Universidad de las Ciencias Informáticas

Facultad 6

Centro de Inmunología Molecular

SIMDEC

Sistema de Manejo de Datos de Ensayos Clínicos: Diseño e implementación de la Base de Datos.

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autor(es): Arodys E. Domínguez Vaillant

Duniel Miranda Gutiérrez

Tutor(es): Ing. Maypher Román Durán

Ing. Sergio Barrios Díaz.

Lic. Adrián Garrido Hernández.

Ciudad de La Habana, Julio del 2007

"Año 49 de la Revolución"

“Sin motivación no hay amor, y sin amor por la tarea que se cumple, no hay resultados”.

Raúl Castro

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 _____ días del mes de _____ del año _____.

_____ _____

Arodys Eugenio Domingue Vaillant (Autor) Duniel Miranda Gutiérrez(Autor)

_____ _____

Ing. Maypher Román Durán (Tutor) Ing. Sergio Barrios Díaz (Tutor)

_____

Lic. Adrián Garrido Hernández(Tutor)

DATOS DE CONTACTO.

Ingeniero: Maypher Román Durán
Universidad de Ciencias Informáticas, Habana, Cuba
E-mail: maypher@uci.cu

Ingeniero: Sergio Barrios Díaz
Universidad de Ciencias Informáticas, Habana, Cuba
E-mail: sbarrios@uci.cu

Licenciado: Adrián Garrido Hernández
Universidad de Ciencias Informáticas, Habana, Cuba
E-mail: agadez@uci.cu

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma titulado “Sistema de Manejo de Datos de Ensayos Clínicos: Diseño e implementación de la Base de Datos”, fue desarrollado en la Universidad de las Ciencias Informáticas en coordinación con el Centro de Inmunología Molecular. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

Totalmente **X**

Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes:

Los trabajos de diploma analizados forman parte de un proyecto de colaboración de diversos centros del polo con la UCI para el desarrollo de un sistema informático para el manejo de la información clínica. Cada uno de ellos abordó el análisis y diseño de uno de los módulos que conformaran el sistema final. En todos los casos se realizó un exhaustivo levantamiento de requisitos, que culminó con el desarrollo de un prototipo no funcional que cubre las necesidades de nuestra institución. Adicionalmente se realizó el correspondiente diseño, lo que sienta las bases para la continuidad del proyecto en el próximo curso académico. Los trabajos hasta aquí desarrollados no tienen un valor de uso en sí mismos. Pero de culminarse la programación del sistema final este podría ahorrarle al CIM y a otras instituciones del polo una inversión en software comerciales superior a los 500000 dólares, redundando adicionalmente una mejoría considerable de su competitividad a nivel mundial.

Y para que así conste, se firma la presente a los _2_ días del mes de __Junio____ del año __2007__

Dr. C Kalet León Monzón	Director Biología de Sistema
Representante de la entidad	Cargo
_____	_____
Firma	Cuño

Dedicatoria

A mis padres por ser mis mayores guías en esta tarea.

A mi hermano por su apoyo y su preocupación incondicional.

A mis amigos y compañero de estudios por su apoyo y estar siempre presentes,

Arodys

A mi madre por tomarme en su corazón y llevarme de su mano,

A mis hermanos por el cariño que me brindan en todo momento,

A mis amigos y colegas por su preocupación y apoyo incondicional.

Duniel

Agradecimientos

De Arodys:

Quiero agradecer a todas aquellas personas que con su apoyo incondicional han hecho posible la realización exitosa de este trabajo:

A Fidel y la Revolución cubana por permitir mis estudios y graduarme en esta universidad de gran prestigio nacional e internacional. Asimismo agradezco a esta Universidad por aportarme tantos conocimientos.

A mis padres por su preocupación, dedicación y amor incondicional, se que hoy cumplo uno de sus mayores anhelos.

A mi hermano por su preocupación y ayuda desinteresada, se que seguirá mis pasos y sentirá gran felicidad al verme graduado.

A mi familia y mis vecinos por su preocupación y estar siempre pendiente de mis estudios.

A mis amigos: Arcel, Nela, Karenia, que desde el primer año de la carrera venimos juntos, me han ayudado en todo y siempre estarán presentes porque más que mis amigos, son mis hermanos.

A mi amigo Arles que siempre me ayudó y se preocupó cada momento como iba el desarrollo de mi tesis. Gracias mi hermano, siempre estarás presente.

