Universidad de las Ciencias Informáticas
Facultad 7

Título: Gestión de Datos en el Módulo de Gestión de la Información de los Bancos de Sangre en hospitales.

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

Autor: Yanelis Preval Creach.
Tutor: Lic. Angel Fabra Torres.
Asesor: Lic. Juana Isabel Pérez.

Ciudad de la Habana
Julio 2007

"La informática no se debe limitar a una práctica técnica, debe servir para darle el rumbo correcto a la sociedad."

Anónimo

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los 6 días del mes de Julio del año 2007.

Yanelis Preval Creach

_____

Firma del Autor

Angel Fabra Torres

_____

Firma del Tutor

DATOS DE CONTACTO

Lic. Angel Fabra Torres

Graduado de Licenciado en Ciencias de la Computación por la Universidad de Oriente, año 2004. Actualmente labora como profesor en la Universidad de las Ciencias Informáticas. Posee la categoría docente de Instructor.
e-mail: afabra@uci.cu

Lic. Juana Isabel Pérez Rodríguez

Graduada de Licencianda en Lengua Inglesa de la Facultad de Lenguas Extranjeras del Instituto Superior Pedagógico “Enrique José Varona”, año 2000. Actualmente labora como profesora en la Universidad de las Ciencias Informáticas de la asignatura Idioma Extranjero. Posee la categoría docente de Instructora.

e-mail: jane@uci.cu

AGRADECIMIENTOS

Agradezco a todos los profesores que aportaron algo en mi vida estudiantil contribuyendo en mi formación.

Al profesor Rudel Cardenas Díaz por guiarme y orientarme durante la realización de este trabajo.

Al profesor Yosvanys Sánches Corales, cariñosamente Yovanotti, por todo el apoyo brindado.

A mi tutor Angel Fabra Torres por guiarme y apoyarme durante el desarrollo del mismo.

A mis compañeros de estudio Alain, Alexander, Daniel, Kenia, Keila y a todos aquellos que de una forma u otra dieron su aporte para que este trabajo fuera posible.

A todos ustedes, les doy mil gracias por haberme ayudado a ver este sueño convertido en realidad. Les digo hasta pronto, y les garantizo que los extrañaré muchísimo

DEDICATORIA

Dedico este trabajo de diploma:

- *A mis padres, por la enseñanza que me han dado, su amor, paciencia, sacrificios, dedicación y consejos. Además porque son las personas más importantes en mi vida.*
- *A nuestro comandante Fidel Castro Ruz, por darme la posibilidad de estudiar en esta gran universidad.*
- *A mi gran amor Richard, por sus consejos, paciencia y amor y por ser el centro de mi vida. Prometo que sabré recompensar todo tu sacrificio y dedicación.*
- *A mi amiga Yunia que siento como hermana, y que en las buenas y malas siempre estuvo a mi lado.*
- *A mis amigas de la universidad Keila, Nadiezka, Yenisel, Kenia, Yanersy por compartir y apoyarme en los momentos buenos y malos.*

A todos ustedes les doy muchas gracias por toda su comprensión y amor, ustedes forman parte de este sueño.

RESUMEN

En los bancos de sangre existe un gran manejo de información. Esto se debe a la cantidad de procesos que se llevan a cabo, con el fin de cumplir su tarea de proporcionar sangre con calidad total a todo el que la necesite. El control de la información se hace cada vez más difícil debido al volumen de la misma, el cual crece diariamente.

En este trabajo se realiza un estudio detallado del funcionamiento de los bancos de sangre, principalmente de la información que es necesaria recoger, con el objetivo de diseñar una base de datos que permita almacenar toda la información que se gestiona en estas instituciones de forma manual.

Para llevar a cabo el objetivo trazado se hará uso de herramientas que posibilitarán el diseño y desarrollo de la base de datos. Tal es el caso de Case Studio como herramienta para modelarla y PostgreSQL como sistema gestor de base de datos. Con el uso de estas herramientas se garantiza una parte importante del desarrollo, logrando satisfacer las necesidades de estos importantes centros.

Con el resultado de este trabajo se obtendrá una base de datos con una buena seguridad, que garantice la confidencialidad de los datos y protección de los mismos. Con ella se contribuirá a que la consulta, almacenamiento y control de la información que se manipula en los bancos de sangre, sea cada vez más fácil, lo que traerá como consecuencia una mejor calidad de los servicios asistenciales prestados.

TABLA DE CONTENIDOS

TABLAS DE CONTENIDOS

ÍNDICE DE FIGURAS Y TABLAS

ÍNDICE DE FIGURAS Y TABLAS

INTRODUCCIÓN

La informatización de la sociedad es el proceso de utilización ordenada y masiva de las Tecnologías de la Información y las Comunicaciones (TIC) en la vida cotidiana. Con ella se satisfacen las necesidades de todas las esferas de la sociedad, en su esfuerzo por lograr eficacia en todos los procesos y por consiguiente, un aumento en la calidad de vida de los ciudadanos. Una sociedad que aplique la informatización en todas sus esferas y procesos será más eficaz y competitiva.

La informática en el mundo ha tenido un crecimiento vertiginoso. El auge de las comunicaciones con las nuevas tecnologías de la información no permite que ninguna esfera económica o social pueda pensar en el desarrollo si no es con la presencia de esta herramienta tan importante.

Son innumerables los beneficios que brinda la computación: rapidez en la obtención de resultados, almacenamiento de grandes volúmenes de información, facilidades para encontrar información adecuada y/o actualizada por parte de científicos, investigadores, profesionales, estudiantes. En Cuba se trabaja para ir incorporando de forma progresiva estos elementos al servicio de toda la población.[1]

Se ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las TIC; y lograr una cultura digital como una de las características imprescindibles del hombre nuevo. Esto facilitaría a la sociedad acercarse más hacia el objetivo de un desarrollo sostenible.

Cuba enfrenta el reto de informatizar su sociedad con el propósito de integrarse plenamente a la infraestructura de la información y hacer uso óptimo de las nuevas tecnologías, lo que permitirá lograr incrementos sustanciales en la productividad, calidad y la eficiencia en toda la actividad tanto industrial como de servicios.[2]

El Sector de la Salud no está ajeno a esto y desarrolla la informática médica, que es la intercepción entre las ciencias de la salud y las del procesamiento de la información.

---

[1] KATIA CARABALLOSO GRANADO, Y. T., MABEL CHAU LEY *Las Nuevas Tecnologías de la Información y las Comunicaciones en las Bibliotecas Universitarias: presentación de un caso*, 25 de noviembre de 2006]. Disponible en: http://www.sociedadelainformacion.com/Principal/n10articulos07/Las%20Nuevas%20Tecnologias.pdf

[2] GARCÍA, R. F. S. *Revista de Ciencias Médicas LA INFORMATICA MEDICA EN CUBA*., 25 de noviembre de 2006]. Disponible en: http://www.cpicmha.sld.cu/hab/vol6_2_00/hab070200.htm

INTRODUCCIÓN

El Sector de la Salud no está ajeno a esto y desarrolla la informática médica, que es la intercepción entre las ciencias de la salud y las del procesamiento de la información.

El uso de la Informática en la Medicina es una de las aplicaciones más comunes e importantes desde hace varias décadas, y ha permitido al sector de la salud, no sólo contar con métodos novedosos, sencillos y eficaces de gestión administrativa en consultas, hospitales y centros de investigación biomédica, sino también disponer de complejos software que reducen la posibilidad de error en el diagnóstico de las enfermedades, y que aceleran su formulación. Todo esto conlleva a que se logre un perfeccionamiento de la calidad asistencial ofrecida a la sociedad, facilitar las funciones del personal de la salud y colaborar con la gestión administrativa, asistencial, docente y de investigación.

La seguridad del enfermo durante la asistencia sanitaria, es un tema que tradicionalmente ha preocupado a la sociedad, y es por ello que continuamente se diseñan estrategias que disminuyan el error del personal médico. Precisamente se ha encontrado un gran apoyo en las tecnologías de la información, las cuales ya han demostrado enormes beneficios para la medicina.

A raíz del surgimiento de las computadoras, comenzaron a crearse sistemas sencillos de información, tanto con fines administrativos como financieros.[3]

En la década de los 70's es cuando aparecen en el ámbito médico los primeros sistemas de información "médica" que, posteriormente, dan lugar a los Sistema de Información Hospitalaria (SIH), tan indispensables en la actualidad. Los SIH son sistemas de información y están orientados a satisfacer las necesidades de almacenar y procesar datos médicos de cualquier institución hospitalaria. Esto permite que el trabajo de los hospitales sea más óptimo y eficiente. El impacto de estos sistemas en las instituciones de salud es fuerte, pues elevan la calidad de la atención al paciente y servicios brindados.[4]

Los SIH actuales distan de ser perfectos y la realidad es que se desarrollan numerosas versiones para mejorar continuamente los ya implementados. Estos están compuestos por módulos, que representan los subsistemas que lo conforman, y Banco de Sangre es uno de estos módulos.

---

[3] Ídem referencia 1
[4] FLORINA GATICA LARA, F. J. F. P., ANA MARÍA HERNÁNDEZ LÓPEZ. *Sistema de Información Hospitalaria*, 23 de mayo de 2007]. Disponible en: http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf

INTRODUCCIÓN

Los bancos de sangre son los responsables de la disposición de productos sanguíneos para la realización de los diferentes procedimientos médicos que se les prescriben a los pacientes en los servicios asistenciales.[5]

Un problema que se presenta diariamente en el banco de sangre es el manejo del donante con donaciones anteriores. Tradicionalmente se han empleado tarjeteros que son utilizados en la admisión y que deben permitir a la recepcionista reunir algunos elementos orientadores sobre la conveniencia o no del ingreso al banco de sangre de tales personas. Estos tarjeteros son engorrosos de manipular y de actualizar y no cuentan con toda la información necesaria. Adicionalmente, por razones bioéticas, no contienen los resultados de las pesquisas realizadas a la sangre para la determinación de gérmenes que se trasmiten a través de las transfusiones, ya que esta es considerada una información confidencial: sífilis (VDRL), virus de la inmunodeficiencia humana (VIH), hepatitis viral tipo B (HVB), hepatitis viral tipo C (HVC).

Un aspecto de gran importancia en el momento del ingreso del futuro donante al banco de sangre es la fecha de la última donación. Aunque existen amplias variaciones en el mundo en relación con el intervalo mínimo entre donaciones de sangre, ha sido establecido en el país como medida de protección a la salud del individuo, que deben haber transcurrido más de 3 meses de la fecha de la última donación para admitir al donante. Sorprendentemente, muchos olvidan su reciente donación o incluso la niegan. También es conveniente conocer los antecedentes patológicos o de carácter epidemiológico que pueden haber sido diagnosticados en el interrogatorio o en el examen médico practicado en una donación anterior. Estos datos se recogen normalmente en la historia clínica y se archivan, y su consulta resulta difícil al momento de la nueva donación, lo que de ordinario no se practica.

Finalmente, se desea conocer si el donante tuvo algún percance en su donación anterior, lo que permitiría también al médico tomar una decisión al respecto; tal es el caso de los donantes que hacen una reacción vagal severa, hipersensibilidad local o los que tienen un sistema venoso de difícil venipuntura.

[5] CHÁVEZ", I. N. D. C. I. *Banco de Sangre*, 27 de noviembre de 2006]. Disponible en: http://www.cardiologia.org.mx/incic/ban_sangre/

INTRODUCCIÓN

Con respecto a las transfusiones, es necesario conocer con antelación la información correspondiente a los pacientes que serán transfundidos en un día específico debido a una intervención quirúrgica, por ejemplo: el grupo sanguíneo, factor RH, Nombre y Apellidos. Esto ayuda a verificar la existencia y disponibilidad de sangre, así como de sus componentes, con el fin de aprobar o cancelar la transfusión y postergarla para otro día. Este proceso es complejo de efectuar, pues el mismo día en que el paciente es informado que será sometido a una operación, es enviado a realizarse el grupo sanguíneo y factor RH.

Estos datos son registrados por una secretaria en un libro que carece de formularios. En este los pacientes son registrados consecutivamente, lo que puede dificultar la búsqueda de los mismos en el futuro ya que las operaciones pueden ser programadas para un día y mes determinado y el registro de sus datos pudo haber sido meses antes.

Teniendo en cuenta la importancia de todos los procesos que se llevan a cabo en los bancos de sangre; buscando la posibilidad de aminorar los errores que puedan introducirse o producirse por el personal médico encargado del control de toda la información; buscando la manera de que se pierda menos tiempo a la hora de prestar servicios y analizando el gran flujo de información existente, se identifica el siguiente **problema:** ¿Cómo automatizar la gestión de la información relacionada con los procesos llevados a cabo en los bancos de sangre de los hospitales cubanos?

Este problema presenta como **objeto de estudio:** Proceso de gestión de la información en los bancos de sangre.

El **campo de acción** que abarca es: Proceso de almacenamiento de la información gestionada en los procesos que se realizan en los bancos de sangre del país.

Objetivo general

Diseñar una base de datos que mejore la gestión de la información referente a los procesos vinculados a los bancos de sangre del país.

Las **tareas** que se llevaran a cabo para dar cumplimiento al objetivo trazado son:

- Realizar un estudio de las herramientas, definidas por el arquitecto, para diseñar y desarrollar la base de datos (BD).

- Obtener los artefactos de la metodología RUP que describen la BD.
- Diseñar y estructurar un modelo relacional que permita visualizar la BD.
- Implementar vistas, dominios y procedimientos almacenados del módulo que permitan el acceso a los datos.
- Hacer una validación teórica del diseño de la BD propuesto.

Métodos utilizados para la realización del trabajo

Se hacen uso de métodos teóricos y empíricos que contribuyen a la exitosa realización de este trabajo. Los métodos teóricos ayudan a un mejor estudio de las características del proceso de gestión de la información en los bancos de sangre. Los métodos empíricos permitirán recolectar la información necesaria para diseñar una base de datos que almacene toda la información gestionada en los bancos de sangre.

Se utilizan de los métodos teóricos: el método Modelación y el método Analítico Sintético; y de los métodos empíricos: el método Entrevista. El uso de estos, permitirá llegar a la conclusión de cuáles son los procesos que necesiten ser automatizados y cuál es la información a almacenar en la BD.

A través del **método analítico sintético** se analizan todas las teorías y documentos existentes relacionados con el fenómeno que se estudia, con el objetivo de extraer de estos, los elementos más importantes para la realización de la BD.

A través de **la modelación** se representará de manera simplificada la realidad formulada por una persona con total conocimiento acerca de todas las acciones y procesos que se llevan a cabo en la institución. Con la explicación de dichas personas y la aplicación del método analítico sintético se modelará la base de datos teniendo en cuenta la información que requiere ser almacenada. Para ello se usará el modelo teórico el cual se auxilia de símbolos para designar las propiedades del sistema real que se desea estudiar.

La entrevista es otro de los métodos que se usará y constituye uno de los métodos empíricos. Esta es muy importante para conocer detalladamente el funcionamiento de un banco de sangre pues a través de

ella se obtendrá toda la información necesaria. Es apoyándose en la entrevista que se obtiene la explicación de la que se habla en el método de modelación.

La entrevista está constituida por preguntas relacionadas con el tema del que se quiere conocer, que serán las que ayuden a dialogar con el cliente, de manera que a este le sea más fácil realizar una clara explicación de todo el proceso de interés. La entrevista será no estructurada, pues durante el diálogo pueden surgir preguntas no previstas.

Son estos tres métodos los que se utilizará para cumplir el objetivo de la investigación.

Con el presente trabajo se espera obtener un producto que como resultado de esta investigación permita la integración de todas las aplicaciones. Con él se garantizará que la información sea consistente, que no esté duplicada y que además sea precisa. Además, tendrá como una de las ventajas principales que no existan sistemas aislados, incapaces de proporcionar una gran utilidad y beneficios al usuario final. Al concebirse de manera integrada se logra un mejoramiento de la actividad administrativa, asistencial, docente y de investigación; pues los datos generados en los distintos niveles de atención en el Sistema Nacional de Salud (SNS), tienen un proceso de captura, registro, procesamiento, validación y análisis de la información inestimable, lo cual incrementa su consistencia, veracidad y oportunidad.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

Actualmente existen muchas empresas y sitios webs que necesitan almacenar y administrar un gran volumen de información. Para ello emplean gestores de base de datos que son sistemas que permiten la realización y administración de las BD. En el mundo existen muchos gestores que ofrecen innumerables ventajas para administrar una BD, tan necesarias y útiles en la actualidad para empresas, sitios, etc. que necesitan almacenar un gran volumen de información.

Actualmente existe una gran competencia entre los gestores, ¿Cuál es el mejor gestor?, ¿Cuál debo usar?, son preguntas que normalmente surgen cuando se presenta un problema y para su solución se necesita del desarrollo de una BD.

En este capítulo se realizará un análisis de la herramienta utilizada para modelar una BD. Además se hará un estudio de los gestores más conocidos a nivel mundial con el objetivo de seleccionar el más adecuado para construir una BD capaz de almacenar una gran cantidad de datos de manera íntegra y segura. Se estudiará también algunos SIH que contienen el módulo de banco de sangre y sistemas que sólo son dedicados a automatizar el trabajo en un banco de sangre, resaltando los sistemas gestores de base de datos (SGBD) empleados en el desarrollo de los mismos.

