PostgreSQL y soporta todas las últimas características de PostgreSQL, incluyendo espacios de tablas, nombres de argumentos en funciones y más. Su interfaz gráfica es sumamente atractiva e incluye un modo guiado de trabajo.

Ofrece un variado grupo de herramientas de gran alcance para los usuarios experimentados tales como diseñador visual de la base de datos, constructor visual de la pregunta, modos y realizar cualquier otra

operación que usted pueda necesitar en su trabajo con preguntas de la base de datos. Tiene sistema requerido mínimo de los instrumentos para esos usuarios que sean nuevos en el servidor de PostgreSQL y necesiten solamente su funcionalidad básica.

Características:

- Soporte completo para PostgreSQL hasta la versión 8.2
- Administración y navegación rápida de bases de datos.
- Administración fácil de todos los objetos PostgreSQL.
- Administración efectiva de seguridad.
- Capacidades de exportación e importación de datos.
- Modo guiado para labores de mantenimiento.
- Interfaz atractiva. [12]

1.5 Sistemas automatizados.

En la esfera de la salud, actualmente, existen diferentes producciones de *software* que tienen como objetivo resolver los problemas en este sector. Internacionalmente las aplicaciones dirigidas a brindar este tipo de servicios en los centros hospitalarios están bien fortalecidas siendo de mayor demanda en los países capitalistas.

La puesta en marcha de los servicios informáticos en los hospitales o centros de salud han ayudado en gran medida al establecimiento de sistemas de información que testifican la recogida rápida e inequívoca de todos los datos que durante la historia del centro cuantifican la actividad realizada en el mismo, de igual modo proporcionan el registro constante y la posterior recuperación con fines de diversos datos sobre las características de los procesos atendidos en ellos.

La posibilidad de tener tecnologías con una capacidad de almacenamiento, tratamiento y recuperación de la información creciente, ha sosegado las bases para anexar a los sistemas informativos hospitalarios datos de tipo clínico, hasta hace muy poco casi siempre retirados de los mismos.

Es bueno destacar algunos *software* que se han desarrollado a nivel internacional y que se comercializan en la actualidad:

eMedicalRecs.com

Es un *software* en el que se dispone de un sistema de gestión de información que puede ser utilizado en consultorios o clínicas médicas donde podrá almacenar, consultar y actualizar las historias clínicas de sus pacientes. De esta manera se mantiene actualizada la base de datos desde internet. [13]

MedFile 5.x (*Software* de Historias Clínicas Electrónicas)

Es un *software* de fácil uso, efectivo y con un precio accesible, diseñado para satisfacer las necesidades de archivo de HC (Expedientes) y manejo de Turnos de un Consultorio o Institución médica en el que se desempeñen uno o varios profesionales (hasta 200 en la versión multiusuario).

MedFile 5.x permite crear y mantener HC Electrónicas de sus pacientes en un formato especial de base de datos, asignar Turnos para la consulta con agenda personalizada para cada médico, y emitir Prescripciones y Órdenes Médicas en forma altamente personalizable y configurable.

Cuenta con un Módulo de Imágenes médicas que incorpora herramientas gráficas adecuadas para visualizar, exportar, imprimir y copiar imágenes médicas de cualquier tipo (endoscopia, radiología, tomografía, resonancia magnética, ultrasonido), o fotografías digitales, que se archivan junto con la HC.

Tanto las fichas clínicas como las imágenes, prescripciones y listado de turnos pueden ser impresos. MedFile 5.x cuenta además con la capacidad única de configurar Fichas Clínicas Personalizadas para la mayoría de las especialidades médicas, y la posibilidad de adjuntar diagnósticos según las clasificaciones CIE-9CM y CIE-10 de la OMS, en español.

La asignación de Turnos es también configurable y altamente flexible, permitiendo establecer días de atención, franjas horarias, duración de la consulta, feriados y días no laborables y asignar "sobreturnos" para cada uno de los usuarios.

MedFile 5.x ha sido diseñado en concordancia con las principales especificaciones de seguridad y privacidad reguladas por HIPAA (Acta de Seguridad y Portabilidad de la Seguridad Social de los EEUU):

- Implementación de algoritmos de encriptación para que los datos de sus pacientes no puedan ser revelados a terceros.
- Sistema de contraseñas de ingreso.
- Registro de actividades (Logfile) para evitar intrusiones no autorizadas.
- Desconexión automática de la sesión por inactividad, con tiempo programable, para evitar que MedFile quede abierto y sea utilizado por personas no autorizadas.
- Administración de hasta 200 usuarios con permisos de lectura y escritura seleccionados por el Administrador, para mantener la privacidad de sus pacientes.
- Inicio de sesión de cada usuario con nombre y contraseña.
- Nueva estructura de base de datos, con encriptación de 448 bits para mantener la privacidad de los pacientes, en concordancia con requerimientos de la HIPAA.
- Copia de seguridad de la base de datos e imágenes de los pacientes. (Backup / Restore).
- Exportación de base de datos demográficos a planilla tipo Excel, programable, que permite manejar los datos de pacientes fuera del entorno MedFile y por ejemplo, imprimir mailings, etc.
- Funcionamiento en red.
- Opciones mono usuario o multiusuario.

Actualización por Internet, automática o accionada por el usuario. [14]

Gowin HC - Historia clínica electrónica

Gowin Mw es un sistema de la información que ha sido desarrollado para los SIH, tiene el objetivo de recopilar toda la información referente a la historia clínica de los pacientes que pertenecen a la organización. Consta de una base de datos post-relacional que proporciona, como característica fuera de lo común, tres opciones integradas de acceso a los datos: una robusta base de datos de objetos, SQL de alto rendimiento y un acceso directo al motor multidimensional. Permite el desarrollo rápido de aplicaciones Web, proporciona una extraordinaria velocidad de proceso de transacciones, escalabilidad masiva y consultas en tiempo real sobre datos transaccionales. [15]

BIOCOM

Brinda el servicio de digitalización de historias clínicas y toda la documentación administrativa haciendo más ágil y segura la gestión de la documentación. Ha enfocado sus esfuerzos y estudios a la consultoría documental y el reto de mejorar la gestión de las instituciones de salud.

Se integra en un sistema ágil el almacenamiento de los datos clínicos del paciente con la reserva de Turnos y otras utilidades del consultorio. Está disponible en un sistema que integra la programación visual con una base de datos Access de fácil instalación. [16]

En Cuba la necesidad de integración en el sistema sanitario viene reflejado claramente en la afirmación de la política cubana: "los retos de los sistemas de salud –ofrecer servicios de calidad y accesibles- sólo podrán ser abordados si se implantan y generalizan sistemas de salud modernizados, interoperables y plenamente integrados".

Esto significa que será preciso desarrollar servicios seguros para interconectar hospitales, laboratorios, farmacias, centros de atención primaria y residencias para la tercera edad y por otra parte está facilitando a los gestores una herramienta que les ayuda a evaluar, comprender y analizar los recursos disponibles y a las tomas de decisiones.

Además se ha mantenido el desarrollo de este tipo de sistemas controlados por el Centro de Desarrollo Informático para la Salud Publica (CEDISAP), contando con la ayuda de la UCI, donde en estos momentos se encuentra un SIH, el cual esta en prácticas en hospitales del país, con el objetivo de probar su funcionamiento.

Pero este sistema no es lo más factible posible debido a que está desarrollado bajo una plataforma que no es la más correcta, utilizando *software* propietario como herramientas de desarrollo, lo que trae como consecuencia que se presenten innumerables limitaciones:

SIH

- No posee arquitectura orientada a servicios lo que lo convierte en un sistema creado en una isla de información con utilidad y beneficios muy limitados, lo que imposibilita que el sistema se integre por si solo a la RED TELEMATICA DE INFOMED.
- Escasa o ninguna documentación el proceso de análisis, diseño e implementación del sistema, lo que hace extremadamente difícil el mantenimiento.
- No posee buena escalabilidad, ya que al aumentar el tamaño de la base de de datos, la aplicación no puede adaptarse a los volúmenes de datos generados por las consultas y el trabajo se dificulta, perdiendo de esta forma la rapidez en las prestaciones de servicios.
- Está desarrollado sobre una plataforma propietaria.
- La interfaz del sistema no es amigable.
- Los sistemas de búsqueda son estáticos.
- No es un sistema multiplataforma.

Conclusiones:

Al observar todas las posibles opciones de BD, fue necesario hacer un estudio de que utilizar para lograr vencer los objetivos de manera más óptima.

Como SGBD se utiliza PostgreSQL, para mantener la política que sigue la UCI, además por tener características como ser flexible y estable aún con una gran cantidad de datos contenidos en el y varias personas conectadas a la vez. También, se debe tener en cuenta que sería útil lograr una cierta comunicación entre este sistema y el SIH.

Para el diseño final de la BD se utiliza el *Rational Rose,* como herramienta para obtener el diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño y el CASE Studio como herramienta de diseño de esquema de datos, ya que permite desarrollar diagramas para la mayor parte de sistemas gestores de bases de datos existentes.

Además como herramienta para la administración y desarrollo de PostgreSQL Database Server se utiliza *EMS SQL Manager for PostgreSQL* ya que posee una variedad de características que permiten su uso y un soporte completo para PostgreSQL hasta la versión 8.2.