A mis amigos: Lissandra, Karelia, Yulieimis, Yaritza, Licet, Yusmy, Herick, Danay, Daurys, Anarelys y Osdalme, por ayudarme en todo para que este sueño se hiciera realidad, siempre serán mis amigos.

A Duniel, que aparte de ser mi dúo de tesis lo considero mi hermano y hemos pasado junto todo este tiempo de la carrera y ahora esta etapa importante de nuestras vidas, la tesis.

A mis amigos Bismarck y Frank por su preocupación, siempre serán mis hermanos.

A Yeniel, Cantillo, Yaisel, Jorgito, Ricardo y Yuri por su preocupación y ayuda incondicional, siempre serán mis amigos.

A mis compañeros de cuarto: Yudel, Yaiser, Pupo, Manuel, Yoendris Carlos y Pedro L. por su preocupación y ayuda desinteresada.

A mis colegas del proyecto: Linet, Yordanis, Elvira, Mailín, Karel, Dunia, Lucia; porque juntos hemos pasado todo este tiempo de tesis y se han preocupado por la realización de este trabajo.

A Aida que no por mencionarla antes deja de ser menos importante, pues me ha ayudado, aconsejado mucho y me ha hecho ver la vida de una forma profesional.

A Janny, Danay y Onailet por su apoyo.

A Yurima, Yanet Alonso, Yetel por ser mis colegas en estos años de la universidad.

A mis compañeros de la universidad: Daliagna, Yiselis, Elennis, Jorge Daniel, Yoendris L., Andy, Yaikiel, Rigoberto, Jose L., Yaquerleidys, Yindra, Mónica.

A mis tutores por su preocupación y aporte importante en este trabajo.

Los que de manera involuntaria he dejado de citar mis disculpas y mis más sinceros agradecimientos.

A todos muchísimas gracias.

De Duniel:

Creo haber llegado al momento más difícil de mi carrera, y es el de intentar resumir en nombres de personas e instituciones lo agradecido que en este momento me siento. Pienso que ésta sección, a lo mejor, no sea suficiente para mencionar a todos aquellos que han puesto un granito de empeño y su ayuda desinteresada para cumplir este, mi sueño, y si existe alguien que no sienta su presencia en mis palabras, reciba mis más sinceras disculpas y tenga la certeza que siempre tendrá mi gratitud.

Quiero agradecer a mis padres, por permitirme nacer y escalar hasta el lugar donde me encuentro hoy. En especial con todo el corazón, a mi madre, por ser la estrella que me ha guiado por el camino correcto, por ser la fuerza que me hace levantar la cabeza y continuar adelante, aún en tiempos de debilidad, y por ser sobre todas las cosas, la amiga que siempre ha estado presente para darme el consejo adecuado. Sé que hoy cumplo uno de sus mayores anhelos.

A mis hermanos Dianelis y David, por su preocupación y apoyo incondicional, se que hoy sienten una gran felicidad al verme cumplir este sueño.

Mis agradecimientos a Arodys, por quien siento una gran admiración y respeto, gracias por su empeño constante, por los momentos en que me despertó cuando me vencía el cansancio, que no fueron pocos. Por ser un amigo insuperable y un hermano: Gracias.

Para todos aquellos, que por sobre todas las pruebas de la vida, estuvieron presente, brindando su apoyo, gracias. A Yulieimis, Yaritza, Yusmilia, Licet, Daliagna quienes nunca se incomodaron con mis preguntas constantes, gracias, son maravillosas. A mis amigos

eternos de las buenas y las malas, de las copas y las resacas; a Yaikiel, Andy, Rigo, Yaiser, Lisbell.

A todos mis colegas del proyecto, que no por último son menos importante, a Aida, Mailin, Elvira, Linet, Lucia, Karel, Dunia, Yordanys, Bismark, a todos por su ayuda, muchas gracias.

A Yaimila, Yehilit por su apoyo desinteresado. A mis vecinos en el laboratorio: Janny, Danay, Yaque, Jose Luis, a todos por su preocupación, gracias.

A mis tutores por su interés en la realización exitosa de este trabajo, y por todo lo que han aportado en mi formación profesional: muchas gracias.

Mi agradecimiento al Comandante en Jefe Fidel por darnos esta Revolución y la posibilidad de estudiar en un centro de tanto prestigio como este. A la Universidad de las Ciencias Informáticas por permitir elevar mi intelecto y cultura al nivel de un profesional digno de estos tiempos.

Para todos los amigos de 5 años de estudio, de fiestas y trabajo, de lluvias y sol, de amaneceres y marchas, a todos, gracias.