1.1 Sistemas automatizados existentes vinculados al tema tratado.

Actualmente existen varios sistemas dedicados al manejo de la información en los bancos de sangre. Seguidamente se hace referencia a algunos de ellos:

Delphyn

Es un sistema que sólo se dedica a la administración y manejo de los datos de un banco de sangre. Este sistema es una herramienta flexible para el manejo de los datos en los bancos de sangre, posibilitando una informatización de todas las actividades (donantes y pacientes) permitiendo a los usuarios un fácil

control de los componentes sanguíneos. Contiene módulos para el manejo de donantes, pacientes y el almacenamiento de los componentes sanguíneos.[6]

Es desarrollado sobre una arquitectura cliente/servidor. Puede usar cualquiera de los gestores: Oracle, SQL Server, Interbase, Sybase e Informix, en los servidores de Windows NT/2000 y de Unix.[7]

xBlood

Es un sistema para el apoyo operacional y el control administrativo integral de un banco de sangre o centro transfusional. Es flexible, fácil de utilizar y económico. Controla el manejo de donadores, receptores, así como los procesos y movimientos efectuados a las unidades de sangre. Permite un acceso controlado a las diversas funciones del banco de sangre llevando de la mano al usuario a través de las diferentes etapas de la donación por medio de un flujo de trabajo, así como de los procesos de transfusión, validación, cesión, reservación y destrucción de las unidades.[8]

Entre sus características generales se encuentra la posibilidad que brinda de utilizar cualquier BD para almacenar la información de pacientes y resultados. Además está disponible una versión monousuario y otra multiusuario para el trabajo en red.[9]

Care2x

Care2x es un SIH que integra datos, funciones y flujo de tareas en un entorno de cuidados de la salud. Tiene módulos para las diferentes funciones y departamentos como son admisión, turnos de paciente ambulatorio, historia clínica básica, administración de recursos humanos, quirófanos, banco de sangre y otros. Usa una BD única y un formato de datos único. La versión beta actual soporta la BD MySQL que resuelve la incompatibilidad de datos. Los datos pueden ser intercambiados a través de la red completa. Utiliza un lenguaje estándar en la BD, SQL, que soluciona el problema de dependencia de plataforma. Con este lenguaje se provee un mejoramiento en la adaptación de los módulos de programa existentes,

[6] DCSOFT. *DELPHYN Blood Bank Data Management System* 14 de marzo de 2007]. Disponible en: http://www.dcsoftintl.com/MedicApps.asp
[7] Ídem referencia 2
[8] GROUP, V. *xBlood - Sistema para el manejo de bancos de sangre*, 5 de febrero de 2007]. Disponible en: http://www.virtus.com.mx/xblood/index.html
[9] Ídem referencia 4

así como en la creación de nuevos módulos de programa de manera fácil y rápida, provocando que los costos y el tiempo de desarrollo disminuyan substancialmente.[10]

Este sistema presenta además un buen mecanismo de seguridad y su acceso es protegido por un sistema de autenticación. Los datos son únicamente liberados a una organización que lo solicita cuando el dueño de los datos tiene establecido un permiso para dicha organización.[11]

OpenHIS

OpenHIS es el SIH del nuevo proyecto del Grupo BioLinux para cubrir la necesidad de un sistema informático hospitalario integral. OpenHIS es un desarrollo creado totalmente desde cero, con nuevo diseño, y con el respaldo que el Grupo BioLinux puede ofrecer en su larga trayectoria en análisis, diseño y desarrollo de sistemas informáticos para el ámbito de la salud. Significa Entorno Abierto de Información para Salud (Open Health Information System Environment) y esta diseñado con HOME (Hospital Object Model Environment), un nuevo modelo de objetos para el análisis y creación de sistemas informáticos hospitalarios, desarrollado por el Grupo BioLinux.[12]

Como capa para la interacción con BDs se utiliza ADOdb, esto provee una abstracción con respecto al motor de BD (MySQL, POSTGRES), aunque actualmente solo se soporta MySQL 4.0.1 o una versión mayor.[13]

Galen Hospital

Este sistema se encuentra instalado en el hospital “Hermanos Ameijeiras” en su versión 2.0. Este es un SIH desarrollado en plataforma Windows 32 bits (Windows NT y Windows 98) con una configuración Cliente/Servidor y el uso del gestor de bases de datos relacional SQL Server 7.0 como reservorio de la información. Para la construcción de la BD se utilizó el Sistema de Gestión de Bases de datos Relacional SQL Server 7.0, a partir del cual se realizaron consultas y procedimientos.[14]

---

[10] LATORILLA, E. *Sitio de care2x*, 12 de marzo de 2007]. Disponible en: http://www.care2x.org/
[11] Ídem referencia 6
[12] BIOLINUX, G. *Sistema Informático Hospitalario Abierto.*, 12 de marzo de 2007]. Disponible en: http://www.openhis.com.ar/blog/
[13] Ídem referencia 8
[14] CÉSAR, C. R. *BOLETIN - Instituto de Información Científica y Tecnológica*, 12 de marzo de 2007]. Disponible en: http://www.idict.cu/UserFiles/File/Boletines/Novedades/boletin02_0105/Inicio020105.html

Uno de los módulos del "Galen" es el de Banco de Sangre, con el que se puede gestionar la distribución de sangre y hemoderivados al mantener una revisión del stock por valores máximos, mínimos y fechas de vencimiento, además de emitir órdenes de aprovisionamiento de productos. También genera los exámenes necesarios para garantizar la compatibilidad de los componentes almacenados con la sangre del paciente y controla la entrada de bolsas enviadas por las unidades proveedoras.[15]

1.2 Análisis de las tecnologías a utilizar

A continuación se realizará un análisis de los gestores de base de datos más conocidos a nivel mundial. Además se fundamentará el uso de la herramienta case utilizada para modelar la BD.

1.2.2 Sistemas gestores de base de datos

Entre los SGBD más usados a nivel mundial se encuentran, Oracle, SQLServer, MySQL y PostgreSQL. De ellos se realizará un estudio de sus principales características con el fin de elegir cuál utilizar en la solución del problema.

Oracle

Oracle es un SGBD fabricado por la Corporación Oracle. Se considera como uno de los sistemas de bases de datos más completos, destacándose entre sus características generales:

- Soporte de transacciones, que no son más que una interacción con una estructura de datos que, aún siendo compleja y estar compuesta por varios procesos que se han de aplicar uno después del otro, se pretende que sea equivalente a una interacción atómica. Es decir, que se realice de una sola vez.
- Estabilidad, o sea, que su nivel de fallos disminuye, en dependencia de la estabilidad que se requiera.
- Escalabilidad, es decir, su posibilidad de estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos, siendo capaz de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.

[15] Ídem referencia 10

- Es multiplataforma.[16]

Su uso es un poco restringido para aquellas personas que no pueden adquirirlo debido a su elevado precio, que es de varios miles de euros (según versiones y licencias). Un aspecto que ha sido criticado por algunos especialistas es la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.[17]

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS (*Relational Data Base Management System* traducido al español como Sistema Administrador de Bases de Datos Relacionales) con licencia libre como PostgreSQL, MySql o Firebird.[18]

SQLServer

Microsoft SQL Server es un SGBD basado en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

SQL Server es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial. El motor de la BD SQL Server ofrece almacenamiento más seguro y confiable tanto para datos relacionales como estructurados, lo que le permite crear y administrar aplicaciones de datos altamente disponibles.[19]

Entre sus características figuran:

- Soporte de transacciones.
- Gran estabilidad.
- Gran seguridad.

[16] WIKIPEDIA. *Oracle*, 9 de marzo de 2007]. Disponible en: http://es.wikipedia.org/wiki/Oracle
[17] Ídem referencia 13
[18] Ídem referencia 13
[19] SERVER, M. S. *Sitio de Microsoft*, 9 de marzo de 2007]. Disponible en: http://www.microsoft.com/spain/sql/productinfo/overview/what-is-sql-server.mspx

- Escalabilidad. [20]

En los últimos tiempos ha surgido una nueva versión del SQL Server llamada SQL Server 2005 la que incorpora una gran cantidad de características que no tenían versiones anteriores, como son: mejoras en la seguridad, restauración online que mejora la disponibilidad de SQL Server, ya que únicamente los datos que se están recuperando quedan como no disponibles. El resto de la base de datos permanece online y disponible. También se cuenta de mejoras en la replicación de las base de datos.

Una de sus más significativas desventajas es la condición de software propietario, y hoy la tendencia es a usar software libre por las libertades que ofrece. Además Microsoft SQL Server no es multiplataforma, pues sólo está disponible en Sistemas Operativos de Microsoft.[21]

MySQL

MySQL es un SGBD para bases de datos relacionales. Su condición de open source hace que se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido de manera positiva en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.[22]

Entre las características de MySQL se destacan:

- Rapidez en el tiempo de respuesta.
- Funciona en diferentes plataformas.
- Soporte a grandes BD.

A pesar de haber incorporado en su última versión las características que antes no tenía incluida, como es el caso de los trigger, procedimientos almacenados, vistas, consultas, subconsultas entre otras, aun existen restricciones que le impiden ser lo suficientemente óptimo.[23]

---

[20] WIKIPEDIA. *Microsoft SQL Server*, 11 de marzo de 2007]. Disponible en: http://es.wikipedia.org/wiki/Microsoft_SQL_Server
[21] Ídem referencia 17
[22] PÉREZ, J. M. *¿Qué es MySQL?*, 8 de marzo de 2007]. Disponible en: http://www.esepestudio.com/articulo/desarrollo-web/bases-de-datos-mysql/Que-es-MySQL.htm
[23] MYSQL. *Restricciones en características de MySQL*, 9 de marzo de 2007]. Disponible en: http://dev.mysql.com/doc/refman/5.0/es/restrictions.html

PostgreSQL

PostGreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. PostGreSQL es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99. Entre sus características principales se encuentran:

- Replicación (soluciones comerciales y no comerciales) que permiten la duplicación de bases de datos maestras en múltiples sitios de réplica.
- Presenta todas las características necesarias para gestionar y consultar la información almacenada en la BD, tales como vistas, trigger, procedimientos almacenados.
- Presenta un buen mecanismo de bloqueo para evitar el bloqueo innecesario de registros.
- Funciones de compatibilidad para ayudar en la transición desde otros sistemas menos compatibles con SQL.[24]

También PostgreSQL puede ser ampliado por el usuario de muchas maneras. Por ejemplo añadiendo nuevos:

- Tipos de datos
- Funciones
- Operadores
- Funciones Agregadas.
- Lenguajes procedurales. [25]

Y además por su condición de software libre y open source, puede ser usado, modificado y distribuido por todos de manera gratuita, para cualquier propósito, sea privado, comercial o académico.[26]

Entre las desventajas que posee PostgreSQL está, que consume gran cantidad de recursos, tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento. Es de dos a tres veces más lento que MySQL. El primer contacto con este gestor es un poco duro porque

[24] POSTGRESQL. *Advantages*, 8 de marzo de 2007]. Disponible en: http://www.postgresql.org/about/advantages
[25] ---. *What is PostgreSQL?*, 8 de marzo de 2007]. Disponible en: http://www.postgresql.org/docs/8.1/interactive/preface.html#INTRO-WHATIS
[26] Idem referencia 22

la sintaxis de algunos de sus comandos no es nada intuitiva. También resultan engorrosas las pequeñas variaciones que presenta este gestor en algunos de los tipos de datos que maneja.[27]

1.2.1 Herramientas CASE (*Computer Aided Software Engineering,* Ingeniería de Software Asistida por Ordenador)

Para realizar el diseño de la base de datos es necesario contar con alguna herramienta que permita modelarla. En este caso se utilizará la herramienta "Case Studio".

CaseStudio

CASE STUDIO es una herramienta profesional que permite diseñar BD, 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, PostgreSQL, Access, etc.) y permite además generar scripts SQL, aplicar procesos de retroingeniería (reverse engineering) a las BD, o sea, a partir del modelo de tablas llegar al modelo lógico. Además permite usar plantillas de diseño personalizables y crear detallados informes en HTML y RTF.[28]

1.3 Tecnologías a utilizar

Después de citar las características más significativas de los gestores más populares a nivel mundial, se realiza un resumen comparativo para elegir que gestor utilizar en la aplicación.

Tanto Oracle, como SQL Server son software propietario, por lo que estas opciones se desechan a pesar de lo potente que son.

PostgreSQL y MySQL cumplen con la característica de ser software libre.

MySQL es más rápido que PostgreSQL, y por eso es más usado donde la velocidad es más importante. PostgreSQL es reconocido por ser más robusto y poseer más recursos que MySQL. PostgreSQL ya tiene incorporado, probado y por lo tanto con menos probabilidad de fallos; los recursos incorporados en la

[27] PECOS, D. *PostGreSQL vs. MySQL* 11 de marzo de 2007]. Disponible en: http://www.netpecos.org/docs/mysql_postgres/index.html
[28] YUSTE, M. *Potente utilidad de modelado para varias bases de datos*, 7 de marzo de 2007]. Disponible en: http://software.elpais.com/ie/27636-CASE-Studio-2

última versión de MySQL. Un ejemplo de estos recursos son: las transacciones, disparadores (triggers), procedimientos almacenados, vistas y cláusulas de integridad (constraints).[29]

PostgreSQL, continua siendo más eficiente en varios aspectos, posee un sofisticado mecanismo de bloqueo (MVCC: Multi-Version Concurrency Control) con el cual evita los bloqueos innecesarios y proporciona un mejor tiempo de respuesta en ambiente de grandes volúmenes. Además, soporta tamaños ilimitados de registros, bases de datos y tablas, acepta varios tipos de sub-consultas, soporta más tipos de datos y cuenta con un buen mecanismo de FAILSAVE (seguridad contra fallas, por ejemplo falla repentina del sistema).[30]

Como resultado del análisis de los gestores más populares a nivel mundial se decide utilizar PostgreSQL ya que lo que se persigue es diseñar y construir una base de datos utilizando software libre por todos los beneficios y libertades que supone el software libre. Estas ventajas, garantizan que un grupo pueda aprovechar lo ya desarrollado por otros, ampliarlo, redistribuirlo y mejorarlo para beneficio de la comunidad entera, sin límites de fronteras, ni razas o credos. Las características de este gestor, expuestas anteriormente, demuestran que es idóneo para las BD grandes como las que requiere un hospital. Aunque la velocidad de respuesta que ofrece este gestor con BD relativamente pequeñas puede parecer un poco deficiente, esta misma velocidad la mantiene al gestionar BDs realmente grandes, cosa que resulta loable. Además es multiplataforma y posee una estabilidad y confiabilidad legendaria, tiene gran escalabilidad, es capaz de ajustarse a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta. También tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los SGBD de alto nivel, como puede ser Oracle.[31]

[29] DBRUNAS. *¿PostgreSQL o MySQL cuál escoger ?*, 11 de marzo de 2007]. Disponible en: http://www.dbrunas.com.ar/search.php?query=PostgreSQL+o+MySQL+cu%E1l+escoger+%3F&type=all&mode=search
[30] Ídem referencia 25
[31] Ídem referencia 24

La herramienta CASE para el modelado de la base de datos que se utilizará es el “Case Studio” ya que es una herramienta fácil de usar, sirve para construir aplicaciones grandes y permite la exportación del modelo de la base de datos para PostgreSQL que es el SGBD a utilizar.

Conclusiones

En este capítulo se realizó un estudio de los posibles SGBD a utilizar. Para ello se tuvieron en cuenta sus características, ventajas y desventajas. Como resultado del análisis realizado se determinó usar como gestor de base de datos PostgreSQL y para el modelado de la base de datos la herramienta Case Studio.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

Introducción

En este capítulo se trata de manera concreta la solución propuesta al objetivo del trabajo. Se definen los requisitos funcionales y no funcionales que se deben tener en cuenta para el desarrollo de la BD. Además se fundamenta la arquitectura que tendrá la BD que es una Arquitectura Centralizada. Se analiza y justifica la estrategia de integración que se tiene concebida, para integrar banco de sangre con otros módulos. Se realiza un análisis detallado de todas las tablas de la base de datos para un mejor entendimiento de la misma. Además se analizan unos cuantos aspectos a tener en cuenta para optimizar las consultas y proporcionar así, que sea mejor el tiempo de respuesta de las consultas y transacciones.

2.1 Estrategia de integración de la solución a otros módulos

La integración del sistema es a nivel de negocio. Por cuestiones de seguridad de la información, no está permitido que entre las BDs haya ningún tipo de comunicación a no ser a través del negocio.

El módulo de banco de sangre necesita información de otros módulos, que es necesaria para su funcionamiento. Se utiliza del módulo de Configuración la tabla cfg_pais, cfg_diagnostico, cfg_funcionario, y del módulo Inscripción - Admisión, la tabla ia_datos_persona.

Para obtener la información requerida se accede a la BD a través del negocio, realizando consultas para obtener la información que se necesite, y no directamente a la base de datos de ninguno de estos módulos, con el objetivo de proteger la información y mantener la integridad de la misma.

2.2 Arquitectura de la Base de Datos del Módulo de Banco de Sangre.

La base de datos del módulo de banco de sangre, se diseñó utilizando un modelo de BD relacional porque además de ser una decisión de los arquitectos, es el más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Como su nombre lo indica, este modelo se basa en el uso de relaciones, y a través de estas, la información puede ser recuperada o almacenada mediante consultas, que ofrecen una amplia flexibilidad y poder, para administrar la información.