CAPITULO 2. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

En este capitulo se hará una selección y argumentación de los requisitos funcionales y no funcionales del sistema propuesto. Además se describe y fundamenta la arquitectura utilizada para la Base de Datos. Se detalla el diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño.

2.1 Selección y argumentación de los requisitos funcionales y no funcionales del sistema propuesto.

El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una descripción de los requisitos del sistema (es decir, las condiciones o capacidades que el sistema debe cumplir) suficientemente buena como para que pueda llegarse a un acuerdo entre el cliente (incluyendo los usuarios) y los desarrolladores sobre qué debe y que no debe hacer el sistema.

2.1.1 Requerimientos funcionales.

RF1- Autentificar usuario: El sistema le pide al usuario: (Usuario y Contraseña de dominio*),* y por aquí se conoce a que grupo de usuarios pertenece y a que lugares del sistema puede acceder y con que permisos.

RF 1.1 Comparar Usuario y contraseña con los usuarios del sistema.

RF 1.2 Asignar privilegios.

RF 2- Gestionar HC: El personal de archivos accede a las interfaces que le permita clasificar las HC, eliminarla, registrar su entrada-salida del archivo, darle una ubicación dentro del archivo, unirlas en caso de varias HC o realizar alguna búsqueda, el sistema de acuerdo a la opción que marque el personal de archivos le muestra las interfaces que van a llevar a cabo la realización de esa función.

RF 2.1 Clasificar HC.

RF 2.2 Asignar ubicación a la HC.

RF 2.3 Unir HC.

RF 2.4 Registrar entrada-salida de La HC.

RF 2.5 Insertar nueva HC.

RF 2.6 Eliminar HC.

RF 3- Buscar HC: Busca una HC por distintos campos como por ejemplo por datos personales del paciente, por estado de la HC, por departamentos que han pedido esta HC que se desea, por una fecha determinada donde el sistema de acuerdo a la opción que marque el usuario muestra las HC siempre verificando la veracidad de los datos de la búsqueda

RF 4- Crear Reporte: crea reportes estadísticos del movimiento de las HC en el archivo.

2.1.2 Requisitos no funcionales.

Los requisitos no funcionales se refieren a las propiedades o cualidades del sistema. Estos responden a cualidades que el producto debe tener y las características para que este sea atractivo, confiable, usable y seguro.

RNF 1- Requerimientos de rendimiento

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

RNF 2- Requerimientos de soporte

Se le debe dar mantenimiento periódicamente a los servidores de bases de datos controlando la integridad de la información. Además se harán salvas de los datos cada cierto tiempo.

RNF 3-Requerimientos políticos, culturales y legales.

Las herramientas propuestas para la Base de Datos deberán responder a los intereses de la Constitución de la República de Cuba, asimismo no existirán prioridades en el servicio según el nivel social, cultural o étnico. No se permitirán la divulgación de los datos. Todos los procesos responderán a las resoluciones establecidas por el Ministerio de Salud Pública cumpliendo las normas instituidas en el Código Penal.

RNF 4-Requerimientos de confiabilidad

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

RNF 6-Requerimientos de hardware.

Requerimientos para un servidor: 512Mb RAM (Recomendado 1Gb RAM o superior), 1GHz o superior, 60Gb HDD.

RNF 7-Requerimientos de software.

La Base de Datos debe corre en sistemas operativos Windows 98 o superior y sistemas Unix, Linux. Para sistemas Windows se deber tener instalado Microsoft Framework 2.0 y en sistemas Linux la plataforma Mono 1.2 o superior.

RNF 8-Restricciones en el diseño y la implementación

Se utilizará Case Studio para el diseño y modelado de la Base de Datos.

La Base de Datos será implementada utilizando como Sistema Gestor de Base de Datos PostgreSQL 8.2.

La comunicación de de las terminales clientes con el servidor será a través de conexiones de fibra óptica.

RNF 9-Requisitos de Independencia:

Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Hay que obtener un buen diseño de la base de datos para lograr evitar la aparición de información repetida o redundante por lo que seria muy factible lograr una redundancia nula.

RNF 10-Requisitos de Consistencia:

Consistencia: En aquellos casos en los que no se ha logrado esta redundancia nula, es necesario vigilar y validar por códigos de programación que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

RNF 11-Requisitos de Integridad.

Integridad. Se deben adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, hay que proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

RNF 13-Requisitos de Tiempo de respuesta.

Tiempo de respuesta. Se debe lograr el menor tiempo de respuesta posible minimizando el tiempo que el SGBD tardará en dar la información solicitada y en almacenar los cambios realizados, porque así se evita la ocupación de recursos del Servidor de BD y los mismos tiempos de respuestas de los usuarios.

2.2 Descripción de la arquitectura y su fundamentación.

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

- La arquitectura centralizada es la más clásica. En ella, el SGBD está implantado en una sola plataforma u ordenador desde donde se gestiona directamente, de modo centralizado, la totalidad de los recursos. Es la arquitectura de los centros de proceso de datos tradicionales. Se basa en tecnologías sencillas, muy experimentadas y de gran robustez.
- La conexión en red de varias computadoras permite que algunas tareas se ejecuten en un sistema servidor y que otras se ejecuten en los sistemas clientes. Esta división de trabajo ha conducido al desarrollo de sistemas de bases de datos cliente-servidor.
- La distribución de datos a través de las distintas sedes o departamentos de una organización permite que estos datos residan donde han sido generados o donde son más necesarios, pero continuar siendo accesibles desde otros lugares o departamentos diferentes. El hecho de guardar varias copias de la base de datos en diferentes sitios permite que puedan continuar las operaciones sobre la base de datos aunque algún sitio se vea afectado por algún desastre natural, como una inundación, un incendio o un terremoto. Se han desarrollado los sistemas de bases de datos distribuidos para manejar datos distribuidos geográfica o administrativamente a lo largo de múltiples sistemas de bases de datos.
- El procesamiento paralelo dentro de una computadora permite acelerar las actividades del sistema de base de datos, proporcionando a las transacciones unas respuestas más rápidas, así como la capacidad de ejecutar más transacciones por segundo. Las consultas pueden procesarse de manera que se explote el paralelismo ofrecido por el sistema informático subyacente. La necesidad del procesamiento paralelo de consultas ha conducido al desarrollo de los sistemas de bases de datos paralelos. No debe confundirse el SGBD con la arquitectura que se elige para implantarlo. Algunos SGBD sólo se pueden implantar en una de las arquitecturas y otros en todas ellas. [17]

Existen varias formas de diseñar una base de datos, ellas se modelan en dos tipos de categorías: distribuidas y centralizadas.

Los arquitectos del proyecto determinaron utilizar como una arquitectura centralizada en SIH. Posibilitando que desde un servidor central se procesen los datos de todos los módulos del proyecto, existiendo un administrador en este servidor central que tenga un control de todas las base de datos que comprende el proyecto, permitiendo acelerar las actividades del sistema de base de datos utilizado, proporcionando a

las transacciones unas respuestas más rápidas, así como la capacidad de ejecutar más transacciones por segundo. Las consultas pueden procesarse de manera que se explote el paralelismo ofrecido por el sistema informático subyacente.

Arquitectura centralizada

Los sistemas de bases de datos centralizados se ejecutan en un único sistema informático sin interaccionar con ninguna otra computadora. Tales sistemas 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. [18]