Y a todo aquel que luego del abrazo y del saludo, preguntó ¿Cómo va la Tesis?, muchas, muchas gracias.

Resumen

Los Ensayos Clínicos (EC) juegan un papel fundamental en el estudio de enfermedades crónica no transmisible, principalmente el cáncer.

Actualmente, hay varios cientos de bases de datos de EC en todo el mundo, pero con escasa relación entre ellas. El almacenamiento de la información de estos EC presenta dificultades debido a la cantidad que se almacena, siendo insuficiente, y los Sistemas de Bases de Datos en que se almacenan, que no son gestores profesionales. En nuestro país una de las entidades que se dedica al estudio de los ensayos clínicos es el Centro de Inmunología Molecular (CIM), el cual no está ajeno a los problemas existentes en la recogida de información, antes mencionados.

En busca de soluciones surge la investigación actual, en el marco del trabajo del proyecto: Ensayos Clínicos (EC) del CIM, desarrollado conjuntamente con la Universidad de las Ciencias Informáticas (UCI).

En la investigación se realiza el diseño e implementación de la base de datos que contribuya al manejo de los datos que se generan en los EC.

Índice

Índice de Figuras.

Introducción

“El desarrollo de un nuevo producto o la introducción de un producto ya conocido en una nueva indicación médica trae aparejado diferentes investigaciones, entre las que se encuentra el ensayo clínico como etapa final, que permite el registro y comercialización de dicho producto para ser utilizado en los seres humanos”. [1].

Un Ensayo Clínico (EC) es un tipo de estudio clínico en el que se evalúan nuevos fármacos o tratamientos médicos con un protocolo de investigación estrictamente controlado.[2]

El proceso de recolección de la información registrada en un EC constituye un aspecto, que una vez finalizada la inclusión de los sujetos en el estudio, puede demorar la terminación del mismo. El protocolo del estudio establece una norma general según la frecuencia de las evaluaciones. Sin embargo es necesario conocer la frecuencia individual de cada paciente, a manera de mantenerlo informado, así como evitar posibles pérdidas durante todo el proceso de evaluación.

En el mundo, desde hace varios años, se trabaja en implementar los EC en versión electrónica, no solo los Cuadernos de Recogida de Datos (CRD), sino las bases de datos que recogen toda la información que se encuentran en estos CRD, comunicaciones, aprobaciones, revisiones reguladoras, gerencia del proyecto, entre otros.

“Actualmente, hay varios cientos de bases de datos de EC en todo el mundo, pero con escasa coordinación entre ellas. La plataforma de registros pretende unirlos en una red mundial que proporcione un único punto de acceso a la información que almacenan. La iniciativa no será un registro en sí, sino que proporcionará una serie de estándares para todas las bases de datos. Además de tipificar la información que debe darse al registrar un ensayo, crea un sistema internacional de identificación de estudios que otorgará un número de referencia único a cada ensayo cualificado”.[3]

En Cuba en el estudio de los EC se destaca el Centro de Inmunología Molecular (CIM), inaugurado el 5 de diciembre de 1994, perteneciente al polo científico del Oeste de La Habana, que tiene como principal misión obtener y producir nuevos bio-fármacos destinados al tratamiento del cáncer y otras enfermedades

crónicas no transmisibles e introducirlos en la Salud Pública cubana. Hacer la actividad científica y productiva económicamente sostenible y realizar aportes importantes a la economía del país.[4]

El CIM realiza, en hospitales altamente especializados, EC para el diagnóstico de tumores por imágenes y tratamiento de cáncer de diferentes orígenes y otras enfermedades del sistema inmune. Además ha venido desarrollando una serie de bio-moléculas para el tratamiento de diferentes enfermedades, principalmente el cáncer. En la medida que se avanza en el desarrollo de estos productos, se avanza en las fases de los EC de fases I, I/II a fases II, II/III y fases IV. Esto trae aparejado el incremento del número de pacientes a estudiar y por consiguiente el número de hospitales que se enrolan en el reclutamiento de pacientes.