Las "relaciones", no son más que tablas bidimensionales constituidas por filas (tuplas) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la BD. Cada

instancia es una tupla de la relación, mientras que los atributos de la relación representan las propiedades de la entidad.[32]

Por ejemplo, en la BD se tiene que representar las fichas de donación que se les hace a los donantes cada vez que realizan una donación. Aquí se puede definir una relación llamada "ficha_donacion", cuyos atributos describen las características de las fichas de donación (Tabla 1). Cada tupla de la relación "ficha_donacion" representará una ficha de donación concreta.

Tabla 1. Tabla representativa, correspondiente a una ficha de donación.

ficha_donacion						
nro_ficha	**fecha**	**id_persona**	**id_tipo_donante**	**Peso**	**Cant_sangredonada**	**hemo**
1	25/04/2007	1	1	60	2	15

Para distinguir una tupla de otra, se recurre al concepto de "llave primaria". En este caso la llave primaria de la relación esta constituida por varios atributos, como son nro_ficha, fecha. Los atributos de la llave primaria por razones de integridad no pueden tomar valores nulos, pues ya no permitirían identificar una tupla concreta en una relación.

Además del uso del modelo relacional, se sigue una arquitectura centralizada ya que existirá un servidor central en el que se encontrará la BD de Banco de Sangre. A dicho servidor central se conectarán cada uno de los puestos de trabajo. (Ver figura 1)

[32] HTMLPOINT. *El modelo relacional*, 7 de julio de 2007]. Disponible en: http://www.htmlpoint.com/sql/sql_03.htm

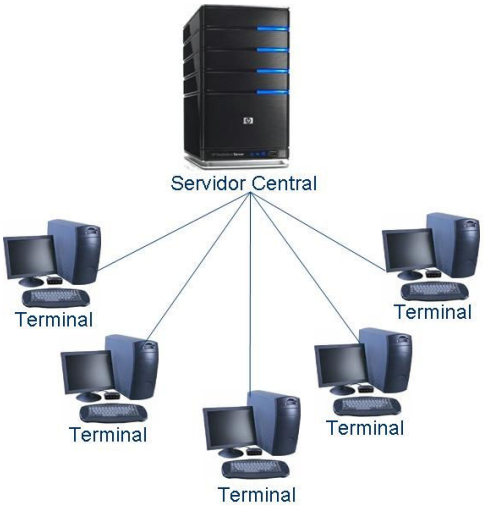

Figura 1. Arquitectura de la base de datos

2.3 Requisitos funcionales y no funcionales del sistema propuesto

Para lograr un buen funcionamiento de la BD, es necesario que se cumplan una serie de requerimientos funcionales y no funcionales. Estos constituyen condiciones que al cumplirlas se satisfacen las necesidades del usuario final.

2.3.1 Requisitos funcionales

Los requisitos o requerimientos funcionales, son aquellas condiciones o capacidades que debe cumplir la aplicación. La BD del módulo de banco de sangre debe permitir:

RF1 Registrar donante: Registrar los donantes en su primera donación guardando los datos generales correspondientes al mismo.

RF2 Actualizar donante: Permitir actualizar datos generales del donante en el caso que lo requiera (Ej. Cambio de dirección)

RF3 Eliminar donante: Permitir eliminar un donante determinado. (Ej. Donante que se detecte con infecciones de transmisión sexual, u otra enfermedad, etc.)

RF4 Buscar donante: Permitir buscar un donante determinado a través de una búsqueda básica por número de identidad, o avanzada teniendo en cuenta diversos criterios.

RF5 Imprimir Lista Donantes: El sistema deberá proporcionar un listado de donantes, atendiendo a los más diversos criterios de búsqueda.

RF6 Validar frecuencia mínima de donación: El sistema deberá permitir buscar un donante registrado y determinar el tiempo transcurrido entre la última donación y la fecha actual, comparar dicho tiempo con la frecuencia mínima de donación (3 meses) e informar atendiendo a dicho criterio si el donante está capacitado para donar, o no.

RF7 Mostrar ficha del donante: El sistema deberá brindar para cada donante, su ficha personal, con sus datos generales y un listado con todas sus visitas al Banco de Sangre, mostrando las Historias Clínicas correspondientes a cada visita.

RF8 Crear HC: El sistema deberá permitir crear nuevas historias clínicas para donantes registrados, omitiendo aquellos datos con carácter general (Nombre, Apellidos, sexo, etc.), solo editando los datos variables entre una donación y otra.

RF9 Eliminar HC: El sistema debe permitir eliminar las historias clínicas de los donantes que se determine que ya no son de utilidad para el sistema por la antigüedad o por un donante que cause baja del registro.

RF10 Actualizar HC: El sistema debe permitir que el técnico de laboratorio pueda editar la historia clínica del donante para registrar los datos en los que este interviene.

RF11 Solicitar Unidades por transfusión: El sistema debe permitir a los médicos realizar una solicitud de sangre para un paciente determinado.

RF12 Cancelar Solicitud: El sistema debe permitir al médico cancelar una solicitud realizada.

RF13 Confirmar Transfusión: El sistema debe permitir al médico informar que la solicitud realizada fue consumida por transfusión.

RF14 Reservar Unidades: El sistema debe permitir al técnico destinar unidades a las solicitudes realizadas, teniendo en cuenta la disponibilidad.

RF15 Liberación de unidades por transfusión: El sistema deberá permitir sustraer del depósito las unidades que sean liberadas por transfusión.

RF 16 Reintegración de unidades devueltas: El sistema debe permitir recibir de vuelta las unidades que hayan estado reservadas y no hayan sido consumidas, debido a una cancelación de solicitud.

RF17 Eliminar unidades por vencimiento: El sistema deberá permitir eliminar unidades del depósito por haber expirado.

RF18 Registro de las unidades recibidas de otros centros: Registra los datos de cada solicitud realizada a centros externos y el envío correspondiente a dicha solicitud.

RF19 Registro de las unidades enviadas a otros centros: Registrar los datos de las unidades enviadas a otros centros.

RF20 Mostrar disponibilidad paciente: El sistema deberá permitir consultar para un paciente dado, previamente a una transfusión: Grupo Sanguíneo, Factor RH, cantidad de sangre a suministrar, nombre del técnico de laboratorio.

RF21 Registro previo a la transfusión: Después de haber realizado el grupo sanguíneo de un paciente dado, se registra con antelación a la transfusión programada, el grupo sanguíneo, factor RH, así como otros datos de interés de dicho paciente.

RF22 Visualizar Reportes: El sistema deberá proporcionar reportes estadísticos diarios y mensuales correspondientes a información relacionada con las donaciones, transfusiones y pedidos.

RF23 Buscar Paciente: El sistema debe permitir buscar a un paciente atendiendo a diversos criterios.

RF24 Autenticar Usuario: El sistema debe exigirle a los usuarios autenticarse y darle acceso solamente a los elementos que tengan interacción con ellos, permitiendo la seguridad y fiabilidad de la información.

2.3.2 Requisitos no Funcionales

Los requerimientos no funcionales son aquellas cualidades que debe tener la aplicación. Existen diferentes clasificaciones de los requerimientos no funcionales. Por ejemplo, se pueden clasificar en: requerimientos de rendimiento, requerimientos de seguridad y privacidad, requerimiento de confiabilidad, entre otros. A continuación se mencionan las propiedades que debe tener la BD para su buen

funcionamiento.

Requerimientos de rendimiento

- El tiempo de respuesta de una petición al servidor deber rápido para la toma de decisiones.

Requerimientos de soporte

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

Requerimientos de seguridad y privacidad

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

Requerimientos de confiabilidad

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

Requerimientos de hardware

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

Restricciones en el diseño y la implementación

- Se utilizará un grupo de bibliotecas de clases definidas, dentro de ellas se encuentran Hermes7 para la comunicación HL7 y NpgSql para la conexión al servidor de bases de datos.
- El servidor debe contar con una plataforma de funcionamiento que utilice.
- La comunicación de de las terminales clientes con el servidor será a través de conexiones de fibra óptica.

2.4 Modelo de objetos o diagrama de clases persistentes obtenido a partir del diagrama de diseño.

El modelo de clases persistentes, es obtenido a partir del diagrama de diseño realizado por el analista. Este diagrama se utiliza para definir finalmente las entidades que son persistentes y por ende forman parte de las tablas de la BD. (Ver figura 2)

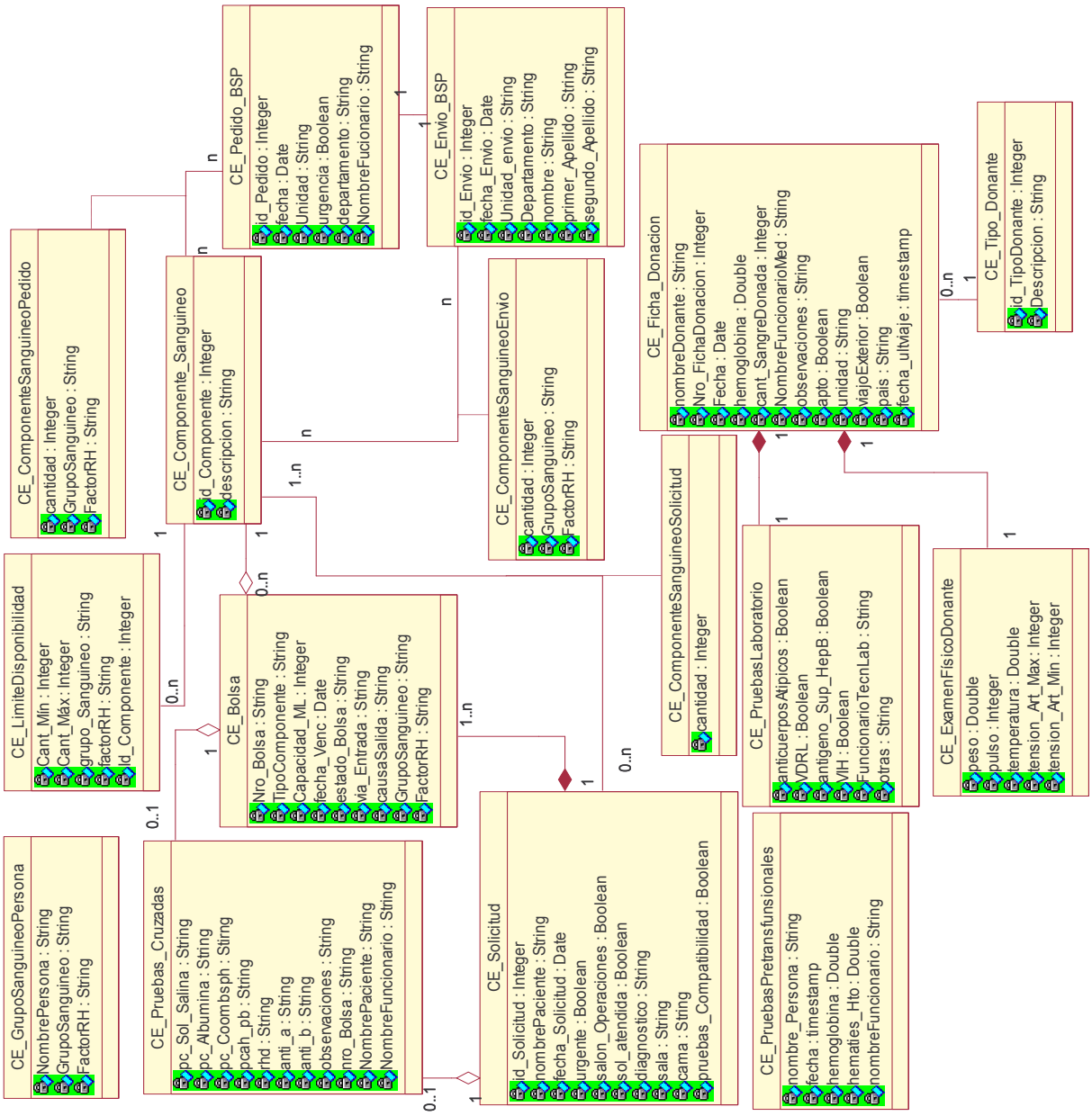

Figura 2. Diagrama de clases persistentes

2.5 Descripción de las clases

A continuación se describen las clases del diagrama de clases persistentes. Esta descripción es bastante general y se limita a dar a conocer cuáles son sus atributos y métodos con una breve descripción de estos últimos.

Tabla 2. Descripción de la entidad CE_Bolsa.

Nombre: CE_Bolsa	
Tipo de clase: Entidad	
Atributo	**Tipo**
Nro_Bolsa	String
GrupoSang	String
FactorRH	String
TipoComponente	String
Capacidad_ML	Integer
fecha_Venc	timestamp
estado_Bolsa	String
via_Entrada	String
causaSalida	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearBolsa()	Crea una bolsa.
Destruir()	Destruye un objeto bolsa.
SetNro_Bolsa(nrobolsa string)	Establece un valor al atributo Nro_Bolsa.
GetNro_Bolsa()	Se obtiene el valor que tiene el atributo Nro_Bolsa.
SetGrupoSang(gs string)	Establece un valor al atributo GrupoSang.
GetGrupoSang()	Se obtiene el valor que tiene el atributo GrupoSang.
SetFactorRH(factor string)	Establece un valor al atributo FactorRH.
GetFactorRH()	Se obtiene el valor que tiene el atributo FactorRH.

SetTipoComponente(comp string)	Establece un valor al atributo TipoComponente.
GetTipoComponente()	Se obtiene el valor que tiene el atributo TipoComponente.
Setcantidad_ML(cant integer)	Establece un valor al atributo cantidad_ML.
GetCapacidad_ML()	Se obtiene el valor que tiene el atributo cantidad_ML.
Setfecha_Venc(fechavenc timestamp)	Establece un valor al atributo fecha_Venc.
Getfecha_Venc()	Se obtiene el valor que tiene el atributo fecha_Venc.
Setestado_Bolsa(estado string)	Establece un valor al atributo estado_Reservacion.
Getestado_Bolsa()	Se obtiene el valor que tiene el atributo estado_Reservacion.
Setvia_Entrada(viaent string)	Establece un valor al atributo via_Entrada.
Getvia_Entrada()	Se obtiene el valor que tiene el atributo via_Entrada.
SetcausaSalida(csalida string)	Establece un valor al atributo causaSalida.
GetcausaSalida()	Se obtiene el valor que tiene el atributo causaSalida

Tabla 3 Descripción de la entidad CE_Componentes_Sanguineos

Nombre: CE_Componente_Sanguineo	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_Componente	Integer
descripcion	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearComponente()	Crea un objeto componente sanguíneo.
Destruir()	Destruye un objeto componente sanguíneo.
Setid_Componente(id_comp integer)	Establece un valor al atributo id_Componente.
Getid_Componente()	Se obtiene el valor que tiene el atributo id_Componente.
Set descripcion(desc string)	Establece un valor al atributo descripcion.
Getdescripcion()	Se obtiene el valor que tiene el atributo descripción.

Tabla 4. Descripción de la entidad CE_Pedido_BSP

Nombre: CE_Pedido_BSP	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_Pedido	Integer
fecha	timestamp
Unidad	String
urgencia	Boolean
departamento	String
NombreFucionario	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearPedido()	Crea un objeto pedido.
Destruir()	Destruye un objeto pedido.
Setid_Pedido(pid_pedido integer)	Establece un valor al atributo id_Pedido.
Getid_Pedido ()	Se obtiene el valor que tiene el atributo id_Pedido.
Setfecha(nro_ficha timestamp)	Establece un valor al atributo fecha.
Getfecha()	Se obtiene el valor que tiene el atributo fecha.
SetUnidad(pfecha string)	Establece un valor al atributo Unidad.
GetUnidad()	Se obtiene el valor que tiene el atributo Unidad.
Seturgencia(purgencia boolean)	Establece un valor al atributo urgencia.
Geturgencia()	Se obtiene el valor que tiene el atributo urgencia.
Setdepartamento(pdpto string)	Establece un valor al atributo departamento.
Getdepartamento()	Se obtiene el valor que tiene el atributo departamento.
SetNombreFucionario(pfunc integer)	Establece un valor al atributo NombreFucionario.
GetNombreFucionario()	Se obtiene el valor que tiene el atributo NombreFucionario.

Tabla 5. Descripción de la entidad CE_Envio_BSP

Nombre: CE_Envio_BSP	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_Envio	Integer
fecha_Envio	timestamp
Unidad_envio	String
Departamento	String
nombre	String
primer_Apellido	String
segundo_Apellido	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearEnvio()	Crea un objeto Envio.
Destruir()	Destruye un objeto envio.
Setid_Envio (id_env integer)	Establece un valor al atributo id_Envio.
Getid_Envio ()	Se obtiene el valor que tiene el atributo id_Envio.
Setfecha_Envio(fechaenv timestamp)	Establece un valor al atributo fecha_Envio.
Getfecha_Envio()	Se obtiene el valor que tiene el atributo fecha_Envio.
SetUnidad_envio(uni_env string)	Establece un valor al atributo Unidad_envio.
GetUnidad_envio()	Se obtiene el valor que tiene el atributo Unidad_envio.
SetDepartamento(dpto string)	Establece un valor al atributo Departamento.
GetDepartamento()	Se obtiene el valor que tiene el atributo Departamento.
Setnombre(nom string)	Establece un valor al atributo nombre.
Getnombre()	Se obtiene el valor que tiene el atributo nombre.
Setprimer_Apellido(p_apell string)	Establece un valor al atributo primer_Apellido.
Getprimer_Apellido()	Se obtiene el valor que tiene el atributo primer_Apellido.
Getprimer_Apellido()	Se obtiene el valor que tiene el atributo primer_Apellido.