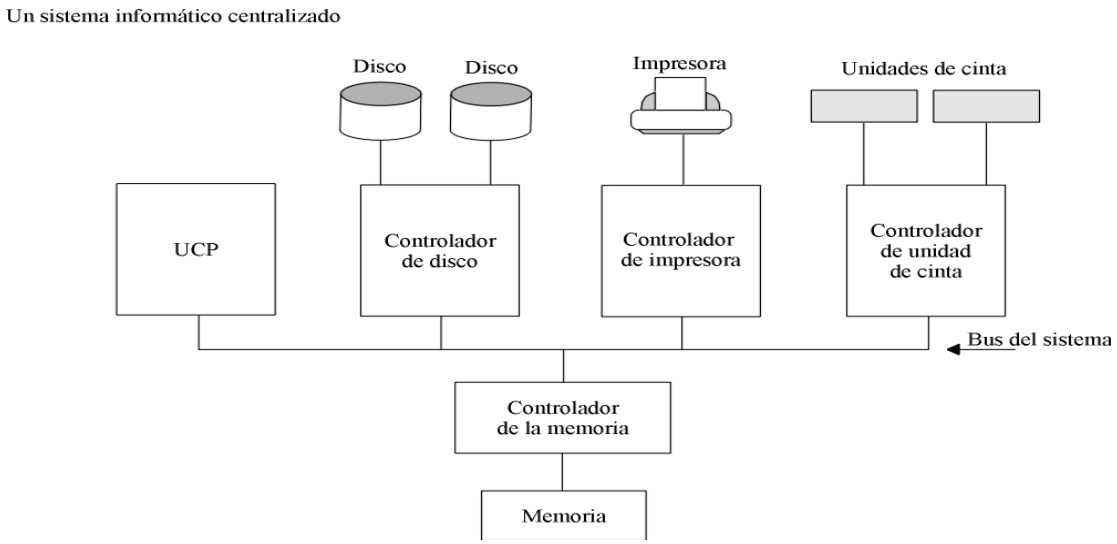


Fig.2: Esquema de un sistema informatico centralizado

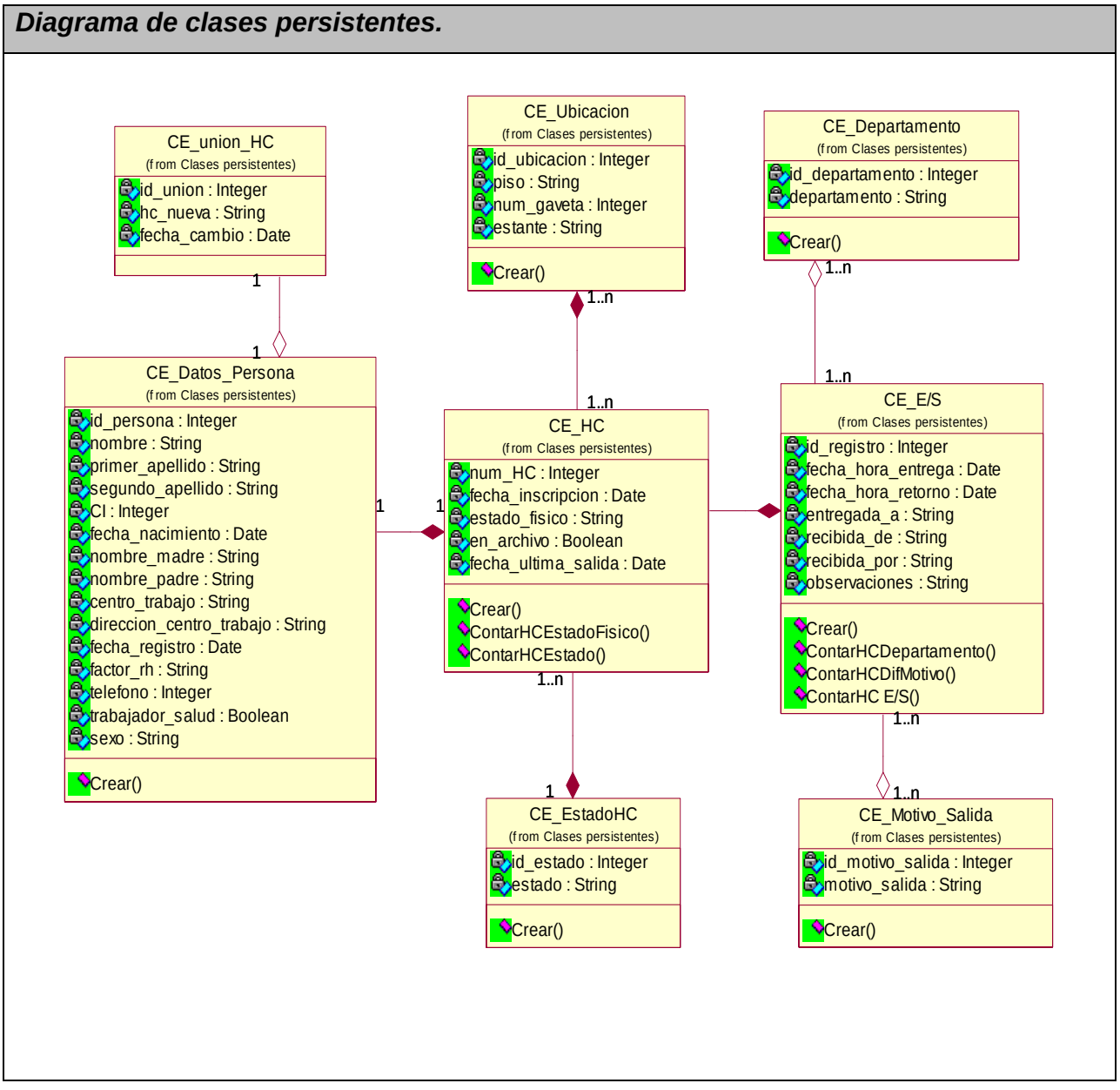
El proyecto SIH está compuesto por varios módulos que comprenden los procesos de gestión en los hospitales cubanos, cada uno con sus características específicas. El módulo Archivo de Historia Clínica funciona de manera independiente y no se relaciona con ninguno de del demás módulos existentes. Se realizan restricciones en cuanto al acceso a los datos que aquí se manejan, por lo que se aplica la restricción de usuarios del sistema.

Por tanto la arquitectura centralizada que se aplicará en el módulo quedaría de la siguiente manera:

Se tendrá un servidor centrar o computador que contenga todas las bases de datos de los módulos, incluyendo la de Archivo de Historias Clínicas. Existirá un administrador que es el encargado de gestionar toda la información que se manipule en este servidor y es el responsable de establecer las restricciones de seguridad, donde cada trabajador desde su puesto de trabajo accederá solo a la instancia del *software* que le corresponda.

2.3 Diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño.

Diagrama de clases persistentes.



2.4 Descripción de las clases.

Nombre: (1)	
Tipo de clase (interfaz, controladora, entidad)	
Atributo	**Tipo**
(2)	(3)
Para cada responsabilidad:	
Nombre:	(4)
Descripción:	(5)

Leyenda:

(1) Nombre de la clase.

(2) Nombre de cada uno de los atributos.

(3) Tipo de dato de cada uno de los atributos.

(4) Nombre de la responsabilidad (operación) con los parámetros que requiera.

(5) Breve explicación de en qué consiste la responsabilidad.

Nombre: CE_Datos_Persona	
Tipo de clase: Entidad.	
Atributo	**Tipo**
id_persona	Interger
nombre	String
primer_apellido	String
segundo_apellido	String
CI	String
fecha_nacimiento	Date

nombre_madre	String
nombre_padre	String
centro_trabajo	String
direccion_centro_trabajo	String
fecha_registro	Date
factor_rh	String
telefono	String
trabajador_salud	Boolean
sexo	String
Para cada responsabilidad:	
Nombre:	Crear()
Descripción:	Crea una persona en el sistema

Nombre: CE_union_HC	
Tipo de clase: Entidad.	
Atributo	**Tipo**
Id_union	integer
Hc_nueva	String
Fecha _ cambio	Date
Para cada responsabilidad:	
Nombre:	
Descripción:	

Nombre: CE_Ubicacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_ubicacion	integer
piso	string
Num_gaveta	integer
estante	string
Para cada responsabilidad:	
Nombre:	Crear()
Descripción:	Crea la ubicación de una HC.

Nombre: CE_Departamento	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id _ departamento	Integer
departamento	string
Para cada responsabilidad:	
Nombre:	Crear()
Descripción:	Crea los departamentos de un hospital.

Nombre: (CE_HC)	
Tipo de clase: Entidad	
Atributo	**Tipo**
num_HC	Integer
fecha _ inscripción	Date
estado_fisisco	String

en_archivo	Boolean
fecha_ultima_salida	Date
Para cada responsabilidad:	
Nombre:	Crear()
Descripción:	Crea una nueva HC en el archivo.
Nombre:	ContarHCEstadoFisico()
Descripción:	Cuenta las HC que se encuentran en el archivo.
Nombre:	ContarHCEstado()
Descripción:	Cuenta las HC según su estado.

Nombre: CE_EstadoHC	
Tipo de clase: Entidad	
Atributo	**Tipo**
id _ estado	Integer
estado	String
Para cada responsabilidad:	
Nombre:	Crear()
Descripción:	Crea los estados de las HC

Nombre: CE_E/S	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_registro	Integer
fecha_hora_entrega	Date
fecha_hora_retorno	Date
entregada_a	String

recibida_de	String
recibida_por	String
observaciones	String
Para cada responsabilidad:	
Nombre:	Crear()
Descripción:	Crea una entrada o una salida de las HC del archivo.
Nombre:	ContarHCDepartamento()
Descripción:	Cuenta las HC que salieron del archivo por departamentos.
Nombre:	ContarHCDifMotivo()
Descripción:	Cuenta las HC que salieron del archivo por motivo de salida
Nombre:	ContarHCE/S()
Descripción:	Cuenta las HC que se le dieron entrada y salida del archivo.

Nombre: CE_Motivo_Salida	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_motivo_salida	Integer
motivo _ salida	String
Para cada responsabilidad:	
Nombre:	Crear()
Descripción:	Crea los motivos de salida de las HC.

2.4 Diseño de la BD.

2.4.1 Diagrama Entidad Relación de la BD.

Diagrama Entidad Relación de la BD Archivo_HC.