Los EC llevan asociados una gran cantidad de documentación (datos primarios, imágenes, CRD, aprobaciones, modificaciones, entre otros), necesaria para cumplir con las buenas prácticas clínicas exigidas por todas las agencias reguladoras a nivel mundial. Esta documentación debe ser almacenada no menos de 15 años posterior al cierre del estudio, de modo que pueda ser inspeccionada en cualquier momento por las agencias reguladoras. Se podría decir que por cada paciente de un EC se genera por lo menos 1000 datos diferentes, lo cual permite fácilmente estimar el gran volumen de información que se genera y maneja en esta actividad. Por ejemplo, en los 2500 pacientes que manejó el CIM en el 2005 se generaron no menos de 2 millones 500 mil datos, los cuales son de muy diversos tipos, desde información referida a mediciones cuantitativas del estado de un paciente (presión sanguínea, hemoglobina, entre otros), hasta imágenes que caracterizan el tamaño del tumor.[5]

En la actualidad, los EC en el CIM y en Cuba se desarrollan de la manera clásica, toda la información se encuentra en papel, y la transmisión de información de los hospitales a los centros promotores, y viceversa, se hace generalmente vía correo electrónico o vía telefónica. “Un complicado proceso de monitoreo de la información clínica se realiza visitando periódicamente cada uno de los sitios participantes (incluyendo las provincias orientales), con el objetivo de controlar la calidad y uniformidad de los datos recogidos. Finalmente bases de datos en versión electrónica se preparan a partir de los CRD llenados en los hospitales, con el objetivo de facilitar el análisis final de los resultados de cada ensayo”. [5]

Actualmente las bases de datos para la recogida de información de los EC en el CIM se confeccionan con las aplicaciones de software Microsoft Access y EPINFO, teniendo que hacerse una entrada de datos doble desde los CRD, por parte de dos operadores de máquina no relacionados con la recogida primaria

de los datos del estudio. La generación de estas bases de datos resulta compleja, dado la gran variabilidad de la información recogida en los hospitales (por ejemplo uso de unidades diferentes para una determinación de interés) y la dificultad intrínseca del proceso de transcribir de manera fiable información de un formato impreso a un formato electrónico. Pero más aún, estas bases de datos no contienen toda la información de interés referente al ensayo y por la simplicidad de su diseño no permite la realización de meta-análisis estadísticos complejos dentro de un mismo ensayo o entre diferentes EC.

Por todo lo antes expuesto, se plantea como **problema científico** de la investigación: ¿Cómo contribuir a un almacenamiento homogéneo de los datos asociados a los EC?

Por lo cual el **objeto de estudio** de esta investigación es la automatización de los procesos de almacenamiento de información de los EC.

De ello se deriva que el **campo de acción** que abarca esta investigación es el diseño e implementación de la base de datos para la automatización de los procesos de almacenamiento de información de los EC en el CIM.

Esta investigación tiene como **objetivo general**: Diseñar e implementar una base de datos que almacene de forma homogénea la información de los CRD para los EC.

Este objetivo se desglosa en los siguientes **objetivos específicos**:

- ✓ Validar teóricamente el diseño de la base de datos
- ✓ Diseñar e implementar la base de datos.
- ✓ Realizar una validación funcional.

Para cumplir cada uno de los objetivos propuestos se plantean una serie de **tareas**:

- ✓ Estudio de las tendencias y tecnologías actuales de las base de datos a nivel mundial.
- ✓ Selección de la topología de base de datos a utilizar.
- ✓ Análisis de la integridad de los datos en la base de datos.

- ✓ Estudio de la normalización de la base de datos.
- ✓ Análisis de redundancia de información de la base de datos.
- ✓ Análisis de la seguridad de la base de datos.
- ✓ Estudio de trazabilidad de las acciones en la base de datos.
- ✓ Diseño e implementación de la base de datos.
- ✓ Revisión y selección de herramientas para pruebas de carga intensiva.
- ✓ Pruebas a la base de datos con la herramienta seleccionada.

El documento está estructurado por 3 capítulos.

En el **Capítulo 1: Fundamentación Teórica,** se brinda información de la institución para la cual se trabaja e información referida a los ensayos clínicos. Se describen las bases de datos existentes en la actualidad relacionadas con la recogida de información de los ensayos clínicos, al igual que los modelos de bases de datos, tecnologías, herramientas y metodologías, partiendo, como base, de sus ventajas y desventajas, para la selección y justificación de la solución propuesta.

En el **Capitulo 2: Descripción y Análisis de la Solución Propuesta**, se describe la solución propuesta, haciendo énfasis en los requisitos funcionales y no funcionales, al igual que el diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño, todas estas cuestiones referentes al trabajo del analista. Se hará una descripción de todas las clases, el diseño de la base de datos, incluyendo el diagrama entidad relación y la descripción de aquellas tablas importantes. También se hace una pequeña descripción de la arquitectura propuesta, así como un análisis de optimización de consultas.