Setsegundo_Apellido(s_apell string)	Establece un valor al atributo segundo_Apellido.

Tabla 6. Descripción de la entidad CE_Solicitud

Nombre: CE_Solicitud	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_Solicitud	Integer
nombrePaciente	String
fecha_Solicitud	Timestamp
urgente	Boolean
salon_Operaciones	Boolean
sol_atendida	Boolean
diagnostico	String
sala	String
cama	String
pruebas_Compatibilidad	Boolean
Para cada responsabilidad:	
Nombre:	Descripción:
CrearPrueba()	Crea un objeto solicitud.
Destruir()	Destruye un objeto solicitud.
Setid_Solicitud(pid_solic integer)	Establece un valor al atributo id_Solicitud.
Getid_Solicitud()	Se obtiene el valor que tiene el atributo id_Solicitud.
SetnombrePaciente(pnomb_pte string)	Establece un valor al atributo nombrePaciente.
GetnombrePaciente()	Se obtiene el valor que tiene el atributo nombrePaciente.
Setfecha_Solicitud(pfecha timestamp)	Establece un valor al atributo fecha_Solicitud.
Getfecha_Solicitud()	Se obtiene el valor que tiene el atributo fecha_Solicitud.
Seturgente(purg boolean)	Establece un valor al atributo urgente.
Geturgente()	Se obtiene el valor que tiene el atributo urgente.
Setsalon_Operaciones(psalon boolean)	Establece un valor al atributo salon_Operaciones.

Setsalon_Operaciones()	Se obtiene el valor que tiene el atributo salon_Operaciones.
Setsol_atendida(psol_atendida boolean)	Establece un valor al atributo sala sol_atendida.
Getsol_atendida()	Se obtiene el valor que tiene el atributo sol_atendida.
Setdiagnostico(pdiag string)	Establece un valor al atributo diagnostico.
Getdiagnostico()	Se obtiene el valor que tiene el atributo diagnostico.
Setsala(pidsala string)	Establece un valor al atributo sala.
Getsala()	Se obtiene el valor que tiene el atributo sala.
Setcama(pcama string)	Establece un valor al atributo cama.
Getcama()	Se obtiene el valor que tiene el atributo cama.
Setpruebas_Compatibilidad(pprueba boolean)	Establece un valor al atributo pruebas_Compatibilidad.
Getpruebas_Compatibilidad()	Se obtiene el valor que tiene el atributo pruebas_Compatibilidad.

Tabla 7. Descripción de la entidad CE_ComponenteSanguineo_PedidoBSP

Nombre: CE_ComponenteSanguineo_PedidoBSP	
Tipo de clase: Entidad	
Atributo	**Tipo**
cantidad_unidades	Integer
grupo_sanguineo	String
factor_rh	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearComponenteEnvio()	Crea un objeto que tiene como propiedad la cantidad por componentes sanguíneos que se solicitan a otros centros.
Destruir()	Destruye un objeto creado.
Setcantidad_unidades(cant integer)	Establece un valor al atributo cantidad_unidades.
Getcantidad_unidades()	Se obtiene el valor que tiene el atributo cantidad_unidades.

Setgrupo_sanguineo(gs string)	Establece un valor al atributo grupo_sanguineo
Getgrupo_sanguineo()	Se obtiene el valor que tiene el atributo grupo_sanguineo.
Setfactor_rh(factor string)	Establece un valor al atributo factor_rh.
Getfactor_rh()	Se obtiene el valor que tiene el atributo factor_rh.

Tabla 8. Descripción de la entidad CE_ComponenteSanguineo_EnvioBSP

Nombre: CE_ComponenteSanguineo_EnvioBSP	
Tipo de clase: Entidad	
Atributo	**Tipo**
cantidad_unidades	Integer
grupo_sanguineo	String
factor_rh	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearComponenteEnvio()	Crea un objeto que tiene como propiedad la cantidad por componentes sanguíneos que se envían a otros centros.
Destruir()	Destruye un objeto creado.
Setcantidad_unidades(cant integer)	Establece un valor al atributo cantidad_unidades.
Getcantidad_unidades()	Se obtiene el valor que tiene el atributo cantidad_unidades.
Setgrupo_sanguineo(gs string)	Establece un valor al atributo grupo_sanguineo
Getgrupo_sanguineo()	Se obtiene el valor que tiene el atributo grupo_sanguineo.
Setfactor_rh(factor string)	Establece un valor al atributo factor_rh.
Getfactor_rh()	Se obtiene el valor que tiene el atributo factor_rh.

Tabla 9. Descripción de la entidad CE_ComponenteSanguineo_Solicitud

Nombre: CE_ComponenteSanguineo_Solicitud	
Tipo de clase: Entidad	
Atributo	**Tipo**
cantidad_unidades	Integer
Para cada responsabilidad:	

Nombre:	Descripción:
CrearComponentePedido()	Crea un objeto que tiene como propiedad la cantidad por componentes sanguíneos que se solicitan.
Destruir()	Destruye un objeto creado.
Setcantidad_unidades(cant integer)	Establece un valor al atributo cantidad_unidades.
Getcantidad_unidades()	Se obtiene el valor que tiene el atributo cantidad_unidades.

Tabla 10. Descripción de la entidad CE_ExamenFísicoDonante

Nombre: CE_ExamenFísicoDonante	
Tipo de clase: Entidad	
Atributo	**Tipo**
peso	Double
pulso	Integer
temperatura	Double
tension_Art_Max	Integer
tension_Art_Min	Integer
Para cada responsabilidad:	
Nombre:	Descripción:
CrearExamenFisico()	Crea un objeto de tipo examen físico.
Destruir()	Destruye un objeto examen fisco.
Setpeso(ppeso double)	Establece un valor al atributo peso.
Getpeso ()	Se obtiene el valor que tiene el atributo peso.
Setpulso(ppulso integer)	Establece un valor al atributo pulso.
Getpulso()	Se obtiene el valor que tiene el atributo pulso.
Settemperatura(temp double)	Establece un valor al atributo temperatura.
Gettemperatura()	Se obtiene el valor que tiene el atributo temperatura.
Settension_Art_Max (tensionmax integer)	Establece un valor al atributo tension_Art_Max.
Gettension_Art_Max()	Se obtiene el valor que tiene el atributo tension_Art_Max.

Settension_Art_Min(nom string)	Establece un valor al atributo tension_Art_Min.
Gettension_Art_Min()	Se obtiene el valor que tiene el atributo tension_Art_Min

Tabla 11. Descripción de la entidad CE_Ficha_Donacion

Nombre: CE_Ficha_Donacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombreDonante	String
Nro_FichaDonacion	Integer
Fecha	timestamp
hemoglobina	Double
cant_SangreDonada	Integer
NombreFuncionario	String
observaciones	String
apto	Boolean
unidad	String
viajeExterior	Boolean
país	String
fecha_ultimo_viaje	Timestamp
Para cada responsabilidad:	
Nombre:	Descripción:
CrearFichaDonacion()	Crea un objeto ficha de donacion.
Destruir()	Destruye un objeto ficha de donacion.
SetnombreDonante(nom string)	Establece un valor al atributo nombreDonante.
GetnombreDonante()	Se obtiene el valor que tiene el atributo nombreDonante.
SetNro_FichaDonacion(nro_ficha integer)	Establece un valor al atributo Nro_FichaDonacion.
GetNro_FichaDonacion()	Se obtiene el valor que tiene el atributo Nro_FichaDonacion.
SetFecha(pfecha timestamptz)	Establece un valor al atributo Fecha.

GetFecha()	Se obtiene el valor que tiene el atributo Fecha.
Sethemoglobina(phemo double)	Establece un valor al atributo hemoglobina.
Gethemoglobina()	Se obtiene el valor que tiene el atributo hemoglobina.
Setcant_SangreDonada(pcant_sang integer)	Establece un valor al atributo cant_SangreDonada.
Getcant_SangreDonada()	Se obtiene el valor que tiene el atributo cant_SangreDonada.
SetNombreFuncionario(pfunc string)	Establece un valor al atributo NombreFuncionario.
GetNombreFuncionario()	Se obtiene el valor que tiene el atributo NombreFuncionario.
Setobservaciones(pobsrv string)	Establece un valor al atributo observaciones.
Getobservaciones()	Se obtiene el valor que tiene el atributo observaciones.
Setapto(papto boolean)	Establece un valor al atributo apto.
Getapto()	Se obtiene el valor que tiene el atributo apto.
Setunidad(puni string)	Establece un valor al atributo unidad.
Getunidad()	Se obtiene el valor que tiene el atributo unidad.
SetviajeExterior(ve boolean)	Establece un valor de verdadero o falso si el donante tuvo o no viajes al exterior respectivamente.
GetviajeExterior()	Se obtiene el valor que tiene el atributo viajeExterior.
Setpais(ppais string)	Establece un valor al atributo país, que es al último país al que ha viajado.
Getpais()	Se obtiene el valor que tiene el atributo pais.
Setfecha_ultimo_viaje(fecha timestamptz)	Establece un valor al atributo fecha_ultimo_viaje.
Getfecha_ultimo_viaje()	Se obtiene el valor que tiene el atributo fecha_ultimo_viaje.

Tabla 12. Descripción de la entidad CE_Prueba_Cruzada

Nombre: CE_Prueba_Cruzada	
Tipo de clase: Entidad	
Atributo	**Tipo**
NombrePaciente	String
nro_Bolsa	String
pc_Sol_Salina	String
pc_Albumina	String
pc_Coombsph	String
pcah_pb	String
rhd	String
anti_a	String
anti_b	String
observaciones	String
NombreFuncionario	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearPruebaCruzada()	Crea un objeto prueba cruzada.
Destruir()	Destruye un objeto creado.
SetNombrePaciente(nombpte integer)	Establece un valor al atributo NombrePaciente.
GetNombrePaciente()	Se obtiene el valor que tiene el atributo NombrePaciente.
Setnro_Bolsa(nrobolsa string)	Se establece un valor al atributo nro_Bolsa.
Getnro_Bolsa ()	Se obtiene el valor que tiene el atributo nro_Bolsa.
Setpc_Sol_Salina(pc_sol_sal string)	Establece un valor al atributo pc_Sol_Salina.
Getpc_Sol_Salina()	Se obtiene el valor que tiene el atributo pc_Sol_Salina.
Setpc_Albumina(pc_albm string)	Establece un valor al atributo pc_Albumina.
Getpc_Albumina()	Se obtiene el valor que tiene el atributo pc_Albumina.
Setpc_Coombsph(pc_coomb string)	Establece un valor al atributo pc_Coombsph.

Getpc_Coombsph()	Se obtiene el valor que tiene el atributo pc_Coombsph.
Setpcah_pb(pcah string)	Establece un valor al atributo pcah_pb.
Getpcah_pb()	Se obtiene el valor que tiene el atributo pcah_pb.
Setrhd(prhd string)	Establece un valor al atributo rhd.
Getrhd()	Se obtiene el valor que tiene el atributo rhd.
Setanti_a(pantia string)	Establece un valor al atributo anti_a.
Getanti_a()	Se obtiene el valor que tiene el atributo anti_a.
Setanti_b(pantib string)	Establece un valor al atributo anti_b.
Getanti_b()	Se obtiene el valor que tiene el atributo anti_b.
Setobservaciones(pobserv string)	Establece un valor al atributo observaciones.
Getobservaciones()	Se obtiene el valor que tiene el atributo observaciones.
SetNombreFuncionario(pid_func string)	Establece un valor al atributo id_Funcionario.
GetNombreFuncionario()	Se obtiene el valor que tiene el atributo id_Funcionario.

Tabla 13. Descripción de la entidad CE_PruebasLaboratorio

Nombre: CE_PruebasLaboratorio	
Tipo de clase: Entidad	
Atributo	**Tipo**
anticuerposAtipicos	Boolean
VDRL	Boolean
antigeno_Sup_HepB	Boolean
VIH	Boolean
otras	String
FuncionarioTecnLab	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearPruebaLaboratorio()	Crea un objeto prueba de laboratorio.
Destruir()	Destruye un objeto prueba de laboratorio.

SetanticuerposAtipicos(panticuerpos boolean)	Establece un valor al atributo anticuerposAtipicos.
GetanticuerposAtipicos()	Se obtiene el valor que tiene el atributo anticuerposAtipicos.
SetVDRL(pvdrl boolean)	Establece un valor al atributo VDRL.
GetVDRL()	Se obtiene el valor que tiene el atributo VDRL.
Setantigeno_Sup_HepB(pantigeno boolean)	Establece un valor al atributo antigeno_Sup_HepB.
Getantigeno_Sup_HepB()	Se obtiene el valor que tiene el atributo antigeno_Sup_HepB.
SetVIH(pvih boolean)	Establece un valor al atributo VIH.
GetVIH()	Se obtiene el valor que tiene el atributo VIH.
Setotras(potras string)	Establece un valor al atributo otras.
Getotras()	Se obtiene el valor que tiene el atributo otras.
SetFuncionarioTecnLab(pid_func string)	Establece un valor al atributo FuncionarioTecnLab.
GetFuncionarioTecnLab()	Se obtiene el valor que tiene el atributo FuncionarioTecnLab.

Tabla 14. Descripción de la entidad CE_PruebasPretransfunsionales

Nombre: CE_PruebasPretransfunsionales	
Tipo de clase: Entidad	
Atributo	**Tipo**
nombre_Persona	String
Fecha	timestamp
hemoglobina	Double
hematies_Hto	Double
nombreFuncionario	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearPrueba()	Crea un objeto prueba pretransfuncionales que son pruebas que siempre se le relizan al paciente para tener informacion

	acerca de como se encuetra su organismo para ser sometido a una operación.
Destruir()	Destruye un objeto pruebas pretransfuncionales.
Setid_Persona(pid_pers integer)	Establece un valor al atributo id_persona.
Getid_Persona()	Se obtiene el valor que tiene el atributo id_persona.
SetFecha(pfecha timestamp)	Establece un valor al atributo Fecha.
GetFecha()	Se obtiene el valor que tiene el atributo Fecha.
Sethemoglobina(phemo double)	Establece un valor al atributo hemoglobina.
Gethemoglobina()	Se obtiene el valor que tiene el atributo hemoglobina.
Sethematies_Hto(phematies double)	Establece un valor al atributo hematies_Hto.
Gethematies_Hto()	Se obtiene el valor que tiene el atributo hematies_Hto.
SetnombreFuncionario(pnomb string)	Establece un valor al atributo nombreFuncionario.
GetnombreFuncionario()	Se obtiene el valor que tiene el atributo nombreFuncionario.

Tabla 15. Descripción de la entidad CE_PruebasPretransfunsionales

Nombre: CE_GrupoSanguineoPersona	
Tipo de clase: Entidad	
Atributo	**Tipo**
NombrePersona	String
GrupoSanguineo	String
FactorRH	String
Para cada responsabilidad:	
Nombre:	Descripción:
CrearGrupoSanPersona()	Crea un objeto persona que tendrá el grupo sanguíneo y factor rh al que pertenece.
Destruir()	Destruye un objeto.
SetNombrePersona(pnom integer)	Establece un valor al atributo NombrePersona.
GetNombrePersona()	Se obtiene el valor que tiene el atributo NombrePersona.

SetGrupoSanguineo(pgs string)	Establece un valor al atributo GrupoSanguineo.
GetGrupoSanguineo()	Se obtiene el valor que tiene el atributo GrupoSanguineo.
SetFactorRH(pfactor string)	Establece un valor al atributo FactorRH.
GetFactorRH()	Se obtiene el valor que tiene el atributo FactorRH.

Tabla 16. Descripción de la entidad CE_LimiteDisponibilidad

Nombre: CE_LimiteDisponibilidad	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Componente	Integer
GrupoSanguineo	String
FactorRH	String
Cant_Máx	Integer
Cant_Mín	Integer
Para cada responsabilidad:	
Nombre:	Descripción:
CrearLimite()	Crea un objeto límite que establece el limite de un componente determinado teniendo en cuenta el grupo y factor al que pertenece.
Destruir()	Destruye un objeto.
SetId_Componente(pidcomp integer)	Establece un valor al atributo Id_Componente.
GetId_Componente()	Se obtiene el valor que tiene el atributo Id_Componente.
SetGrupoSanguineo(pgs string)	Establece un valor al atributo GrupoSanguineo.
GetGrupoSanguineo()	Se obtiene el valor que tiene el atributo GrupoSanguineo.
SetFactorRH(pfactor string)	Establece un valor al atributo FactorRH.
GetFactorRH()	Se obtiene el valor que tiene el atributo FactorRH.
SetCant_Máx(pcantmax integer)	Establece un valor al atributo Cant_Máx.
GetCant_Máx	Se obtiene el valor que tiene el atributo Cant_Máx.

SetCant_Min(pcantmin integer)	Establece un valor al atributo Cant_Min.
GetCant_Min()	Se obtiene el valor que tiene el atributo Cant_Min.