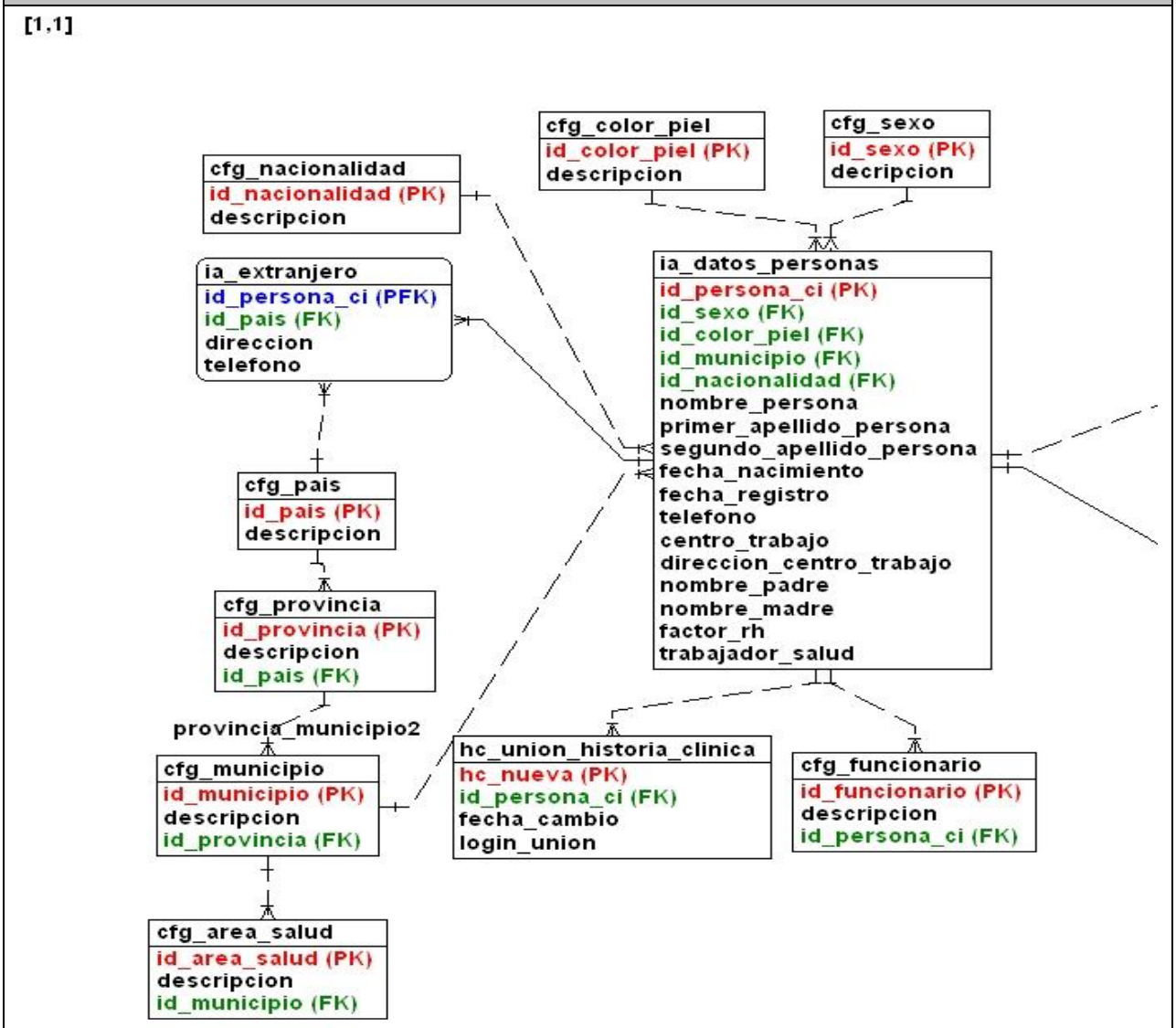


Diagrama Entidad Relación de la BD Archivo_HC.

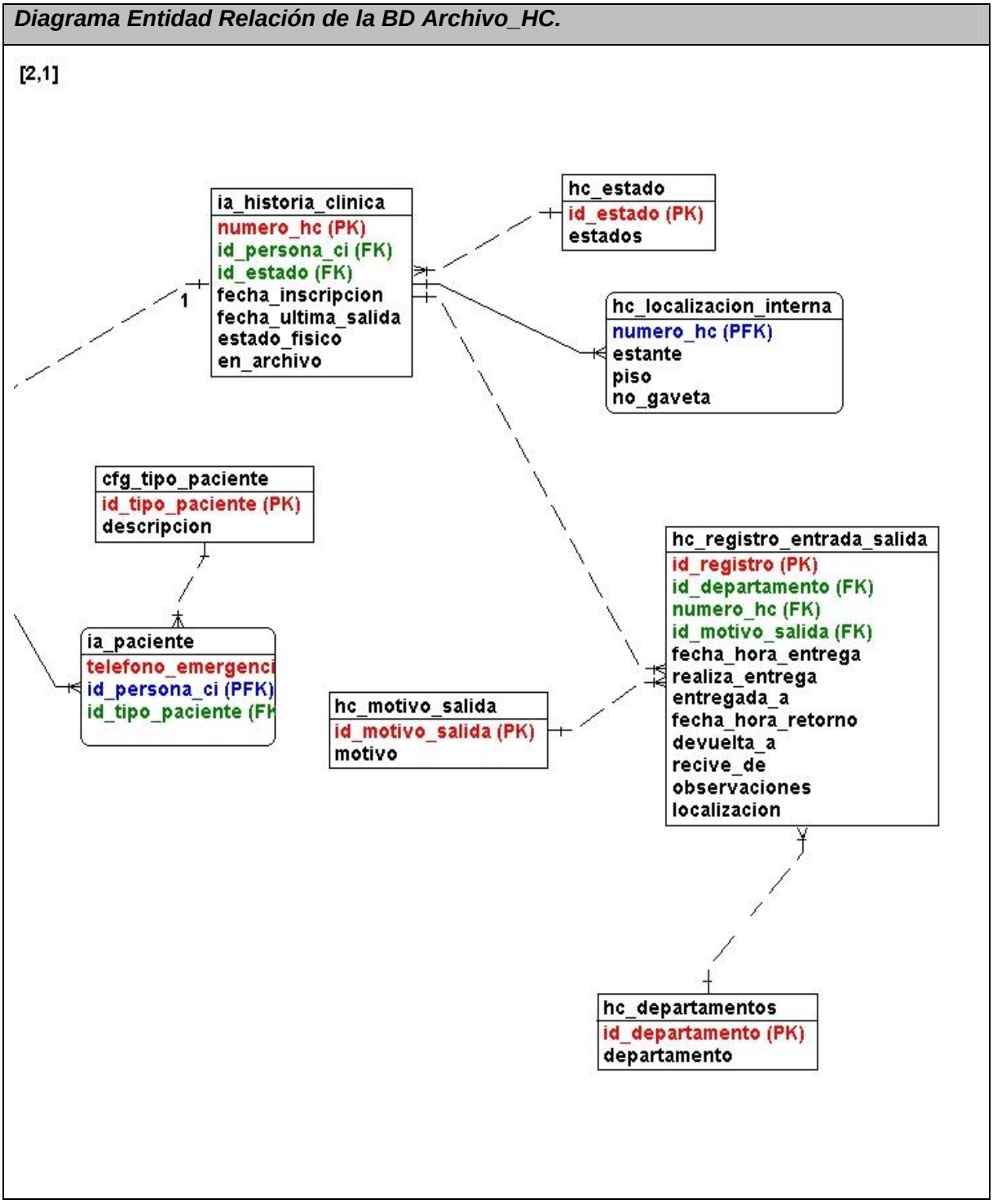
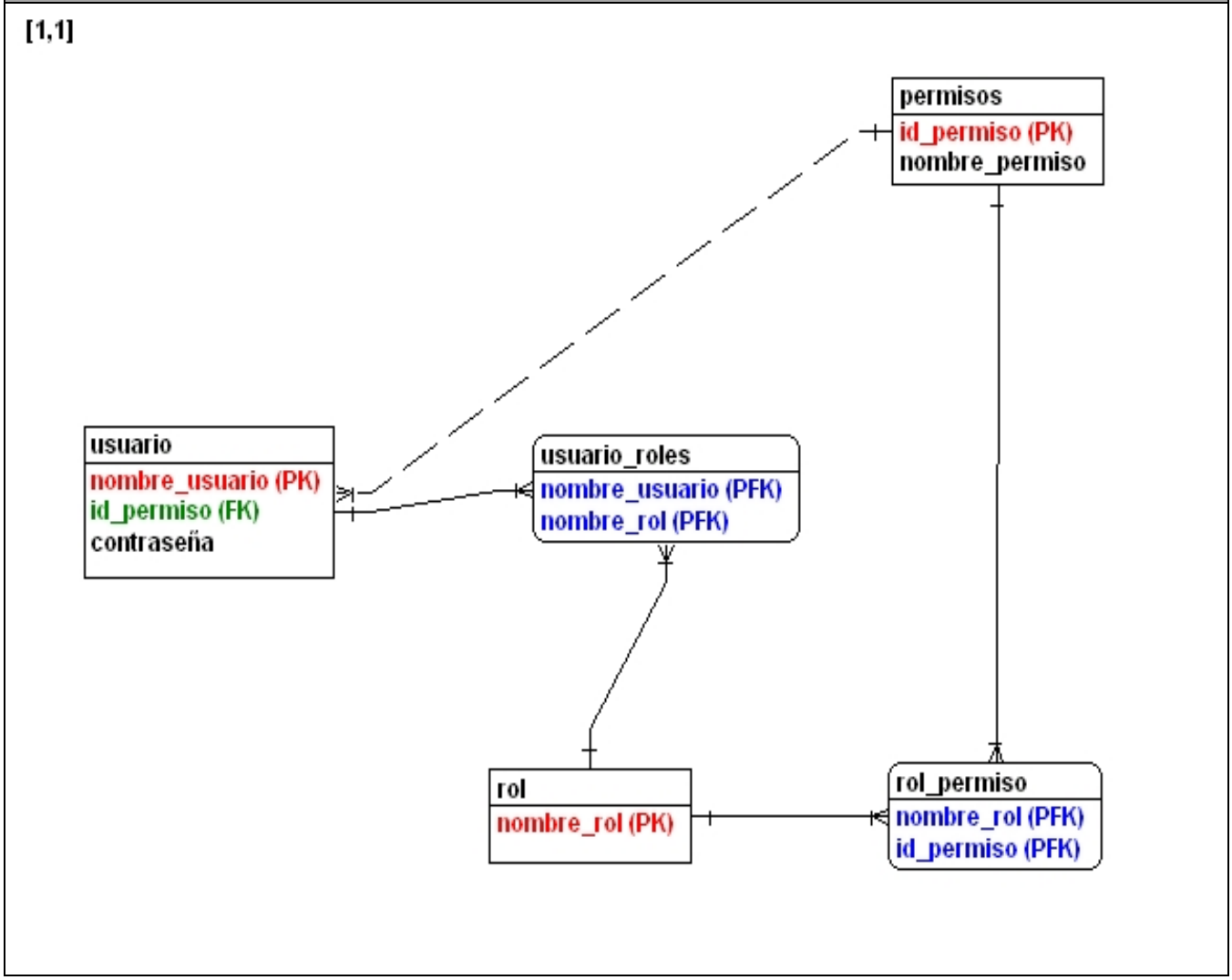