En el **Capítulo 3: Validación del Diseño Realizado**, se brinda información sobre la validación realizada al diseño descrito en el capítulo 2, describiendo en detalle las reglas a tener en cuenta para un diseño de óptima calidad, así como los detalles de una validación funcional a través de la selección de consultas para un llenado voluminoso e inteligente de la BD.

Capítulo 1

Fundamentación Teórica

En el presente capítulo se brinda información de la institución para la cual se trabaja e información referida a los ensayos clínicos. Se describen las bases de datos existentes en la actualidad relacionadas a la recogida de información de los ensayos clínicos, al igual que, los modelos de bases de datos, tecnologías, herramientas y metodologías, partiendo, como base de sus ventajas y desventajas, para la selección y justificación de la solución propuesta.

1.1 Ensayos Clínicos

Los EC permiten a los médicos determinar si un nuevo tratamiento, medicamento o dispositivo contribuirá a prevenir, detectar o tratar una enfermedad. Los EC también ayudan a los médicos a descubrir si estos nuevos tratamientos son inocuos y si son mejores que los tratamientos actuales.

Hay cuatro tipos de ensayos clínicos:

- Ensayos Clínicos de medios de tratamiento, para evaluar nuevos tratamientos, medicamentos o intervenciones quirúrgicas.
- Ensayos Clínicos de medios de prevención, para encontrar maneras de prevenir las enfermedades con medicamentos, vitaminas, vacunas o cambios en el estilo de vida.
- Ensayos Clínicos de medios de detección, para evaluar maneras de detectar o diagnosticar las enfermedades.
- Ensayos Clínicos de calidad de vida, para encontrar maneras de mejorar la vida de las personas que viven con una enfermedad o problema de salud.

Los Ensayos Clínicos tienen asociados cuatro fases:[6]

Fase I: Es el primer paso en la investigación de una sustancia o medicamento nuevo en el hombre. Estudio que proporciona información preliminar sobre el efecto y la seguridad del producto en sujetos sanos (el caso de los EC en pediatría).

Fase II: Se realiza en pacientes que padecen la enfermedad o entidad clínica de interés. Su principal objetivo es proporcionar información preliminar sobre la eficacia del producto, establecer la relación dosis-respuesta del mismo, conocer las variables empleadas para medir eficacia y ampliar los datos de seguridad obtenidos en la Fase I.

Fase III: Son ensayos destinados a evaluar la eficacia y seguridad del tratamiento experimental intentando reproducir las condiciones de uso habituales y considerando las alternativas terapéuticas disponibles en la indicación estudiada. Se realiza con una muestra de pacientes más amplia que en la fase anterior y representativa de la población general a la que irá destinado el medicamento.

Fase IV: Ensayos clínicos que se realizan con un medicamento después de su comercialización. Estos ensayos podrán ser similares a los descritos en las fases I, II, III. Se vigilan los posibles efectos adversos no consignados en las fases anteriores, al igual que su efectividad en distintas poblaciones.

En Cuba con la misión principal de obtener y producir nuevos bio-fármacos destinados al tratamiento del cáncer y otras enfermedades crónicas no transmisibles e introducirlas en la Salud Pública Cubana se inauguró el Centro de Inmunología Molecular (CIM), el 5 de diciembre de 1994.

EL CIM realiza en hospitales altamente especializados EC para el diagnóstico de tumores por imágenes y tratamiento de cáncer de diferentes orígenes y otras enfermedades.

1.2 Procesos de recogida de datos en los Ensayos Clínicos.

Los EC llevan asociados una gran cantidad de documentación (datos primarios, imágenes, CRD, aprobaciones, modificaciones, entre otras), necesaria para cumplir con las buenas prácticas clínicas exigidas por todas las agencias reguladoras a nivel mundial. Para el almacenamiento de una parte de esta información se han creado los Cuadernos de Recogida de Datos (CRD), capaces de almacenar todas las variables recogidas durante la ejecución del ensayo, siendo capaz de registrar todos los datos que se recogen del protocolo.