Tabla 17. Descripción de la entidad CE_Tipo_Donante

Nombre: CE_Tipo_Donante	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_TipoDonante	Integer
Descripcion	String
Para cada responsabilidad:	
Nombre:	Descripción:
CreartipoDonante()	Crea un objeto tipo de donante.
Destruir()	Destruye un objeto.
Setid_TipoDonante(pidtipo_dte integer)	Establece un valor al atributo id_TipoDonante.
Getid_TipoDonante()	Se obtiene el valor que tiene el atributo id_TipoDonante.
SetDescripcion(pdesc string)	Establece un valor al atributo Descripcion.
GetDescripcion()	Se obtiene el valor que tiene el atributo Descripcion.

2.6 Diseño de la base de datos.

A continuación, se hace referencia al diseño de la base de datos y a la descripción de las tablas que la componen.

2.6.1 Diagrama Entidad Relación de la BD.

La figura 3 muestra el diagrama entidad relación, que dará una visión del diseño de la BD. Aquí, como se explicó anteriormente, están presentes las entidades del modelo de clases persistentes. Además surgen otras entidades que surgen como resultado de la normalización, debido a anomalías de actualización. En el diagrama se podrá observar únicamente las entidades con sus respectivas llaves primarias y la relación que existe entre ellas, con el objetivo de que se puedan observar mejor las entidades y sus interrelaciones.

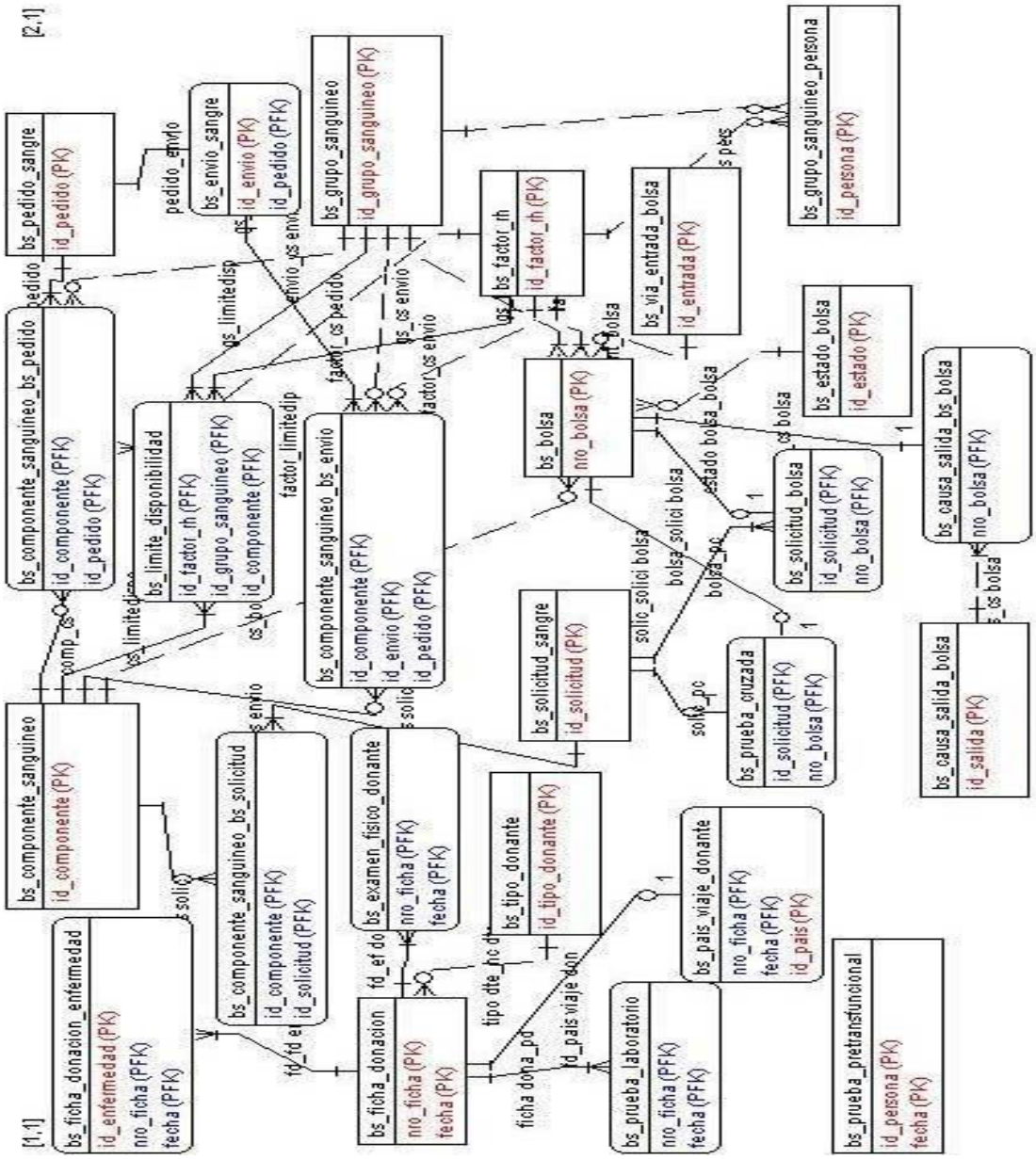

Figura 3. Diagrama Entidad Relación de la BD de Banco de Sangre.

2.6.2 Descripción de las tablas.

Se realizarán las descripciones detalladamente de todas las tablas que están presentes en el modelo relacional de la BD de banco de sangre.

Tabla 18. Detallada descripción de la relación bs_bolsa.

Nombre: bs_bolsa		
Descripción: Almacena las características y atributos que tiene la bolsa en la que se almacena la sangre o componentes sanguíneos.		
Atributo	**Tipo**	**Descripción**
nro_bolsa (PK)	Varchar	Serie que identifica cada una de las bolsas y las diferencia una de las otras.
cantidad_ml	Integer	Cantidad en mililitros, de sangre/componente sanguíneo que contiene la bolsa.
fecha_vencimiento	Timestamp	Fecha en la que se vence una bolsa.
id_estado(FK)	Integer	Identificador del estado en el que se encuentra la bolsa. Esta es una llave foránea que proviene de la tabla bs_estado_bolsa.
id_componente (FK)	Integer	Identificador del componente sanguíneo que posee la bolsa. Esta es una llave foránea que proviene de la tabla bs_componente_sanguineo.
id_entrada (FK)	Integer	Identificador de la vía de entrada de la bolsa. Mediante este se determina por que vía entra la bolsa. Esta llave foránea proviene de la tabla bs_via_entrada.
id_grupo_sanguineo(FK)	Integer	Identificador del grupo sanguíneo que tiene la bolsa. Esta llave foránea proviene de la tabla bs_grupo_sanguineo.
id_factor_rh(FK)	Integer	Identificador del factor rh que tiene la bolsa. Esta llave foránea proviene de la tabla bs_factor_rh.

Tabla 19. Detallada descripción de la relación bs_causas_salida_bolsa

Nombre: bs_causas_salida_bolsa		
Descripción: Tabla nomencladora que almacena todas las posibles vías por las cuales puede salir del depósito una bolsa.		
Atributo	**Tipo**	**Descripción**
id_salida (PK)	Serial	Identificador de las causas de salida.
descripción	Varchar	Todas las causas por las que puede salir una bolsa del depósito.

Tabla 20. Detallada descripción de la relación bs_causas_salida_bs_bolsa

Nombre: bs_causas_salida_bs_bolsa		
Descripción: Tabla que relaciona bs_bolsa con bs_causas_salida_bs_bolsa, de manera que en ella se almacena las bolsas y la causa de salida de cada una de las bolsas que salieron del depósito.		
Atributo	**Tipo**	**Descripción**
nro_bolsa (PFK)	Varchar	Identificador de la bolsa. Esta es una llave foránea y primaria a la vez que sirve para identificar la tabla que se describe. Ella proviene de la tabla bs_bolsa.
id_salida (FK)	Integer	Identificador de la causa por la cuál salió del depósito la bolsa. Esta llave foránea migra de la tabla bs_causas_salida_bolsa.

Tabla 21. Detallada descripción de la relación bs_componente_sanguineo

Nombre: bs_componente_sanguineo		
Descripción: Tabla nomencladora que almacena todos los posibles componentes en los que se puede descomponer la sangre.		
Atributo	**Tipo**	**Descripción**
id_componente (PK)	Serial	Identificador del componente sanguíneo.
descripción	Varchar	Nombre de todos los componentes sanguíneo existentes.

Tabla 22. Detallada descripción de la relación bs_componente_sanguineo_bs_envio

Nombre: bs_componente_sanguineo_bs_envio		
Descripción: Tabla resultante de la relación muchos a muchos que existe entre las tablas bs_componente_sanguineo y bs_envio_sangre permitiendo conocer que componentes sanguíneos y que cantidad de unidades, fueron enviados atendiendo a un determinado pedido.		
Atributo	**Tipo**	**Descripción**
id_componente (PFK)	Integer	Llave foránea y primaria a la vez que migra de la tabla bs_componente_sanguineo.
id_envio (PFK)	Integer	Llave foránea y primaria a la vez que migra de la tabla bs_envio_sangre.
id_pedido (PFK)	Integer	Llave primaria y foránea a la vez que migra de la tabla bs_componente_sanguineo_bs_envio
cant_unidades	Integer	La cantidad de unidades que se envió de un componente sanguíneo determinado.
id_grupo_sanguineo (FK)	Integer	Identificador del grupo sanguíneo al que pertenece el componente que se envía. Esta llave foránea migra de la tabla bs_grupo_sanguineo.
id_factor_rh (FK)	Integer	Identificador del factor rh que tiene el grupo sanguíneo del componente en cuestión. Esta llave foránea migra de la tabla bs_factor_rh.

Tabla 23. Detallada descripción de la relación bs_componente_sanguineo_bs_pedido

Nombre: bs_componente_sanguineo_bs_pedido		
Descripción: Tabla resultante de la relación muchos a muchos que existe entre las tablas bs_componente_sanguineo y bs_pedido_sangre permitiendo conocer que componentes sanguíneos y que cantidad de unidades, fueron solicitados en un determinado pedido.		
Atributo	**Tipo**	**Descripción**
id_componente (PFK)	Integer	Llave foránea y primaria a la vez que migra dela tabla bs_componente_sanguineo. Mediante esta se determina

		que componente sanguíneo fue solicitado.
id_pedido (PFK)	Integer	Identificador del pedido realizado. Esta llave foránea y primaria a la vez que migra de la tabla bs_pedido_sangre.
cant_unidades	Integer	La cantidad de unidades que se pidió de un componente sanguíneo.
id_grupo_sanguineo (FK)	Integer	Identificador del grupo sanguíneo al que pertenece el componente que se envía. Esta llave foránea migra de la tabla bs_grupo_sanguineo.
id_factor (FK)	Integer	Identificador del factor rh que tiene el grupo sanguíneo del componente en cuestión. Esta llave foránea migra de la tabla bs_factor_rh.

Tabla 24. Detallada descripción de la relación bs_componente_sanguineo_bs_solicitud

Nombre: bs_componente_sanguineo_bs_solicitud		
Descripción: Tabla resultante de la relación muchos a muchos entre las tablas bs_componente_sanguineo y bs_solicitud_sangre, en ella se almacenan, los componentes sanguíneos que se solicitaron en una solicitud determinada.		
Atributo	**Tipo**	**Descripción**
id_componente (PFK)	Integer	Es una llave foránea y a la vez primaria que migra de la tabla bs_componente_sanguineo. Mediante esta se puede determinar que componentes sanguíneos se solicitaron.
id_solicitud (PFK)	Integer	Identificador de la solicitud. Es una llave foránea y a la vez primaria que migra de la tabla bs_solicitud.
cant_unidades	Integer	Cantidad del componente que se solicitado.

Tabla 25. Detallada descripción de la relación bs_envio_sangre

Nombre: bs_envio_sangre		
Descripción: Almacena los datos del envío de sangre/componentes sanguíneos que le hace otro centro al banco de sangre.		
Atributo	**Tipo**	**Descripción**
id_envio (PK)	Serial	Identificador del envío.
Id_pedido (PFK)	Integer	Llave foránea y primaria a la vez que migra de la tabla bs_pedido_sangre. Es el identificador del pedido al que se le esta atendiendo.
fecha_envio	Timestamp	Fecha en la que se hizo el envío.
unidad_envio	Integer	Unidad que envía.
departamento	Varchar	Departamento que envía.
nombre	Varchar	Nombre de la persona responsable que hizo el envío.
primer_apellido	Varchar	Primer apellido de la persona que realizó un envío determinado.
segundo_apellido	Varchar	Segundo apellido de la persona que realizó un envío determinado.

Tabla 26. Detallada descripción de la relación bs_estado_bolsa

Nombre: bs_estado_bolsa		
Descripción: Tabla nomencladora que almacena todos los posibles estados en los que puede estar una bolsa.		
Atributo	**Tipo**	**Descripción**
id_estado (PK)	Serial	Identificador del estado de la bolsa.
descripcion	Varchar	Todos los estados en los que puede estar una bolsa.

Tabla 27. Detallada descripción de la relación bs_examen_fisico_donante

Nombre: bs_examen_fisico_donante		
Descripción: Almacena los resultados del examen físico que se le realiza a un donante, cada vez que se presenta a donar, antes de hacer la donación. Este se realiza con el objetivo de saber si se encuentra en condiciones de efectuar la donación. Es una entidad débil que depende de la entidad bs_ficha_donacion.		
Atributo	**Tipo**	**Descripción**
nro_ficha (PFK)	Integer	Número de la historia clínica del donante, que sirve de identificador de la tabla. Esta es una llave foránea y primaria a la vez que migra de la tabla bs_ficha_donacion.
fecha (PFK)	Timestamp	Fecha en la que se le hace la ficha de donación al donante. Esta es una llave foránea y primaria a la vez que migra de la tabla bs_ficha_donacion.
pulso_donante	Integer	Pulso que tiene el donante en ese momento.
peso_donante	Double Precision	Peso que tiene el donante en ese momento.
temperatura_donante	Double Precision	Temperatura que tiene el donante en ese momento.
tensión_arterial_max	Integer	Tensión arterial máxima que tiene el donante en ese momento.
tensión_arterial_min	Integer	Tensión mínima que tiene el donante en ese momento.

Tabla 28. Detallada descripción de la relación bs_factor_rh

Nombre: bs_factor_rh		
Descripción: Tabla nomencladora que almacena los diferentes factores rh que existen.		
Atributo	**Tipo**	**Descripción**
id_factor_rh (PK)	Serial	Identificador del factor rh.
descripcion	Varchar	Todos los factores rh que existen.

Tabla 29. Detallada descripción de la relación bs_ficha_donacion

Nombre: bs_ficha_donacion		
Descripción: Almacena datos propios de la ficha de donación que permiten identificar la persona a la que pertenece la donación realizada, así como el estado de salud en el que se encontraba en el momento que se presentó a donar.		
Atributo	**Tipo**	**Descripción**
nro_ficha (PK)	Integer	Número de la historia clínica del donante, que sirve de identificador de la tabla.
fecha (PK)	Timestamp	Fecha en la que se le hace la ficha de donación al donante. Junto con nro_ficha conforma el identificador de la ficha de la donación, ya que el número de la ficha no es suficiente para identificar cada una de ellas pues un mismo donante puede tener varias fichas de donación.
id_persona (FK)	Integer	Identificador de la persona. A través de esta se obtienen datos como el nombre, apellidos, dirección, etc. necesarios para conocer a que individuo pertenece una historia clínica determinada. Es una llave foranea cuya integridad referencial es a la tabla ia_datos_persona, del módulo Inscripción-Admisión.
id_tipo_donante (FK)	Integer	Identificador del tipo de donante. Es una llave foránea proveniente de la tabla bs_tipo_donante a través del cuál se obtiene el tipo de donante que es, por ejemplo si es familiar, por CDR, etc.
Unidad	Integer	Número de la unidad donde el donante realizará la donación.
hemoglobina	Varchar	La hemoglobina que tiene el donante.
apto	Boolean	Si después de los exámenes realizados el donante es útil o no.

cant_sangredonada	Integer	Cantidad de sangre que donó el donante.
viajes_exterior	Boolean	Si ha realizado viajes al exterior.
observaciones	Varchar	Alguna observación que haga el médico sobre algún percance que haya tenido el donante durante la donación, o algo que considere el médico que debe quedar registrado en la historia clínica del donante.
id_funcionario(FK)	Integer	Es el identificador del funcionario que realiza los exámenes y llena la historia clínica, en este caso, el médico. Su integridad referencial es a la tabla cfg_funcionario del módulo de Inscripción-Admisión.

Tabla 30. Detallada descripción de la relación bs_ficha_donacion_enfermedad.

Nombre: bs_ficha_donacion_enfermedad		
Descripción: Almacena las enfermedades más recientes que ha presentado el donante al momento de la donación. Es una tabla que debiera resultar de la relación muchos a muchos que existiera entre la tabla bs_ficha_donacion y cfg_enfermedades, en el caso que esta última estuviera en la BD de Banco de Sangre. Como esta es una relación imaginaria, por su permanencia en la BD del módulo Configuración en la BD de Banco de Sangre se almacena como otra de sus tablas, la relación que existe entre ellas dos.		
Atributo	**Tipo**	**Descripción**
id_enfermedad (PK)	Integer	Llave primaria y foránea a la vez que migra de la tabla cfg_enfermedades por lo que su integridad referencial es a dicha tabla, del módulo de configuración.
Fecha(PFK)	Timestamp	Fecha en la que el donante se presenta para realizar la donación.
nro_ficha(PFK)	Integer	Identificador de la ficha a la que pertenece las enfermedades registradas.