Diagrama Entidad Relación de la BD Seguridad Archivo_HC



2.4.2 Descripción de las tablas.

Nombre: 1		
Descripción: 2		
Atributo	**Tipo**	**Descripción**
3	**4**	**5**

Leyenda:

(1) Nombre de la tabla.

(2) Descripción general de qué datos almacena la tabla.

(3) Nombre de cada uno de los atributos.

(4) Tipo de dato de cada uno de los atributos.

(5) Breve explicación qué es ese atributo.

Nombre: cfg_nacionalidad		
Descripción: Esta tabla recoge los datos referentes a la nacionalidad.		
Atributo	**Tipo**	**Descripción**
id_nacionalidad	Integer	Este es el identificador de la nacionalidad.
descripción	Varchar	Este dato recoge el nombre de la nacionalidad.

Nombre: cfg_area_salud		
Descripción: Esta tabla recoge los datos referentes a las áreas de salud		
Atributo	**Tipo**	**Descripción**
id_area_salud	Integer	Este es el identificador del área de salud.
descripcion	Varchar	Este dato recoge el nombre del área de salud.
id_municipio	Interger	Llave foránea.

Nombre: cfg_funcionario		
Descripción: Esta tabla recoge los datos referentes a los funcionarios del hospital.		
Atributo	**Tipo**	**Descripción**
id_funcionario	Integer	Este es el identificador de funcionario.
descripcion	Varchar	Este dato recoge el nombre del tipo de funcionario, es decir si es medico, enfermero o trabajador de del hospital.
id_persona_ci	Varchar	Llave foránea.

Nombre: cfg_color_piel		
Descripción: Esta tabla recoge los datos referentes al color de la piel.		
Atributo	**Tipo**	**Descripción**
id_color_piel	Integer	Este es el identificador del color de la piel.
descripcion	Varchar	Este dato recoge el color de la piel.

Nombre: cfg_municipio		
Descripción: Esta tabla recoge los datos referentes a los municipios.		
Atributo	**Tipo**	**Descripción**
id_municipio	Integer	Este es el identificador del municipio.
descripcion	Varchar	Este dato recoge el nombre del municipio.
id_provincia	Interger	Llave foránea.

Nombre: cfg_provincia		
Descripción: Esta tabla recoge los datos referentes a las provincias.		
Atributo	**Tipo**	**Descripción**
id_provincia	Integer	Este es el identificador de la provincia.
descripcion	Varchar	Este dato recoge el nombre de la provincia.
id_pais	Interger	Llave foránea.

Nombre: cfg_sexo		
Descripción: Esta tabla recoge los datos referentes al sexo.		
Atributo	**Tipo**	**Descripción**
id_sexo	Serial	Este es el identificador de sexo.
descripcion	Varchar	Este dato recoge el nombre del sexo.

Nombre: cfg_tipo_paciente		
Descripción: Esta tabla recoge los datos referentes al paciente.		
Atributo	**Tipo**	**Descripción**
id_tipo_paciente	Integer	Este es el identificador del paciente.
descripcion	Varchar	Este dato recoge el nombre del paciente.

Nombre: hc_departamentos		
Descripción: Esta tabla recoge los datos referentes al departamento donde se archivan las historias clínicas.		
Atributo	**Tipo**	**Descripción**
id_departamento	Integer	Este es el identificador del departamento.
departamento	Varchar	Este dato recoge el nombre del departamento.

Nombre: hc_estado		
Descripción: Esta tabla recoge los datos referentes al estado de la historia clínica.		
Atributo	**Tipo**	**Descripción**
id_estado	Integer	Este es el identificador del estado.
estados	Varchar	Este dato recoge el nombre del estado.

Nombre: hc_localizacion_interna		
Descripción: Esta tabla recoge los datos referentes a la localización interna de la historia clínica.		
Atributo	**Tipo**	**Descripción**
numero_hc	Varchar	Llave foránea.
estante	Varchar	Este dato recoge el nombre del estante.

piso	Varchar	Este dato recoge el nombre del piso.
no_gaveta	Integer	Este dato recoge el número de la gaveta.

Nombre: hc_motivo_salida		
Descripción: Esta tabla recoge los datos referentes al motivo de salida de la historia clínica.		
Atributo	**Tipo**	**Descripción**
id_motivo_salida	Integer	Este es el identificador de cada motivo.
motivo	Varchar	Este dato recoge el nombre del motivo.

Nombre: hc_union_historia_clinica		
Descripción: Esta tabla recoge los datos referentes a las uniones de las historias clínicas.		
Atributo	**Tipo**	**Descripción**
hc_nueva	Varchar	Este es el identificador de la unión.
fecha_cambio	Timestamp	Este dato recoge la fecha de los cambios.
login_union	Varchar	Este dato recoge el nombre de la el login de la unión.
id_persona_ci	Varchar	Llave foránea

Nombre: hc_registro_entrada_salida		
Descripción: Esta tabla recoge los datos referentes a las entradas y salidas de las historias clínicas del archivo.		
Atributo	**Tipo**	**Descripción**
id_registro	Serial	Este es el identificador del registro.
fecha_hora_entrega	Timestamp	Este dato recoge la fecha de entrega.

realiza_entrega	Varchar	Este dato recoge el nombre del que realiza la entrega.
entregada_a	Varchar	Este dato recoge el nombre del que recibe.
fecha_hora_retorno	Timestamp	Este dato recoge la fecha del retorno al archivo.
devuelta_a	Varchar	Este dato recoge el nombre del que devuelve.
recive_de	Varchar	Este dato recoge el nombre del que recibe la devolución.
observaciones	Varchar	Este dato recoge las observaciones.
localización	Varchar	Este dato recoge el nombre de la localización.
id_departamento	Interger	Llave foránea
numero_hc	Interger	Llave foránea
id_motivo_salida	Interger	Llave foránea

Nombre: ia_datos_personas		
Descripción: Almacena los datos referentes a las personas.		
Atributo	**Tipo**	**Descripción**
id_persona_ci	Varchar	Este es el identificador de la persona.
nombre_persona	Varchar	Este dato recoge el nombre de la persona.
primer_apellido_persona	Varchar	Este dato recoge el primer apellido de la persona.
segundo_apellido_persona	Varchar	Este dato recoge el segundo apellido de la persona.
fecha_nacimiento	Time stamp	Este dato recoge la fecha de nacimiento.
fecha_registro	Time stamp	Este dato recoge la fecha en que fue registrado.
telefono	Varchar	Este dato recoge el teléfono.

centro_trabajo	Varchar	Este dato recoge el centro de trabajo.
direccion_centro_trabajo	Varchar	Este dato recoge la dirección del centro de trabajo.
nombre_padre	Varchar	Este dato recoge el nombre del padre.
nombre_madre	Varchar	Este dato recoge el nombre de la madre.
factor_rh	Boolean	Este dato recoge la presencia del factor rh.
trabajador_salud	Boolean	Este dato recoge la presencia del trabajador de la salud.
id_sexo	Integer	Llave foránea
id_color_piel	Integer	Llave foránea
id_municipio	Integer	Llave foránea
id_nacionalidad	Integer	Llave foránea

Nombre: usuario		
Descripción: Es una tabla que permite tener el control de los usuarios que trabajan con la aplicación.		
Atributo	**Tipo**	**Descripción**
nombre_usuario	Varchar	Es el identificador de la tabla y nombra al usuario en el dominio de la aplicación.
contraseña	Varchar	Permite una óptima identificación del usuario en la aplicación.