Tabla 31. Detallada descripción de la relación bs_grupo_sanguineo.

Nombre: bs_grupo_sanguineo		
Descripción: Tabla nomencladora que almacena todos los grupos sanguíneos existentes.		
Atributo	**Tipo**	**Descripción**
id_grupo_sanguineo (PK)	Serial	Identificador de cada uno de los grupos sanguíneos almacenados.
descripcion	Varchar	Nombre de todos los grupos sanguíneos existentes.

Tabla 32. Detallada descripción de la relación bs_grupo_sanguineo_persona

Nombre: bs_grupo_sanguineo_persona		
Descripción: Almacena el grupo sanguíneo al que pertenece una persona determinada.		
Atributo	**Tipo**	**Descripción**
id_persona (PK)	Integer	Identificador de la persona a la que se le registra el grupo sanguíneo y factor rh. Es la llave primaria de la tabla que se describe. Su integridad referencial es a la tabla ia_datos_persona del módulo Inscripción – Admisión.
id_grupo_sanguineo(FK)	integer	Identificador del grupo sanguíneo a la que pertenece la persona. Esta es una llave foránea que migra de la tabla bs_grupo_sanguineo.
id_factor_rh(FK)	Integer	Identificador del factor rh que tiene una persona determinada. Esta es una llave primaria que proviene de la tabla bs_factor_rh.

Tabla 33. Detallada descripción de la relación bs_limite_disponibilidad

Nombre: bs_limite_disponibilidad		
Descripción: En esta tabla se registran los límites de la cantidad de sangre por componente sanguíneo grupo y factor, que debe existir en el banco de sangre.		
Atributo	**Tipo**	**Descripción**
id_componente (PFK)	Integer	Llave foránea y primaria a la vez que migra desde la tabla

		bs_componente_sanguineo.
Id_grupo_sanguineo(PFK)	Integer	Llave primaria y foránea a la vez que migra de la tabla bs_grupo_sanguineo.
Id_factor_rh(PFK)	Integer	Llave primaria y foránea que migra de la tabla bs_factor_rh.
cant_min	Integer	Cantidad mínima que debe existir en el banco de sangre, de un componente determinado atendiendo al grupo sanguíneo y factor rh que presenta.
cant_max	Integer	Cantidad maxima que debe existir en el banco de sangre, de un componente determinado atendiendo al grupo sanguíneo y factor rh que presenta.

Tabla 34. Detallada descripción de la relación bs_pais_viaje_donante

Nombre: bs_pais_viaje_donante		
Descripción: Almacena el último país al que viajó el donante si ha realizado algún viaje al exterior. Es una entidad débil que depende de la tabla bs_ficha_donacion y cfg_pais.		
Atributo	**Tipo**	**Descripción**
id_pais (PK)	Integer	Identificador del país al que ha viajado el donante. Esta es una llave que migra desde la entidad cfg_pais, perteneciente a la BD del módulo Configuración. Por lo tanto su integridad referencial es a esa tabla.
nro_ficha(PFK)	Integer	Llave primaria y foránea que migra de la tabla bs_ficha_donacion.
fecha (PFK)	Timestamp	Llave primaria y foránea que migra de la tabla bs_ficha_donación.

Tabla 35. Detallada descripción de la relación bs_pedido_sangre

Nombre: bs_pedido_sangre		
Descripción: Almacena los datos del pedido de sangre/componentes sanguíneos que le hace el hospital a otros centros.		
Atributo	**Tipo**	**Descripción**
id_pedido (PK)	Integer	Identificador del pedido realizado.
fecha_pedido	Timestamp	Fecha en la que se hace el pedido.
unidad	Integer	Unidad que hace el pedido.
urgente	Boolen	Si es de carácter urgente o no.
departamento	Varchar	Nombre del departamento que hace el pedido.
id_funcionario(FK)	Integer	Es el identificador del funcionario que realiza el pedido. A través de este se determina el nombre y apellidos del funcionario que hizo el pedido. La integridad referencial de este atributo es a la tabla cfg_funcionario del módulo configuración.

Tabla 36. Detallada descripción de la relación bs_pruebas_cruzadas

Nombre: bs_pruebas_cruzadas		
Descripción: Almacena los resultados de las pruebas cruzadas realizadas al paciente y a la bolsa minutos antes de ser transfundido. Entidad débil que depende de la tabla bs_solicitud_sangre y bs_bolsa.		
Atributo	**Tipo**	**Descripción**
id_solicitud (PFK)	Integer	Identificador de la solicitud. Es una llave foránea y primaria que proviene de la tabla bs_solicitud_sangre.
nro_bolsa (PFK)	Varchar	Número de la bolsa con la que se le hicieron las pruebas cruzadas al donante. Es una llave foránea y primaria proveniente de la tabla bs_bolsa.
pruebac_solsalina	Varchar	Resultado de esta prueba.
pruebac_albumina	Varchar	Resultado de esta prueba.
prueba_coombsph	Varchar	Resultado de esta prueba.

pcah_pb	Varchar	Resultado de esta prueba.
rhd	Varchar	Resultado de esta prueba.
anti_a	Varchar	Resultado de esta prueba.
anti_b	Varchar	Resultado de esta prueba.
observaciones	Varchar	Observaciones.
id_funcionario(FK)	Integer	Es un Identificador del funcionario que realiza las pruebas, en este caso, el técnico. Su integridad referencial es a la tabla cfg_funcionario del módulo de Configuración.

Tabla 37. Detallada descripción de la relación bs_pruebas_laboratorio

Nombre: bs_pruebas_laboratorio		
Descripción: Tabla que registra, los resultados de las pesquisas realizadas a la sangre de los donantes para determinar si finalmente es útil o no la donación. Entidad débil que depende de la tabla bs_ficha_donacion.		
Atributo	**Tipo**	**Descripción**
nro_ficha (PFK)	Integer	Número de la ficha a la que pertenecen los resultados. Es una llave foránea y primaria a la vez que migra de la tabla bs_ficha_donacion.
fecha (PFK)	Timestamp	Al igual que el atributo anterior este, migra de la tabla bs_ficha_donacion, y forma parte de la llave primaria de esta relación.
anticuerpos_atipicos	Varchar	Resultado de esta prueba, realizada a la sangre donada.
vdrl	Varchar	Resultado de esta prueba.
antigeno_superficie	Varchar	Resultado de esta prueba.
hepatitis_b	Varchar	Resultado de esta prueba.
vih	Varchar	Resultado de esta prueba.
otras	Varchar	Otras enfermedades encontradas en los exámenes

		realizados.
id_funcionario(FK)	Integer	Es el identificador del funcionario que realiza los exámenes. A través de este se determina el nombre y apellidos del funcionario responsable que realizó la pesquisa de la sangre. La integridad referencial de este atributo es a la tabla cfg_funcionario del módulo de Configuración.

Tabla 38. Detallada descripción de la tabla bs_prueba_pretransfunsional

Nombre: bs_prueba_pretransfunsional		
Descripción: Tabla que recoge el resultados de los análisis que debe realizarse un paciente antes de ser operado		
Atributo	**Tipo**	**Descripción**
id_persona (PK)	Integer	Es el identificador de la persona, forma parte de la llave primaria de esta tabla. Tiene su integridad referencial a la tabla ia_datos_persona del módulo de Inscripción – Admisión. Mediante esta se determina a que persona pertenecen los resultados de los análisis.
fecha (PK)	Timestamp	Fecha en que el paciente se realizó los análisis. Junto a la id_persona conforma la llave primaria de esta relación.
hemoglobina	Varchar	Resultado del análisis.
hematíes_hematocristo	Varchar	Resultado del análisis.
id_funcionario(FK)	Integer	Es el identificador del funcionario que realizó los análisis. Su integridad referencial es a la tabla cfg_funcionario del módulo Configuración.

Tabla 39. Detallada descripción de la relación bs_solicitud_bolsa

Nombre: bs_solicitud_bolsa		
Descripción: Tabla que almacenan las bolsas solicitadas para un paciente determinado.		
Atributo	**Tipo**	**Descripción**
id_solictud (PFK)	Integer	Identificador de la tabla bs_solicitud.
nro_bolsa (PFK)	Integer	Identificador de la tabla bs_bolsa.

Tabla 40. Detallada descripción de la relación bs_solicitud_sangre

Nombre: bs_solicitud_sangre		
Descripción: Almacena la solicitud hecha, con antelación, al banco de sangre de la cantidad de sangre necesaria para operar a determinado paciente.		
Atributo	**Tipo**	**Descripción**
id_solicitud (PK)	Integer	Identificador de la solicitud.
fecha_solicitud	Timestamp	Fecha para la cual se hace la solicitud.
caracter_urgente	Boolean	Si es urgente o no la solicitud.
solicitud_atendida	Boolean	Si fue atendida o no la solicitud.
diagnostico	Varchar	Diagnóstico emitido por el médico, este describe el motivo por el cual se realiza la solicitud de sangre.
sala	Varchar	Nombre de la sala donde se encuentra el paciente para el cual se realiza la solicitud, en caso de que este ingresado.
cama	Varchar	Nombre de la cama que ocupa el paciente para el cual se hizo la solicitud, en caso de estar ingresado.
salón_operaciones	Boolean	Si la solicitud es para un salón de operaciones o no.
pruebas_compatilidad	Boolean	Si la solicitud lleva pruebas cruzadas o no.
id_persona(FK)	Integer	Identificador de la persona a través del cual se obtiene la persona para la que se realizó la solicitud. Su integridad referencial es a la tabla ia_datos_persona perteneciente al módulo Configuración

Tabla 41. Detallada descripción de la relación bs_tipo_donante

Nombre: bs_tipo_donante		
Descripción: Tabla nomencladora que almacena todos los posibles tipos de donantes que existen.		
Atributo	**Tipo**	**Descripción**
id_tipo_donante (PK)	Integer	Es un identificador de los tipos de donantes que existen.
descripción	Varchar	Nombre de los tipos de donante que existen.

Tabla 42. Detallada descripción de la relación bs_via_entrada_bolsa

Nombre: bs_via_entrada_bolsa		
Descripción: Tabla nomencladora que almacena todas las posibles vías por las cuales puede entrar al depósito una bolsa.		
Atributo	**Tipo**	**Descripción**
id_entrada (PK)	Integer	Identificador de la vía por la que entró la bolsa.
descripción	Varchar	Vías por las que pude entrar una bolsa.

2.7 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, 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 BD tomando en cuenta el uso que se le va a dar, esto se refiere a cuáles transacciones están definidas sobre la BD y cuál es su frecuencia de ejecución. También se toman en consideración las restricciones de tiempo de respuesta de cada transacción.

En esta sección, sólo se tratará el análisis de optimización de consultas.

Las actualizaciones de la BD de Banco de sangre, 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.

Existen varios aspectos en los que se puede mejorar el rendimiento. Entre ellos se pueden citar los siguientes:

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.

3. En términos de rendimiento es preferible evaluar una condición de igualdad numérica que una condición de igualdad sobre cadenas de caracteres.

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

5. El uso de vistas pueden causar una ejecución ineficiente de consultas. Muchas veces la ejecución de consultas sobre las tablas base es más eficiente.[33]

Para la realización de las consultas y procedimientos almacenados se han tenido en cuenta cada uno de los aspectos planteados, evitando hacer uso de la cláusula DISTINCT debido a lo costosa que resulta su ejecución. Todas las comparaciones a evaluar se han realizado teniendo en cuenta que es mucho más factible evaluar una condición de igualdad numérica que una sobre cadena de caracteres.

Se utilizan vistas en casos considerables, donde se consulta a muchas tablas a la misma vez. Lo más factible en estos casos es hacer la unión de estas tablas en una vista, para luego acceder a la vista desde

[33] MOTA, S. A. *Entonación de Bases de Datos*, Abril-Julio 2005. [25 de mayo de 2007]. Disponible en: http://www.bd.cesma.usb.ve/ci5851/apuntes3.pdf

el procedimiento o consulta que se vaya a realizar. Tal es el caso de la vista bs_bolsa_view que se realiza con el objetivo de agrupar en una sola tabla virtual, por llamarla de alguna manera, todos los campos necesarios a mostrar de las tablas bs_componente_sanguineo, bs_bolsa, bs_via_entrada, bs_estado_bolsa, de manera que a la hora de mostrarlos solo se haga la consulta a esta vista y no a las cuatro tablas, lo que optimiza el tiempo de respuesta.

Otro aspecto en la optimización es el uso de procedimientos almacenados para el acceso a los datos. Esto optimiza considerablemente el tiempo de respuesta debido a que el procedimiento almacenado es un código precompilado que requiere de menos tiempo de ejecución que una consulta.

Conclusiones

Como se pudo apreciar, se cuenta con una base de datos centralizada en cuyo servidor central hay una copia de la BD del módulo de banco de sangre, y a este están conectados todos los puestos de trabajo. La base de datos consta de 25 tablas. De ella 16 tablas provienen del modelo de clases persistentes, obtenido del diagrama de clases de diseño realizado por el analista. Dentro de estas 16 tablas existen 2 que son nomencladoras o codificadoras. De las restantes 9 tablas, 1 es generada de una relación muchos a muchos, 5 surgen como tablas nomencladoras, con el fin de homogenizar la información y las otras 3 surgen como resultado de cuestiones de normalización, con el objetivo de eliminar anomalías de actualización. Para el acceso a la información se ha realizado más de 100 procedimientos almacenados y algunas vistas que facilitan la consulta de la información.

CAPÍTULO 3: VALIDACIÓN DEL DISEÑO

Introducción

En este capítulo se realizará una validación teórica del diseño de la BD realizado. Esta validación tiene el propósito de analizar y comprobar si el diseño realizado es el correcto. Para evaluarlo se tendrán en cuenta una serie de aspectos que son recomendables para un buen diseño además de características deseables en la BD.

3.1 Integridad

La integridad no es más que reglas que deben cumplir los datos almacenados para garantizar que son correctos.[34]

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

Una llave primaria es uno o un conjunto de atributos que permiten identificar a una tupla de otra, de manera única en cualquier momento. Estas son de gran importancia pues constituyen el mecanismo de direccionamiento a nivel de tuplas en el modelo relacional. Es decir, es el único modo de acceder a una tupla específica.

Por otro lado una llave foránea de una relación es un atributo que hace referencia a una llave primaria de otra relación; lo que da pie a que una relación pueda tener varias llaves foráneas.

Y por último un valor nulo es un valor que está fuera de la definición de cualquier dominio el cual permite que el atributo quede sin valor alguno.

Las dos reglas de integridad tienen que ver precisamente con los conceptos antes mencionados y son:

- Reglas de integridad de Relaciones: Estas son las encargadas de asegurar que ninguna clave primaria de una entidad, puede tener valores nulos y siempre deberá ser única.[35] Esto se ve

[34] ANDRÉS, M. M. M. *Reglas de integridad*, 6 de junio de 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node52.html

[35] VÍCTOR HUGO DORANTES GONZÁLEZ, F. M. L., JOSÉ NEIF JURY FABRE. *Curso de Bases de Datos y PostgreSQL*, 6 de junio de 2007]. Disponible en: http://es.tldp.org/Tutoriales/NOTAS-CURSO-BBDD/notas-curso-BD/

reflejado en la BD en la clave de la entidad bs_bolsa, que es el número de la bolsa, cuyo valor es único para cada bolsa y nunca va a ser nulo.

Además dentro de este tipo de regla se encuentran las restricciones de comprobación que se crean en la definición de la tabla que sirven para chequear algún atributo en específico.[36]

En la BD del módulo Banco de Sangre se tienen en cuenta algunas restricciones de comprobación para algunos atributos que de no cumplirse estarían incumpliendo con la integridad relacional:

- La tensión arterial mínima debe tomar valores menores o iguales que la tensión arterial máxima.
- La cantidad máxima por componentes que debe existir en el banco de sangre, información que se recoge en la tabla bs_componente_sanguineo, debe tomar valores mayores o iguales que la cantidad mínima por componentes.

- Reglas de integridad referencial.

  Es aquella donde las claves ajenas, o llaves foráneas, de una tabla hija se tienen que corresponder con la clave primaria, o llave primaria, de la tabla padre con la que se relaciona. Esto se pone de manifiesto en la BD, en la tabla bs_bolsa que tiene varias llaves foráneas, tal es el caso de id_componente. El valor de esta clave en la entidad, bs_bolsa, que es en este caso la entidad hija, tiene que corresponderse con la llave primaria de la tabla bs_componente_sanguineo, que es la entidad padre con la que se relaciona.

  Lo que, en resumen, se debe tener bien presente, es que la BD no debe contener valores de llaves ajenas sin concordancia.[37]

  La integridad referencial es una propiedad de la BD que resulta muy útil e importante. Gracias a ella se garantiza que una entidad siempre se relacione con otras entidades válidas, o sea, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, ni existan datos perdidos.