Nombre: permisos		
Descripción: Es una tabla que permite saber los permisos de que constan los usuarios		
Atributo	**Tipo**	**Descripción**
id_permiso	Integer	Es el identificador de la tabla.
nombre_permiso	Varchar	Nombra al permiso de acuerdo al identificador del mismo.

Nombre: rol		
Descripción: Es una tabla que permite saber de acuerdo al rol de cada usuario que permiso adquiere en la aplicación.		
Atributo	**Tipo**	**Descripción**
nombre_rol	Varchar	Es el identificador de la tabla y permite obtener el nombre específico de cada rol usado en la aplicación.

2.8 Análisis de optimización de consultas.

El rendimiento de una base de datos se mide, entre otros aspectos, por los tiempos de respuesta de sus transacciones y por el *throughput*, que es el número de transacciones por unidad de tiempo. Si estos tiempos están fuera del rango esperado es necesario llevar a cabo un proceso de entonación de la base de datos.

La entonación de una base de datos relacional como es el caso, se puede dividir en dos componentes fundamentales: la entonación del diseño físico y la entonación de las consultas.

Con respecto al diseño físico, se define la organización primaria de los archivos que van a contener las tablas de una base de datos tomando en cuenta el uso que se le va a dar, esto se refiere a cuáles transacciones están definidas sobre la base de datos y cuál es su frecuencia de ejecución. También se toman en consideración las restricciones de tiempo de respuesta de cada transacción.

Aquí en esta sección sólo se tratará el análisis de optimización de consultas.

Las actualizaciones por lo general no tienen en sí mismas, muchos aspectos que se puedan entonar, basta con conocer cuáles actualizaciones tienen lugar en una BD y con cuál frecuencia. Sin embargo, las consultas tienen más variedad de formas y espacio para mejorar el rendimiento.

Según Shasha, citado en Optimización de consultas sobre fuentes de datos con capacidades limitadas: visión general existen 8 tipos de consultas, que se explican a continuación:

1. Consulta puntual *("point query").* La condición de búsqueda es igualdad y la consulta devuelve máximo una tupla.

2. Consulta múltiple *("multipoint query").* La condición de búsqueda es igualdad y la consulta devuelve varias tuplas.

3. Consulta por rango *("range query").* El rango se especifica sobre los valores de un atributo definido en un dominio ordenado, la consulta devuelve todas las tuplas cuyo valor del atributo de la condición está dentro del rango especificado.

4. Consulta de prefijo *("prefix match query").* Se expresa sobre un atributo o secuencia de atributos, la condición de búsqueda especifica sólo el prefijo del valor buscado.

5. Consulta extrema *("extremal query").* Devuelve las tuplas cuyo valor en uno o más atributos es un mínimo o un máximo.

6. Consulta de ordenamiento ("*ordering query*"). Es aquella consulta que incluye una cláusula de ordenamiento de las tuplas resultantes por el orden del valor de un atributo en particular.

7. Consulta de agrupación *("grouping query").* Es aquella consulta cuyas tuplas resultantes se agrupan de acuerdo al valor de uno o más atributos, generalmente, a cada grupo se le aplica una función agregada.

8. Consulta con *Join ("join query").* Es aquella consulta que asocia dos o más relaciones a través de atributos comunes. La condición sobre la cual se asocian las tablas puede usar la igualdad o algún otro comparador. [19]

Antes de considerar si una consulta debe ser reescrita para mejorar su rendimiento es necesario analizar si la consulta se está ejecutando muy lentamente. Dos indicadores de ello son:

- La consulta hace demasiados accesos a disco, por ejemplo, siendo una consulta puntual recorre toda la tabla.
- Al revisar el plan de ejecución se observa que hay índices relevantes que no se utilizan.

Existen varios aspectos en los que se puede mejorar el rendimiento de una consulta por ejemplo:

1. La cláusula DISTINCT 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. Las subconsultas en muchos manejadores se ejecutan ineficientemente. Una razón para ello es que al estar anidada no se utiliza algún índice relevante, al eliminar la subconsulta es posible lograr que el manejador utilice el índice apropiado.

4. Por otro lado, las tablas temporales pueden tener un efecto positivo al reescribir consultas que contienen subconsultas correlacionadas complejas. Las subconsultas correlacionadas pueden ejecutarse ineficientemente, pero si se logra calcular primero las tuplas de la subconsulta y almacenarlas en una tabla temporal, se puede reescribir la consulta original, sin la subconsulta, accediendo a la tabla temporal.

5. Otro uso beneficioso de las tablas temporales es para evitar el uso de la cláusula ORDER BY que puede ser costosa.

6. Las condiciones de *join* se pueden evaluar más eficientemente contra un índice primario. Y en general, en términos de rendimiento es preferible evaluar una condición de igualdad numérica que una condición de igualdad sobre cadenas *(‘strings”)* de caracteres.

7. Es preferible no usar la cláusula HAVING si la condición deseada se puede expresar en la cláusula WHERE.

8. Es importante conocer las particularidades del manejador, esto se refiere a conocer cómo han sido implementadas las rutinas de procesamiento de las consultas. Por ejemplo, es importante saber, al existir una condición disjuntiva (con OR) en la consulta si se usan o no los índices existentes.

9. Otra particularidad importante del manejador que es bueno conocer es si el orden de las tablas en la cláusula FROM puede afectar la implementación del join utilizada, posiblemente esto es relevante para joins que involucran cinco tablas o más.

10. El uso de vistas pueden causar una ejecución ineficiente de consultas. Muchas veces la ejecución de consultas sobre las tablas base es más eficiente. [20]

Conclusiones:

En el desarrollo de este capítulo quedaron expuestos los requisitos funcionales y los requisitos no funcionales que debe cumplir la BD. Así como, el diseño de la misma, además de una descripción bien detallada de cada una de sus tablas que conforman el diseño. Se describió y fundamentó la arquitectura que presenta la BD lo que posibilita un mejor entendimiento. También se realizó un análisis de la optimización de consultas en PostgresSQL.

CAPITULO 3. VALIDACIÓN DEL DISEÑO REALIZADO.

En este capitulo se realizará una validación teórica y funcional del diseño de la BD, donde se tratarán aspectos como la integridad, normalización de la BD, análisis de redundancia de información, seguridad y la trazabilidad de las acciones.

3.1 Validación teórica del diseño.

3.1.1 Integridad.

La integridad de la base de datos, se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

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 de muchas maneras diferentes. Por ejemplo:

- Pueden añadirse datos no válidos a la base de datos, tales como un pedido que especifica un producto no existente.
- Pueden modificarse datos existentes tomando un valor incorrecto, como por ejemplo si se reasigna una HC en un estante no existente.
- Los cambios en la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de energía.
- Los cambios pueden ser aplicados parcialmente, como por ejemplo si se añade un pedido de un producto sin ajustar la cantidad disponible para vender.
- Una de las funciones importantes de un DBMS relacional es preservar la integridad de sus datos almacenados en la mayor medida posible.

Reglas de integridad en una Base de Datos.

Los conceptos básicos de integridad en el modelo relacional son el de llave primaria, llave foránea, valores nulos *y* dos reglas de integridad.

Una llave primaria es uno o un conjunto de atributos que permiten identificar a las *n-adas* de manera única en cualquier momento.

Una llave foránea de una relación es un atributo que hace *referencia* a una llave primaria de otra relación; esto da pie a que una relación pueda tener varias llaves foráneas.

Un valor nulo es un valor que está fuera de la definición de cualquier dominio el cual permite dejar el valor del atributo ``*latente*''.

Datos Requeridos: Establece que una columna tenga un valor no NULL. Se define efectuando la declaración de una columna es NOT NULL cuando la tabla que contiene las columnas se crea por primera vez, como parte de la sentencia CREATE TABLE.

Chequeo de Validez: Cuando se crea una tabla cada columna tiene un tipo de datos y el DBMS asegura que solamente los datos del tipo especificado sean ingresados en la tabla.

Integridad de entidad: Establece que la clave primaria de una tabla debe tener un valor único para cada fila de la tabla, sino la base de datos perderá su integridad. Se especifica en la sentencia CREATE TABLE. El DBMS comprueba automáticamente la unicidad del valor de la clave primaria con cada sentencia INSERT Y UPDATE. Un intento de insertar o actualizar una fila con un valor de la clave primaria ya existente fallara.

Integridad referencial: asegura la integridad entre las claves ajenas y primarias (relaciones padre/hijo).

Es de gran importancia señalar que para no corromper la integridad referencial de la base de datos se tuvieron en cuenta los cuatro tipos de actualizaciones que la pueden afectar:

- La inserción de una fila hijo es cuando no coincide la clave ajena con la clave primaria del padre.

- La actualización en la clave ajena de la fila hijo, donde se produce una actualización en la clave ajena de la fila hijo con una sentencia UPDATE y la misma no coincide con ninguna clave primaria.
- La supresión de una fila padre, donde si una fila padre tiene uno o más hijos se suprime, las filas hijos quedaran huérfanas.
- La actualización de la clave primaria de una fila padre, donde si una fila padre tiene uno o mas hijos se actualiza su clave primaria, las filas hijos quedaran huérfanas.

Para mantener esta integridad referencial en la Base de Datos Archivo_HC se trazaron las políticas siguientes:

- Restricción, logrando impedir modificar o borrar una tupla con clave primaria relacionada con una secundaria.
- Actualización en cascada, permite modificar la tupla, actualizando las relacionadas.
- Anulación, con esto se logra anular en cascada las relacionadas. [21]

Específicamente se aplicó la restricción para el borrado, y la actualización en cascada para las modificaciones.

Para lograr la integridad de los datos en la BD también se tuvo presente la integridad del dominio, es decir, se realizaron restricciones para algunos atributos, estos verifican que no haya valores fuera del dominio establecido.

- dom_cfg_colorpiel: este verifica que el texto que describe el color de la piel solo pueda ser blanco, negro o mestizo.

Este dominio se asocia al atributo descripción que se encuentra en la tabla cfg_color_piel.

- dom_cfg_sexol: este verifica que el texto que describe al sexo solo pueda ser femenino o masculino.

Este dominio se asocia al atributo descripción que se encuentra en la tabla cfg_sexo

- domain_cfg_estado: este verifica que el texto que describe el estado solo pueda ser activa, pasivo o baja.

Este dominio se asocia al atributo descripción que se encuentra en la tabla cfg_estado.

- domain_no_gaveta este verifica que el campo que indica el número de la gaveta sea un número positivo.

Este dominio se asocia al atributo no _ gaveta que se encuentra en la tabla hc_localizacion_interna.

Para mantener esta integridad referencial en la BD Archivo_HC se trazaron las políticas siguientes:

- Restricción, logrando impedir modificar o borrar una tupla con clave primaria relacionada con una secundaria.
- Actualización en cascada, permite modificar la tupla, actualizando las relacionadas.
- Anulación, con esto se logra anular en cascada las relacionadas.

Específicamente se aplicó la restricción para el borrado, y la actualización en cascada para las modificaciones.

3.1.2 Normalización de la Base de Datos.

El modelo conceptual de datos obtenido mediante la técnica de entidad-relación del Archivo de HC será refinado y convertido en un modelo lógico relacional, con la utilización de la normalización, lo que ofrecerá como resultado el conjunto de tablas a implementar en la base de datos.

Este proceso tiene como finalidad, reducir las inconsistencias y redundancias de los datos, facilitar el mantenimiento y evitar las anomalías en las manipulaciones de datos que en esta área se efectúan.

El objetivo será obtener un modelo lógico normalizado que represente las entidades normalizadas y las interrelaciones existentes entre ellas.

Para ello, se toma como punto teórico de partida el concepto de dependencia funcional, que dice:

"Un atributo B depende funcionalmente de otro atributo A, de la misma entidad si a cada valor de A le corresponde sólo un valor de B." Lo anterior se completa mediante la dependencia funcional total y la dependencia transitiva." [22]

El procedimiento de normalización consiste en someter a las tablas que representan entidades a un análisis formal para ver si cumplen, o no, las restricciones necesarias que aseguren evitar los problemas citados con anterioridad. A mayor nivel de normalización, mayor calidad en la organización de los datos y menor peligro para la integridad de los datos. Este procedimiento consiste en ir alcanzando formas normales.

Cuando se está en presencia de una primera relación que contiene enormes problemas de redundancia, consistencia e integridad de los datos, es necesario mejorar estas relaciones. Estas mejoras deben dar como resultado tablas equivalentes y mejores que sus respectivas originales, y poseer siempre tres propiedades: conservación de la información (de atributos y de tuplas), conservación de dependencias y mínima redundancia de los datos. Las mejoras introducidas obligan a plantear hasta que Forma Normal es necesario llegar, es decir, a que nivel de depuración.

Existen básicamente 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.

Pero no siempre es una buena idea tener una base de datos conformada en el nivel más alto de normalización, puede llevar a un nivel de complejidad que pudiera ser evitado si estuviera en un nivel más bajo de normalización.

Primera Forma Normal (1NF)

Una relación está en primera forma normal (1FN) si los valores para cada atributo de la relación son atómicos. Esto quiere decir simplemente que cada atributo sólo puede pertenecer a un dominio, que tiene un valor único para cada fila. Esta prohíbe los atributos multievaluados, compuestos y sus combinaciones.

En el primer diseño de la BD del Bloque Quirúrgico se encontraron en ocasiones varios atributos compuestos, al aplicar esta regla se realizaron modificaciones en las tablas para alcanzar este grado de normalización y de esta forma conseguir menor espacio en disco.

Segunda forma normal (2FN):

Una relación está en segunda forma normal si está en la 1ª FN y todos los atributos no clave dependen de la clave completa y no sólo de una parte de esta.

Este paso sólo se aplica a relaciones que tienen claves compuestas, es decir, que están formadas por más de un atributo como llave. Si un esquema de relación no está en 2ª FN, se le puede normalizar a varias relaciones en 2ª FN en las que los atributos que dependen de una parte de la clave formarán una nueva relación que tendrá esa parte de la clave como clave primaria.

La segunda forma normal compara todos y cada uno de los campos de la tabla con la clave definida. Si todos los campos dependen directamente de la clave se dice que la tabla está es segunda forma normal.

Una vez que ha alcanzado el nivel de la Segunda Forma Normal, se han controlado la mayoría de los problemas de lógica. Se puede insertar un registro sin un exceso de datos en la mayoría de las tablas.

Tercera forma normal (3FN):

Una relación está en tercera forma normal si todos los atributos de la relación dependen funcionalmente sólo de la clave, y no de ningún otro atributo.

Esto significa que en una relación en 3FN, para toda DF: X ⟶Y, X es una clave.

Aquí se observa que si una relación está en tercera forma normal, está también en segunda forma normal, sin embargo lo inverso no siempre es cierto. [23]

La aplicación de estas Formas de Normalización incorpora a la BD del módulo funcionalidades óptimas que ayudan a obtener correctos resultados cuando se realizan consultas.

Otra ventaja de la normalización 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 un menor uso de espacio en disco.

El Modelo entidad Relación de la BD de Archivo de HC se modeló aplicando las reglas de la normalización hasta lograr que se encontrara en 3ª FN. No se modeló hasta otra forma normal más avanzada porque puede conducir a tener una base de datos ineficiente y construir un esquema demasiado complejo para trabajar. Un balance apropiado de sentido común y práctico puede ayudar a decidir hasta que nivel se normalizará.

3.1.3 Seguridad.

La seguridad de la BD en PostgresSQL está implementada en varios niveles:

- Protección de los ficheros de la base de datos. Todos los ficheros almacenados en la base de datos están protegidos contra escritura por cualquier cuenta que no sea la del usuario Postgres.
- Las conexiones de los clientes al servidor de la base de datos están permitidas, por defecto, únicamente mediante sockets Unix locales y no mediante sockets TCP/IP. Ha de arrancarse el demonio con la opción -i para permitir la conexión de clientes no locales.
- Las conexiones de los clientes son restringidos por dirección IP y/o por nombre de usuario mediante el fichero pg_hba.conf situado en PG_DATA.
- Las conexiones de los clientes pueden ser autentificadas mediante otros paquetes externos.
- A cada usuario de Postgres se le asigna un nombre de usuario y (opcionalmente) una contraseña.

Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse en base a esos grupos.

El acceso a la base de datos también es controlado por un mecanismo de contraseñas; un usuario que quiera acceder al sistema debe dar una contraseña y que el sistema la valide.

Autentificación de Usuarios:

Autentificación es el proceso mediante el cual el servidor de la base de datos y el *postmaster* se asegura de que el usuario que está solicitando acceso a la base de datos es en realidad quien dice ser. Todos los usuarios que quieren utilizar Postgres se comprueban en la tabla pg_user para asegurarse que están autorizados a hacerlo. Actualmente, la verificación de la identidad del usuario se realiza de distintas formas:

Desde la *shell* del usuario:

> Un demonio que se lanza desde la *shell* del usuario anota el id original del usuario antes de realizar un *setuid* al id del usuario *postgres*. El id original del usuario se emplea como base para todo tipo de comprobaciones.

Desde la red:

> Si Postgres se instala como distribuido, el acceso al puerto TCP del *postmaster* está disponible para todo el mundo. El ABD configura el fichero pg_hba.conf situado en el directorio PG_DATA especificando el sistema de autentificación a utilizar en base al equipo que realiza la conexión y la base de datos a la que se conecta. Por supuesto la autentificación basada en equipos no es perfecta incluso en los sistemas Unix. Es posible, para determinados intrusos, enmascarar el equipo de origen. Estos temas de seguridad están fuera del alcance de Postgres. [24]

3.1.4 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 BD.

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 *("duration").*

En la base de datos del módulo Archivo de HC la trazabilidad de las acciones 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.

3.1.5 Análisis de la redundancia de información.

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.

La redundancia en el almacenamiento de los datos provoca varios problemas:

- En primer lugar, es necesario realizar una misma actualización lógica varias veces: una vez en cada archivo en el que se almacenen datos. Esto implica una duplicación del trabajo.

- En segundo lugar se desperdicia espacio de almacenamiento al guardar los mismos datos en varios lugares.
- En tercer lugar, es posible que los archivos que representen los mismos datos sean inconsistentes, quizás porque la actualización se haya aplicado a varios archivos y no a otros.

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 Archivo de HC toda la información que se maneja es entorno a la gestión de las HC.