[36] Ídem referencia 35
[37] Ídem referencia 35

En la BD de Banco de Sangre, existen llaves foráneas de otras tablas que no son propias de banco de sangre, o sea que residen en otra base de datos, y dichas llaves son representadas en la BD de Banco de Sangre como atributos normales. En este caso se encuentran id_funcionario, id_diagnostico e id_pais que son llaves foráneas que provienen de las tablas cfg_funcionario, cfg_diagnostico y cfg_pais respectivamente, del módulo Configuración. De la integridad referencial de este atributo, se debe encargar la capa de negocio, debido a que la integración es a nivel de negocio como se explicó en el capítulo anterior. Esto se cumple para todas las llaves foráneas que migran de tablas que no están presentes en la propia BD de Banco de Sangre.

Existe además, otro tipo de integridad, denominado integridad de dominio que no es más que aquella que están asociadas a un dominio de datos específicos y son aplicadas a cada columna definida sobre el dominio. En otras palabras, estas restricciones permiten restringir los valores que puede tomar un atributo.[38]

La BD de Banco de Sangre tiene varias reglas de integridad de dominio que validan los datos proporcionados a un atributo o columna de la tabla:

1. bs_tension_arterial_dom: Este dominio chequea que la tensión arterial no tome valores que excedan a 360 ni estén por debajo de 0.
2. bs_temperatura_dom: Chequea que la temperatura del donante no tome valores que estén fuera del rango establecido que es de 35º a 42º.
3. bs_cantidasangre_donada_dom: Que no permite la cantidad de sangre donada no exceda los 700 mililitros.
4. bs_hemoglobina_dom: Chequea que la hemoglobina esté en el rango de valores de 0 a 160. Teniendo en cuenta que el valor puede ser en distintas medidas.
5. bs_positivo_dom: Chequea que los valores sean mayores que cero.

[38] IBÁÑEZ, M. J. O. *Integridad en Sistemas de Bases de Datos Relacionales* 6 de junio de 2007]. Disponible en: http://dis.um.es/~mjortin/ibd_temas/ibd_t3_integr.pdf

Estos son dominios que establecen los valores que pueden tomar los siguientes atributos respectivamente:

1. tensión_arterial_max y tensión_arterial_min perteneciente a la tabla bs_examen_fisico.
2. temperatura_donante perteneciente a la tabla bs_examen_fisico.
3. cantidad_sangredonada perteneciente a la tabla bs_ficha_donacion.
4. hemoglobina, cuyo valor se recoge tanto en la tabla bs_ficha_donacion como en la tabla bs_prueba_pretransfunsional.
5. En las tablas bs_bolsa, bs_componente_sanguineo_bs_envio, bs_componente_sanguineo_bs_pedido, bs_componente_sanguineo_solicitud, bs_ficha_donacion, bs_limite_disponibilidad, existe un atributo cantidad cuyo valor no debe ser negativo. Esto lo chequea el dominio bs_positivo_dom.

3.2 Normalización de la base de datos

La normalización de la BD es el proceso mediante el cual se dividen las relaciones insatisfactorias distribuyendo sus atributos en relaciones mucho más pequeñas que presentan características deseables. El proceso de normalización es aplicable a los modelos entidad relación y a los modelos relacionales. Este proceso permite, una vez terminado, que en las BDs relacionales no existan datos repetidos en las tablas, que se proteja la integridad de los datos y además contribuyen a evitar los problemas de actualización de los datos en las tablas.[39]

Para normalizar una BD es necesario conocer y aplicar una serie de reglas que van a permitir que el diseño realizado sea correcto. Existen 6 Formas Normales que garantizan un buen diseño de la BD. Básicamente se considera una BD bien diseñada si se encuentra en la Tercera Forma Normal (3ra FN). La BD del módulo de Banco de sangre se llevará hasta 3ra FN.

La Primera Forma Normal (1ra FN) plantea que: “Una relación está en primera forma normal (1FN) si los valores para cada atributo de la relación son atómicos”. Esto quiere decir, que cada atributo

[39] WIKIPEDIA. Normalización de una base de datos, 22 de mayo de 2007]. Disponible en: http://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_una_base_de_datos

sólo puede pertenecer a un dominio y que tiene un valor único para cada fila. Esta forma normal se definió para prohibir los atributos multivaluados, compuestos y sus combinaciones. [40]

Cuando una relación no está en 1FN, se divide en otras relaciones, repartiendo sus atributos entre las resultantes. Normalmente la idea es eliminar el atributo que viola la 1FN de la relación original y colocarlo en una relación aparte junto con la clave primaria de la relación de partida.[41]

Las relaciones poseen una serie de propiedades muy importantes. Entre ellas se encuentra, que no existen tuplas duplicadas y que cada tupla contiene un valor para cada atributo.

Por lo anterior planteado se concluye que todas las relaciones de la base de datos del módulo Banco de Sangre se encuentran en primera forma normal. Con esta se garantiza una atomicidad de todos los atributos, evitando que existan atributos con más de un valor para cada una de las tuplas.

La segunda forma normal plantea que: “Una relación está en segunda forma normal (2FN) si está en la 1FN y todos los atributos no clave dependen de la clave completa y no sólo de una parte de esta”.[42]

Este paso sólo se aplica a relaciones que tienen claves compuestas, es decir, que están formadas por más de un atributo. Si un esquema de relación no está en 2da FN, se le puede normalizar a varias relaciones en 2da 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.[43]

En la BD de Banco de Sangre, existen varias tablas que tienen más de una llave primaria. Esto se debe a que con sólo una de ellas no es suficiente para identificarse. Ellas son: bs_ficha_donacion, bs_prueba_pretransfunsional. La tabla bs_ficha_donacion tiene como llave el número de la ficha y la fecha. Esta llave compuesta es necesaria debido a que el número de la ficha de donación se resetea cada determinado tiempo, es por ello que para identificarla se hace necesaria la fecha. De esta clave dependen totalmente todos los atributos de la ficha de donación por lo que se considera que está en 2FN. Esto significa que si se elimina uno de los atributos que componen la llave, la dependencia pierde validez. Por

[40] Ídem referencia 32
[41] Ídem referencia 29
[42] Ídem referencia 29
[43] Ídem referencia 29

otro lado la tabla bs_prueba_pretransfuncional tiene como llave primaria el identificador de la persona a la que pertenece, más la fecha en la que se realizó las pruebas. Esta clave es necesaria debido a que no basta conocer el identificador de la persona para identificar un resultado determinado, pues una misma persona puede haberse realizado varios análisis de este tipo. Por lo tanto, para diferenciar uno de otro es necesario conocer de quién es y cuándo se lo hizo. De esta clave dependen funcionalmente de forma total todos los atributos que pertenecen a la relación. De esta forma se garantiza que bs_prueba_pretransfuncional está en 2FN.

Las demás entidades tienen una clave primaria simple y como se encuentran en 1FN pues se garantiza que ya están en 2FN.[44]

La tercera forma normal plantea que: "Una relación está en tercera forma normal (3FN) 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 dependencia funcional F: X → Y, X es una clave.

Se puede observar que si una relación está en 3FN, está también en segunda forma normal, sin embargo lo inverso no siempre es cierto.[45]

En la BD no existen dependencias transitivas y por lo tanto no se incumple con la tercera forma normal. Después de haber realizado un análisis de todas las entidades y haber determinado que todas las tablas están en tercera forma normal se concluye que la BD está en 3ra FN.

Una vez concluida la normalización, se decide homogenizar la información creando tablas nomencladoras o codificadoras. Estas tablas evitarán que algún usuario introduzca en la BD algún error ortográfico a la hora de teclear los datos que en algún momento quiera guardar. La introducción de un error en la BD por el usuario final, traería como consecuencia en un futuro que haya algún problema a la hora realizar alguna búsqueda. De ahí la importancia que el usuario escriba la menor cantidad de texto posible para entrarlo a la BD.

---

[44] TELEFORMACIÓN. *Normalización de bases de datos relacionales*, 7 de junio de 2007]. Disponible en: http://teleformacion.uci.cu/file.php/45/CLASES/Conferencias/Conferencia_4/normalizacion_de_BD_relacionales.pdf

[45] NORICK, R. Y. DISEÑO DE BASES DE DATOS RELACIONALES, 27 de mayo de 2007]. Disponible en: http://usuarios.lycos.es/cursosgbd/UD4.htm

Después de la anterior explicación, se justifica el surgimiento de 5 nuevas tablas cuyos nombres son: bs_via_entrada_bolsa, bs_grupo_sanguineo, bs_factor_rh, bs_estado_bolsa, bs_causas_salida_bolsa. Con el surgimiento de estas tablas se producen cambios en las tablas base. Estos cambios se deben a que ya los atributos que hacen referencia a estas relaciones estarían en sus respectivas relaciones como llaves foráneas. Un ejemplo de lo anterior descrito se ve reflejado en la tabla bs_bolsa se tiene como atributos una serie de elementos como son: via_Entrada, estado_Bolsa, TipoComponente, Factor, GrupoSanguineo, causaSalida; que ya no serían atributos sino llaves foráneas: id_entrada, id_estado, id_componente, id_factor_rh, id_grupo_sanguineo, id_salida respectivamente; que migran de las respectivas tablas nomencladoras. Y así sucesivamente ocurre con los atributos que tengan que ver con la vía de entrada de la bolsa, grupo sanguíneo, factor rh, estado de la bolsa y causas de la salida de la bolsa. (Ver anexo 1)

Existen además otros atributos que pueden ser cambiados por llaves foráneas pero su integridad referencial es a otras tablas que no son precisamente tablas de la BD de Banco de Sangre. En este caso se encuentra el NombreDonante perteneciente a la entidad CE_Ficha_Donacion, que puede ser sustituido por el identificador de la tabla ia_datos_persona que se encuentra en la BD del módulo Inscripción – Admisión. En la misma situación se encuentran una serie de atributos que serían sustituidos por el identificador de la tabla correspondiente. De esta manera se evita que existan datos repetidos y se garantiza que se reutilice lo que ha sido creado por otros, proporcionando así, que el trabajo sea más rápido y eficiente. (Ver anexo 2)

Los cambios realizados en las diferentes entidades, que ahora son tablas de la BD, traen como consecuencia que se tengan aún, anomalías en la BD. Lo anterior se pone de manifiesto en la tabla bs_ficha_donacion. En esta tabla se recoge el dato que corresponde a si el donante tuvo o no viajes al exterior. Si la respuesta es positiva entonces se registra el último país al que viajó y la fecha en que lo hizo. Para registrar el país se hace uso de la llave foránea id_pais por lo antes explicado y ello trae como consecuencia, que con su presencia en la tabla bs_ficha_donacion, se obligue a la hora de insertar una ficha de donación, a que el donante a la que pertenece tenga viajes al exterior. Esto no es lo óptimo, pues puede haber donantes que nunca hayan realizado ningún viaje. Debido a esta situación, se decide crear una nueva tabla que contenga como clave primaria la llave del país y la llave compuesta de la ficha de

donación unido al atributo fecha_ultimo_viaje, que pertenece en estos momento a la tabla bs_ficha_donacion, evitando así que aporte un nulo a esta tabla a la que pertenece. (Ver anexo 3 A)

Se presenta otra anomalía de actualización en la tabla bs_bolsa, pues esta se encuentra relacionada con la tabla bs_solicitud con una relación de uno a muchos, de la solicitud a la bolsa, debido a que en una solicitud, puede pedirse una o varias bolsas. Por este motivo la llave de solicitud mira como llave foránea para la tabla bs_bolsa. La anomalía consiste en que se obliga a que cada bolsa que se inserte, esté solicitada, o sea, que le corresponda una solicitud. Esto no es cierto pues pueden existir bolsas que no estén solicitadas. Por esta razón, surge una nueva tabla que guarde como clave primaria la llave de bs_solicitud y la llave de bs_bolsa. Con ello se consigue almacenar las solicitudes y las bolsas pertenecientes a estas. (Ver anexo 3 B)

Un problema similar surge con las tablas bs_causas_salida_bolsa y bs_bolsa entre las que debe existir una relación de uno a muchos, donde la llave de la primera debe migrar como llave foránea para la segunda. Aquí existirá una anomalía a la hora de insertar una bolsa en la tabla bs_bolsa, pues dicha llave foránea exigirá un valor con el que se estaría diciendo que toda bolsa al insertarse debe tener una causa de salida. Por tal motivo se crea una nueva tabla que se llamará bs_causa_salida_bs_bolsa. (Ver Anexo 3 C).

De esta manera queda la BD del módulo Banco de Sangre constituida por 25 tablas, en 3FN y libre de anomalías de actualización.

3.3 Análisis de redundancia de información.

El diseño de una BD debe ser realizado cuidadosamente, procurando cumplir con permitir un fácil acceso a la información. El sistema debe facilitar un alto rendimiento, una buena velocidad de respuesta y una gran consistencia de los datos. Al diseñar una BD se debe tener en cuenta que la información almacenada ocupará irremediablemente un espacio en memoria. Por ello es de vital importancia eliminar la posibilidad de almacenar datos repetidos ya que adicionalmente podrían conducir a que exista inconsistencia en la

información. A este almacenamiento de información repetida se le conoce como redundancia de la información.[46]

Eliminar la redundancia de la información es uno de los objetivos principales de la normalización. Esto no es más que eliminar, o por lo menos tener una mínima redundancia de información en la BD.

Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. No obstante en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias. En ocasiones resulta mejor perder en redundancia por ganar en simplicidad de las consultas y lograr un mejor tiempo de respuesta.[47]

Después de lo anterior visto, se plantea que es necesario analizar si es óptimo o no eliminar totalmente la redundancia, pues puede valorarse el caso de ganar en rendimiento, aunque se sacrifique la capacidad de almacenamiento.

En la BD de Banco de Sangre no existe ninguna redundancia de la información. No ha sido realmente necesario duplicar información y sacrificar la propiedad deseable de logar que no exista información duplicada. Gracias a ello se garantiza, que la información no sea inconsistente.

3.4 Análisis de la seguridad de la base de datos.

La seguridad de los datos se asocia frecuentemente con la integridad de los mismos, pero ambos conceptos son bastante diferentes. La seguridad se refiere a la protección de los datos contra su revelación, su alteración o su destrucción no autorizadas, mientras que la integridad se refiere a la precisión o validez de esos datos. En otras palabras esto significa que la seguridad garantiza que los usuarios tengan permisos de hacer las cosas que están tratando de llevar a cabo y la integridad involucra asegurar que lo que se este tratando de realizar sea correcto.

Los bancos de sangre manipulan información altamente sensible con relación a la salud de diversas personas. Por tal motivo es necesario garantizar una seguridad adecuada en la BD, que es donde se almacenará dicha información. De esta forma se logra que ningún individuo ajeno a ella o sin los permisos

[46] AVENDAÑO, D. E. P. *Bases de datos*, 4 de junio de 2007]. Disponible en: http://www.cs.buap.mx/~dpinto/bd/
[47] WIKIPEDIA. *Sistema de gestión de base de datos*. Disponible en: http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos

requeridos acceda a esta información, ya sea para consultarla o modificarla. Así se garantiza que la información almacenada no se corrompa y sea confiable, tanto para cualquier diagnóstico que de ella dependa, como para el cuidado del estado físico y sentimental de la persona a la que pertenece.

La protección de los datos deberá llevarse a cabo contra fallos físicos, fallos lógicos y fallos humanos (intencionados o no). Estos fallos alteran indebidamente los datos, los corrompen con lo que la base de datos ya no puede servir a los fines para los que fue creada.[48]

La seguridad de la base de datos esta implementada en varios niveles, algunos de ellos son:

1. 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 super usuario de PostgreSQL.
2. A cada usuario de PostgreSQL se le asigna un nombre de usuario y (opcionalmente) una contraseña. Por defecto, los usuarios no tienen permiso de escritura a bases de datos que no hayan creado.
3. Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse en base a esos grupos.[49]

Por lo general no se crean usuarios a los que se les otorga los permisos pertinentes. Es mucho más práctico crear roles a los que se les otorgan permisos determinados. A estos roles se les asignan usuarios en dependencia de la funcionalidad que tendrá. Una vez asignados los usuarios que se determinen a los roles creados, cada usuario tendrá los permisos que tenga ese rol al que pertenece.

La BD consta del super usuario postgres. Este por los privilegios otorgados, es capaz de ejecutar cualquier funcionalidad de la BD, tales como: insertar, eliminar, actualizar, modificar las tablas, y es quien tiene los permisos para provocar cambios drásticos en la BD.

[48] SÀNCHEZ., M. A. D. S. Seguridad de las bases de datos, 30 de mayo de 2007]. Disponible en: http://www.monografias.com/trabajos26/seguridad-base-datos/seguridad-base-datos2.shtml#segur
[49] POSTGRESQL, E. *Guia del administrador de postgres* 30 de mayo de 2007]. Disponible en: http://es.tldp.org/Postgresql-es/web/navegable/admin/security.html

También se cuenta con el rol usuario. En este rol estarán todos los usuarios que tendrán solo los permisos de actualizar, insertar, seleccionar y eliminar información de la BD, además de permitir la ejecución de los procedimientos almacenados para manipular la información. Se creó dentro de este rol el usuario gehosbs, que tendrá los mismos permisos que el rol usuario que es a quien pertenece. Este usuario debe ser correctamente manipulado, pues debe ser asignado sólo a las personas autorizadas a realizar cambios en la información. Para aquellos usuarios que sólo tendrán permisos de lectura, se cuenta con otro rol, denominado usuario limitado que sólo tendrá permisos a consultar información de la BD, y no por el contrario de escribir o modificarla, y mucho menos realizar algún cambio drástico en la BD, como eliminar alguna tabla. Por el momento el usuario perteneciente a este rol es gehosbsr.

Otra medida de seguridad es la realización de copias de la base de datos. Esto se realiza por si ocurre alguna incidencia y se pierden los datos almacenados en ella. Hacer copias de seguridad significa realizar una copia de los ficheros de datos a cualquier otro medio (disco duro, unidades de cinta, cd-rom, etc.) que permita recuperarlos. Se realizarán copias de seguridad diarias de manera automática en el servidor central para cada una de las bases de datos que allí se encuentren.

Además se pueden establecer a través del gestor, las conexiones al servidor de los puestos de trabajo, lo que posibilita el control de las máquinas o puestos de trabajo que se conectarán al servidor.

3.5 Trazabilidad de la acciones.

La trazabilidad es un sistema que deja rastro de las acciones que el usuario realiza con los datos. Por ejemplo, si un usuario elimina un registro, queda constancia del registro que se eliminó, quién lo eliminó y cuándo. De esta forma, puede hacerse un seguimiento de todas las acciones que se han realizado sobre un registro concreto, o todas las acciones que ha realizado un usuario determinado. De esta forma se puede conformar un historial de las acciones sobre los datos.

Es necesario llevar un control de cada una de las acciones que se realizan en la BD. De esta manera se tiene una detallada descripción de lo que ocurre con cada uno de los elementos que componen la BD para poder saber en un determinado momento donde estuvo el error que se cometió, quién y cuándo lo cometió. Así se hace mucho más fácil saber por donde empezar a restablecer el sistema en caso de algún fallo.

El Sistema Gestor de Base de Datos PostgreSQL puede ser configurado para la obtención de estos registros. Esta opción es habilitada en el fichero de configuración de este gestor, denominado “postgres.conf”.

Es importante señalar que en estos logs, el usuario que realizó la operación que aparece registrado es el usuario que se utilizó para conectarse a la BD, o sea, son los usuarios creados a nivel de BD.

Conclusiones

En este capítulo se realizó una validación teórica de la base de datos diseñada donde se tuvieron en cuenta una serie de aspectos con el fin validar el diseño realizado. Se analizó la integridad referencial que guarda cada tabla con las demás con las que se relaciona. Se llevó a cabo la normalización de la base de datos llevándola hasta la tercera forma normal garantizando que no exista redundancia en la BD. Además se crearon roles en la BD por motivos de seguridad, para proteger la información que se ha almacenado. Con estos roles se controlan los permisos de cada usuario que se conecte a la BD. También se guardan las trazas de todas las operaciones que se realizan en la BD. De esta forma se garantiza que se registren cada una de las modificaciones que se realicen en ella.

CONCLUSIONES

En este trabajo, se analizó la necesidad existente en los bancos de sangre de crear una base de datos que permitiera almacenar la información manejada, proporcionando así, que el trabajo sea más eficiente. De esta manera, el acceso a la información registrada se hace más fácil, sencillo, y por lo tanto la calidad de los servicios prestados es mejor.

Para dar respuesta a las necesidades del banco de sangre se diseñó satisfactoriamente una base de datos, que almacena toda la información manejada en estas instituciones. Para ello:

- ✓ Se hizo uso del gestor de base de datos PostgreSQL en su versión 8.2 para la elaboración de la BD. Esta es una potente herramienta que permitió contar con una base de datos eficiente capaz de satisfacer las necesidades de estas instituciones.
- ✓ Se obtuvieron los artefactos de la metodología RUP que describen la BD, tal es el caso de el diagrama de clases persistentes que se obtiene a partir del diagrama de clases del diseño que realiza el analista.
- ✓ Se diseñó y estructuró satisfactoriamente el modelo relacional brindando así una visualización de la BD.
- ✓ Se implementaron satisfactoriamente los procedimientos almacenados que permiten el acceso a los datos.
- ✓ Por último se realizó una validación teórica del diseño realizado.

De esta forma se da por cumplido el objetivo planteado. Esto favorecerá el incremento en la eficiencia de los servicios ofrecidos en estas instituciones.

RECOMENDACIONES

Una vez concluido el trabajo y teniendo en cuenta las experiencias adquiridas durante el desarrollo del mismo se recomienda a la dirección del proyecto:

- ✓ Hacer una validación funcional de la BD donde se realicen pruebas de carga que permitan analizar el rendimiento de la base de datos.
- ✓ Continuar con el estudio de los diferentes procesos que se llevan a cabo en los bancos de sangre con el objetivo de conocer más sobre estos y que este conocimiento proporcione mejoras en la manera de gestionar y almacenar la información.

BIBLIOGRAFÍA

1. ANDRÉS, M. M. M. Reglas de integridad, 6 de junio de 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node52.html
2. AVENDAÑO, D. E. P. Bases de datos, 4 de junio de 2007]. Disponible en: http://www.cs.buap.mx/~dpinto/bd/
3. BIOLINUX, G. Sistema Informático Hospitalario Abierto., 12 de marzo de 2007]. Disponible en: http://www.openhis.com.ar/blog/
4. CÉSAR, C. R. BOLETIN - Instituto de Información Científica y Tecnológica, 12 de marzo de 2007]. Disponible en: http://www.idict.cu/UserFiles/File/Boletines/Novedades/boletin02_0105/Inicio020105.html
5. CHÁVEZ", I. N. D. C. I. Banco de Sangre, 27 de noviembre de 2006]. Disponible en: http://www.cardiologia.org.mx/incic/ban_sangre/
6. Connolly, T.; Begg C.; Strachan, A. Database Systems: A Practical Approach to Design, Implementation and Management. 2nd edition. Addison-Wesley, 1998.
7. Date C. J. Introducción a los sistemas de Bases de Datos, primera parte. La Habana, Editorial Félix Varela, 2003. 326 págs.
8. Date C. J. Introducción a los sistemas de Bases de Datos, segunda parte. La Habana, Editorial Félix Varela, 2003. 173 págs.
9. Date C. J. Introducción a los sistemas de Bases de Datos, tercera parte. La Habana, Editorial Félix Varela, 2003. 381 pág.
10. DBASUPORT. Características de MySQL, 8 de marzo de 2007]. Disponible en: http://www.dbasupport.com.mx/index.php?option=com_content&task=view&id=118&Itemid=30
11. DBRUNAS. PostgreSQL o MySQL cuál escoger?, 11 de marzo de 2007]. Disponible en: http://www.dbrunas.com.ar/search.php?query=PostgreSQL+o+MySQL+cu%E1l+escoger+%3F&type=all&mode=search

12. DCSOFT. DELPHYN Blood Bank Data Management System 14 de marzo de 2007]. Disponible en: http://www.dcsoftintl.com/MedicApps.asp
13. Elmasri, R.; Navathe, S.B. Fundamentos de Sistemas de Bases de Datos. 3ª Edición. Addison-Wesley, Pearson Educación, 2002.
14. Elmasri, R.; Navathe, S.B. Sistemas de bases de datos. Conceptos fundamentales. 2ª Edición. Addison-Wesley Iberoamericana, 1997.
15. FLORINA GATICA LARA, F. J. F. P., ANA MARÍA HERNÁNDEZ LÓPEZ. Sistema de Información Hospitalaria, 23 de mayo de 2007]. Disponible en: http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf
16. GALIANO, J. L. M. Generadores de código, 6 de junio de 2007]. Disponible en: http://moga.awardspace.com/wp-content/uploads/2007/03/gencod.pdf
17. GARCÍA, R. F. S. Revista de Ciencias Médicas LA INFORMATICA MEDICA EN CUBA., 25 de noviembre de 2006]. Disponible en: http://www.cpicmha.sld.cu/hab/vol6_2_00/hab070200.htm
18. GROUP, V. xBlood - Sistema para el manejo de bancos de sangre, 5 de febrero de 2007]. Disponible en: http://www.virtus.com.mx/xblood/index.html
19. Guia Lan TIMES de SQL, James R. Goff, Paul N. Weinberg, Mc Graw Hill, 1998.
20. HTMLPOINT. El modelo relacional, 7 de julio de 2007]. Disponible en: http://www.htmlpoint.com/sql/sql_03.htm
21. IBÁÑEZ, M. J. O. Integridad en Sistemas de Bases de Datos Relacionales 6 de junio de 2007]. Disponible en: http://dis.um.es/~mjortin/ibd_temas/ibd_t3_integr.pdf
22. KATIA CARABALLOSO GRANADO, Y. T., MABEL CHAU LEY Las Nuevas Tecnologías de la Información y las Comunicaciones en las Bibliotecas Universitarias: presentación de un caso, 25 de noviembre de 2006]. Disponible en: http://www.sociedadelainformacion.com/Principal/n10articulos07/Las%20Nuevas%20Tecnologias.pdf
23. LATORILLA, E. Sitio de care2x, 12 de marzo de 2007]. Disponible en: http://www.care2x.org/

24. MOTA, S. A. Entonación de Bases de Datos, Abril-Julio 2005. [25 de mayo de 2007]. Disponible en: http://www.bd.cesma.usb.ve/ci5851/apuntes3.pdf

25. MYSQL. Las principales características de MySQL, 9 de marzo de 2007]. Disponible en: http://dev.mysql.com/doc/refman/5.0/es/features.html

26. ---. Restricciones en características de MySQL, 9 de marzo de 2007]. Disponible en: http://dev.mysql.com/doc/refman/5.0/es/restrictions.html

27. NORICK, R. DISEÑO DE BASES DE DATOS RELACIONALES, 27 de mayo de 2007]. Disponible en: http://usuarios.lycos.es/cursosgbd/UD4.htm

28. PECOS, D. PostGreSQL vs. MySQL 11 de marzo de 2007]. Disponible en: http://www.netpecos.org/docs/mysql_postgres/index.html

29. PÉREZ, J. M. ¿Qué es MySQL?, 8 de marzo de 2007]. Disponible en: http://www.esepestudio.com/articulo/desarrollo-web/bases-de-datos-mysql/Que-es-MySQL.htm

30. POSTGRESQL. Advantages, 8 de marzo de 2007]. Disponible en: http://www.postgresql.org/about/advantages

31. ---. What is PostgreSQL?, 8 de marzo de 2007]. Disponible en: http://www.postgresql.org/docs/8.1/interactive/preface.html#INTRO-WHATIS

32. POSTGRESQL, E. Guia del administrador de postgres 30 de mayo de 2007]. Disponible en: http://es.tldp.org/Postgresql-es/web/navegable/admin/security.html

33. PostgreSQL vs. MySQL vs. Comercial Databases, its all about what you need http://www.devx.com/dbzone/Article/20743/1954?pf=true

34. SÀNCHEZ., M. A. D. S. Seguridad de las bases de datos, 30 de mayo de 2007]. Disponible en: http://www.monografias.com/trabajos26/seguridad-base-datos/seguridad-base-datos.shtml

35. SERVER, M. S. Sitio de Microsoft, 9 de marzo de 2007]. Disponible en: http://www.microsoft.com/spain/sql/productinfo/overview/what-is-sql-server.mspx

36. ---. Sitio de Microsoft. Las 30 características principales de SQL Server 2005 Disponible en: http://www.microsoft.com/spain/sql/productinfo/features/top30features.mspx

37. TELEFORMACIÓN. Normalización de bases de datos relacionales, 7 de junio de 2007]. Disponible en: http://teleformacion.uci.cu/file.php/45/CLASES/Conferencias/Conferencia_4/normalizacion_de_BD_relacionales.pdf

38. VÍCTOR HUGO DORANTES GONZÁLEZ, F. M. L., JOSÉ NEIF JURY FABRE. Curso de Bases de Datos y PostgreSQL, 6 de junio de 2007]. Disponible en: http://es.tldp.org/Tutoriales/NOTAS-CURSO-BBDD/notas-curso-BD/

39. WIKIPEDIA. Microsoft SQL Server, 11 de marzo de 2007]. Disponible en: http://es.wikipedia.org/wiki/Microsoft_SQL_Server

40. ---. Normalización de una base de datos, 22 de mayo de 2007]. Disponible en: http://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_una_base_de_datos

41. ---. Oracle, 9 de marzo de 2007]. Disponible en: http://es.wikipedia.org/wiki/Oracle

42. ---. Sistema de gestión de base de datos. Disponible en: http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos

43. YUSTE, M. Potente utilidad de modelado para varias bases de datos, 7 de marzo de 2007]. Disponible en: http://software.elpais.com/ie/27636-CASE-Studio-2

ANEXOS

Anexo 1. Tabla representativa de atributos que fueron sustituidos por llaves foráneas y tablas de donde migran dichas llaves.

Entidad afectad	Atributo	Llave foránea	Integridad referencial
CE_Bolsa	GrupoSanguineo	id_grupo_sanguineo	bs_grupo_sanguineo
	FactorRH	id_factor_rh	bs_factor_rh
	TipoComponente	id_componente	bs_componente_sanguineo
	estado_Bolsa	id_estado	bs_estado_bolsa
	via_Entrada	id_entrada	bs_via_entrada_bolsa
	causaSalida	id_salida	bs_causa_salida_bolsa
CE_ComponenteSanguineoEnvio	GrupoSang	id_grupo_sanguineo	bs_grupo_sanguineo
	FactorRH	id_factor_rh	bs_factor_rh
CE_ComponenteSanguineoPedido	GrupoSang	id_grupo_sanguineo	bs_grupo_sanguineo
	FactorRH	id_factor_rh	bs_factor_rh
CE_Ficha_Donacion	TipoDonante	id_tipo_donante	bs_tipo_donante
CE_GrupoSanguineo_Persona	GrupoSang	id_grupo_sanguineo	bs_grupo_sanguineo
	Factor	id_factor_rh	bs_factor_rh
CE_LimiteDisponibilidad	GrupoSang	id_grupo_sanguineo	bs_grupo_sanguineo
	FactorRH	id_factor_rh	bs_factor_rh

Anexo 2. Tabla representativa de atributos que son sustituidos por llaves foráneas que provienen de tablas que no pertenecen a bases de datos de otros módulos.

Entidades afectadas	Atributos	Llave foránea	Integridad referencial
CE_Ficha_Donacion	NombreDonante	id_persona	ia_datos_persona del módulo Inscripción – Admisión.
	NombreFuncionario	id_funcionario	cfg_funcionario del módulo Configuración.

	TipoDonante	id_tipo_donante	bs_tipo_donante
	pais	id_pais	cfg_pais del módulo Configuración.
CE_Prueba_Cruzada	NombreFuncionario	id_funcionario	cfg_funcionario del módulo Configuración.
CE_PruebaLaboratorio	Funcionario	id_funcionario	cfg_funcionario del módulo Configuración.
CE_PruebaPretransfunsional	NombFunc	id_funcionario	cfg_funcionario del módulo Configuración.

Anexo 3. Entidades que se obtienen debido a anomalías

A)

B)

C)

GLOSARIO DE TÉRMINOS

backup	(Copia de seguridad) Es la copia total o parcial de información importante del disco duro, CDs, bases de datos u otro medio de almacenamiento. Los backups se utilizan para tener una o más copias de información considerada importante y así poder recuperarla en el caso de pérdida de la copia original.
base de datos	Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada ó estructurada. Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.
Componente Sanguíneo	Cada unidad de sangre se degrada en componentes, como los glóbulos rojos, el plasma y las plaquetas. Una unidad de sangre entera, una vez que está separada, puede ser transfundida a varios pacientes, cada uno con diferentes necesidades.
Factor rh	El factor Rh es una proteína que se encuentra en la cubierta de los glóbulos rojos. Si la proteína del factor Rh está presente en las células, la persona es Rh positivo. Si no hay proteína del factor Rh, la persona es Rh negativo.
Grupo Sanguíneo	Un grupo sanguíneo es una forma de agrupar ciertas características de la sangre que dependen de los antígenos presentes en la superficie de los glóbulos rojos y en el suero de la sangre. Estos grupos son cuatro y se denominan: 0, A, B, AB.
log	Un log es un registro de actividad de un sistema, que generalmente se guarda en un fichero de texto, al que se le van añadiendo líneas a medida que se realizan acciones sobre el sistema. Se utiliza para guardar información sobre la actividad de sistemas variados. A través de ellos se puede encontrar información para detectar posibles problemas en caso de que no funcione algún sistema como debiera o se haya producido una incidencia de seguridad.
Open source	Denominación para aquellas aplicaciones que tienen su código fuente liberado. Se puede obtener su código fuente, arreglarlo y modificarlo para satisfacer las necesidades de sí

	mismo.
scripts	Los scripts son un conjunto de instrucciones generalmente almacenadas en un archivo de texto que deben ser interpretados línea a línea en tiempo real para su ejecución, se distinguen de los programas, pues deben ser convertidos a un archivo binario ejecutable para correrlos.
sistemas gestores de bases de datos	Los sistemas gestores de bases de datos, son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan; que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.
Stock	Se utiliza en español con el sentido general de reserva de alguna cosa disponible para un uso futuro.
transact-SQL	Es una extensión del lenguaje SQL.
versión beta	Versiones de un programa o aplicación que no están completamente depuradas y, por lo tanto, no pueden ser comercializadas. Las versiones beta son versiones de prueba de un software determinado, desarrolladas antes de la versión definitiva. Con el Fin de detectar los posibles errores o bugs en el funcionamiento del programa, estos suelen entregarse previamente a cualificados beta-testers que los prueban y analizan.
web service	Un servicio Web (en inglés *Web service*) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes y ejecutadas sobre cualquier plataforma pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.