Redundancia Mínima:

Para que una base de datos sea efectiva se necesita eliminar en la medida de lo posible las redundancias, es decir, las repeticiones que puedan llevar a error, como el llamar a un mismo campo de distinta manera en varios archivos, ya que si no existe el riesgo de inconsistencia entre las distintas versiones de los mismos datos. Lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

En la BD Archivo de HC se ha tenido en cuenta que si se desea ingresar una nueva HC se realizara un búsqueda y de ser encontrada solo se le agregarían los datos de esta a la ya existente, de esta manera se evitará la repetición de las mismas. Por otra parte no existe la repetición de un mismo campo en otras tablas, es decir que cada campo es único en la BD garantizando que no haya redundancia de información.

Conclusiones:

En este capítulo se realizó la validación teórica del diseño de la BD propuesta, exponiéndose los métodos que se utilizaron para asegurar el buen funcionamiento de la misma.

CONCLUSIONES

Se logró el objetivo trazado con el diseño de una Base de Datos que facilita el acceso a la información de los pacientes mediante la gestión de las Historias Clínicas en el departamento Archivo de HC de los centros hospitalarios del país.

Se han desarrollado satisfactoriamente las tareas trazadas para darle cumplimiento al objetivo:

- Se realizó un estudio preliminar de los SIH más utilizados actualmente, permitiendo un mejor diseño de la misma.
- Se realizo una validación funcional del diseño realizado.
- Se estructuró y diseñó un modelo conceptual de la BD cumpliendo con lo establecido.
- Se realizó la implementación de funciones (procedimientos de almacenado), vistas y dominios de la BD que permiten controlar la gestión de la información relacionada con el módulo Archivo de HC.

RECOMENDACIONES

- Con este trabajo se exhorta a una profundización del tema abordado para así realizar mejoras en el diseño realizado.
- Continuar con la investigación para garantizar nuevas mejoras en futuras versiones de la base de datos del sistema propuesto.
- Para una nueva versión de la BD llevar a cabo la trazabilidad de las acciones a nivel de aplicación, no solo en la BD.

REFERENCIAS BIBLIOGRAFICAS

[1] IS. K. *Computer-based patient records*. New York: Schattauer, 1999. 227, 228, 229 p. KENDALL, K. *Análisis Y Diseño De Sistemas 3ra edición*.

[2] C. J. DATE, *Sistemas de Bases de Datos 1era Parte.*

[3] SCHNEIDER, G. M. A. G., J.L. *An invitation to computer science*. Pacific Grove, 1998. p.

[4] MONOGRAFÍAS.COM. *SQL Server*, 2004. [2007]. Disponible en: http://www.monografias.com/trabajos14/sqlserver/sqlserver.shtml

[5] GÓMEZ, A. D. V. B. *Oracle*, 2004. [2007]. Disponible en: http://www.monografias.com/trabajos25/oracle/oracle.shtml

[6] SUAREZ, A. R. M. Y. M. F. *Propuesta de arquitectura para un sistema de información hospitalaria*, UCI, 2007. p.

[7] KENDALL, K. *Análisis Y Diseño De Sistemas 3ra edición*.

[8] LAITANO, P. A. Y. R. *Ingeniería de Software Asistido por Computadora*, 2004. [2007]. Disponible en: http://www.monografias.com/trabajos14/modelodebase/modelodebase.shtml

[9] TERRA, D. *Toad*, 2003. [2007]. Disponible en: http://descargas.terra.es/ie/26344-Toad

[10] SUAREZ, A. R. M. Y. M. F. *Propuesta de arquitectura para un sistema de información hospitalaria*, UCI, 2007. p.

[11] FILEHEAVEN.COM. *CASE Studio*, 2006. [2007]. Disponible en: http://www.fileheaven.com/descargar/case-studio-2/26647.htm

[12] BAJAME.NET. *EMS SQL Manager 2005 Lite for PostgreSQL* 2007. [2007]. Disponible en: http://www.bajame.net/EMS-SQL-Manager-2005-Lite-for-PostgreSQL-3_5_0_1.htm

[13] CORPORATION, V. *Histotias Clínicas Electrónicas en Español*, 2006. [2007]. Disponible en: http://www.medical-soft.com/

[14] CORPORATION, V. *Histotias Clínicas Electrónicas en Español*, 2006. [2007]. Disponible en: http://www.medical-soft.com/

[15] COMPUTER, V. *Gowin HC - Historia clínica electrónica* 2005. [2007]. Disponible en: http://www.valen.es/cas/gowin_hc.php

[16] GIBBA. *Despalizacion de instituciones de salud*, 1998. [2007]. Disponible en: http://www.biocom.com/servicios/despapelizacion.html

[17] GONZÁLEZ, Ó. G. M. Y. F. R. *ARQUITECTURAS DE SISTEMAS DE BASES DE DATOS*, 1999/2000. [2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node33.html

[18] GONZÁLEZ, Ó. G. M. Y. F. R. *ARQUITECTURAS DE SISTEMAS DE BASES DE DATOS*, 1999/2000. [2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node33.html

[19] MORALES, A. A. M. OPTIMIZACIO N DE CONSULTAS SOBRE FUENTES DE DATOS CON CAPACIDADES LIMITADAS: VISION GENERAL, 2005.

[20] MORALES, A. A. M. OPTIMIZACIO N DE CONSULTAS SOBRE FUENTES DE DATOS CON CAPACIDADES LIMITADAS: VISION GENERAL, 2005.

[21] ANDRÉS, M. M. M. *Ventajas e inconvenientes de los sistemas de bases de datos* 2001. [2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node7.html

[22] Diseño de Base de datos relacionales. Disponible en:

http://usuarios.lycos.es/cursosgbd/UD4.htm

[23] Diseño de Base de datos relacionales. Disponible en:

http://usuarios.lycos.es/cursosgbd/UD4.htm

[24] ZOTTO, M. D. *Guia del Administrador de PostgreSQL*, 2007]. Disponible en:
http://es.tldp.org/Postgresql-es/web/navegable/admin/security.html

BIBLIOGRAFÍA

ANDRÉS, M. M. M. *Ventajas e inconvenientes de los sistemas de bases de datos* 2001. [2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node7.html

BAJAME.NET. *EMS SQL Manager 2005 Lite for PostgreSQL* 2007. [2007]. Disponible en: http://www.bajame.net/EMS-SQL-Manager-2005-Lite-for-PostgreSQL-3_5_0_1.htm

COMPUTER, V. *Gowin HC - Historia clínica electrónica* 2005. [2007]. Disponible en: http://www.valen.es/cas/gowin_hc.php

CORPORATION, V. *Histotias Clínicas Electrónicas en Español*, 2006. [2007]. Disponible en: http://www.medical-soft.com/

Diseño de Base de datos relacionales. Disponible en:

http://usuarios.lycos.es/cursosgbd/UD4.htm

E. Tovar, Algoritmo Evolutivo para Preoptimizar Consultas en Fuentes de Datos con Capacidades Limitadas, Trabajo de Grado de Maestría en Ciencia de la Computación, Universidad Simón Bolívar, Caracas, Venezuela, 2003.

FILEHEAVEN.COM. *CASE Studio*, 2006. [2007]. Disponible en: http://www.fileheaven.com/descargar/case-studio-2/26647.htm

GIBBA. *Despalizacion de instituciones de salud*, 1998. [2007]. Disponible en: http://www.biocom.com/servicios/despapelizacion.html

GÓMEZ, A. D. V. B. *Oracle*, 2004. [2007]. Disponible en: http://www.monografias.com/trabajos25/oracle/oracle.shtml

GONZÁLEZ, Ó. G. M. Y. F. R. *ARQUITECTURAS DE SISTEMAS DE BASES DE DATOS*, 1999/2000. [2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node33.html

KENDALL, K. *Analisis Y Diseño De Sistemas 3ra edicion*.

LAITANO, P. A. Y. R. *Ingeniería de Software Asistido por Computadora*, 2004. [2007]. Disponible en: http://www.monografias.com/trabajos14/modelodebase/modelodebase.shtml

L. Raschid, M. E. Vidal and V. Zadorozhny, An Optimizer Strategy to Generate Good Low Cost Plans for Mediator Queries, Technical Report, University of Maryland, 2001.

MONOGRAFÍAS.COM. *SQL Server*, 2004. [2007]. Disponible en: http://www.monografias.com/trabajos14/sqlserver/sqlserver.shtml

MORALES, A. A. M. OPTIMIZACIO N DE CONSULTAS SOBRE FUENTES DE DATOS CON

CAPACIDADES LIMITADAS: VISIÓN GENERAL, 2005.

P. Seshadri, PREDATOR: Design and Implementation,

1997. Disponible: http://www.cs.cornell.edu/Info/Projects/PREDATOR/designdoc.html

SUAREZ, A. R. M. Y. M. F. *Propuesta de arquitectura para un sistema de información hospitalaria*, UCI, 2007. p.

TERRA, D. *Toad*, 2003. [2007]. Disponible en: http://descargas.terra.es/ie/26344-Toad

ZOTTO, M. D. *Guía del Administrador de PostgreSQL*, 2007]. Disponible en: http://es.tldp.org/Postgresql-es/web/navegable/admin/security.html