Esta documentación debe ser almacenada no menos de 15 años posterior al cierre del estudio, de modo que pueda ser inspeccionada en cualquier momento por las agencias reguladoras. Como un estimado conservador podríamos decir que por cada paciente de un ensayo clínico se genera por lo menos 1000 datos diferentes, lo cual permite fácilmente estimar el gran volumen de información que se genera y maneja en esta actividad. Por ejemplo, en los 2500 pacientes que manejó el CIM en el 2005 se generaron no menos de 2 millones 500 mil datos, los cuales son de muy diversos tipos, desde información referida a mediciones cuantitativas del estado de un paciente (presión sanguínea, hemoglobina, entre otros), hasta imágenes que caracterizan el tamaño del tumor, o hasta información referida al proceso de intercambio de mensajes entre los investigadores que conducen los estudios.

En la actualidad, los EC en el CIM y en Cuba se desarrollan de la manera clásica, toda la información se encuentra en papel, y la transmisión de información de los hospitales a los centros promotores y viceversa se hace generalmente vía correo electrónico o vía telefónica. Un complicado proceso de monitoreo de la información clínica se realiza visitando periódicamente cada uno de los sitios participantes (incluyendo las provincias orientales), con el objetivo de controlar la calidad y uniformidad de los datos recogidos. Finalmente, bases de datos en versión electrónica se preparan a partir de los CRD llenados en los hospitales, con el objetivo de facilitar el análisis final de los resultados de cada ensayo.

Actualmente las bases de datos para la recogida de información de los EC en el CIM se confeccionan con las aplicaciones de software Microsoft Access y EPINFO, teniendo que hacerse una entrada de datos doble desde los CRD, por parte de dos operadores de máquina no relacionados con la recogida primaria de los datos del estudio. La generación de estas bases de datos resulta hoy en extremo compleja, dado la gran variabilidad de la información recogida en los hospitales (por ejemplo uso de unidades diferentes para una determinación de interés) y la dificultad intrínseca del proceso de transcribir de manera fiable información de un formato impreso a un formato electrónico. Pero más aún, estas bases de datos no contienen toda la información de interés referente al ensayo y por la simplicidad de su diseño no permite la realización de meta-análisis estadísticos complejos dentro de un mismo ensayo o entre diferentes ensayos clínicos.

1.3 Surgimiento y desarrollo de las Bases de Datos (BD).

Tras el desarrollo de las tecnologías y la necesidad del almacenamiento de la información surgen los archivos secuenciales como almacenes de datos. Estos daban un acceso muy rápido pero sólo de forma

secuencial (para acceder a una posición, se debía recorrer el archivo entero). Como solución aparecieron los archivos indexados, donde el acceso ya podía ser aleatorio (acceder de una vez a la posición deseada del archivo).

Dada la gran complejidad que iban alcanzando los programas y el volumen de datos que generaban, se requería de un almacenamiento que garantizara un cierto número de condiciones y que permitiera operaciones complejas sin que se violaran estas restricciones. Además cada usuario que accediera a los datos debía tener su trabajo protegido de las operaciones que hicieran el resto de usuarios.

Como respuesta a estas necesidades, surgieron las BD, que constituyen una colección de datos interrelacionados que se puede utilizar por uno o más programas de aplicación, puede considerarse una colección de datos variables en el tiempo, surgieron como primera topología la jerárquica donde los datos se situaban siguiendo una jerarquía. Las BD jerárquicas tenían el problema que los accesos a los datos eran unidireccionales, y era más complicado hacer el camino inverso (pero posible, aunque el tiempo de cálculo era mayor). Como variante a esta topología surgieron las BD de red, modelo ligeramente distinto del jerárquico, el cual permitía que un mismo nodo tenga varios padres, aún así no era una solución factible. Para dar absoluta libertad a las relaciones entre tablas surgieron las BD relacionales. Estas trajeron dos cosas muy importantes: las propiedades ACID y un lenguaje común de acceso a los datos: SQL.

Las propiedades ACID son:

Atomicidad. Cada transacción del usuario debe tratarse de forma atómica. O se ejecuta todo o nada, es decir, el trabajo se realiza en su totalidad o no se realiza en ningún caso.

Consistencia. Las transacciones han de cumplir las restricciones definidas dentro de la BD. Si no las pueden cumplir, se evita su ejecución. De esta forma se conserva la integridad y coherencia de los datos.

Aislamiento. Una transacción es una unidad de aislamiento, permitiendo que transacciones concurrentes se comporten como si cada una fuera una única transacción que se ejecuta en el sistema. Las transacciones alcanzan el nivel más alto de aislamiento cuando se pueden serializar. En este nivel, los resultados obtenidos de un conjunto de transacciones concurrentes son idénticos a los obtenidos mediante la ejecución en serie de las transacciones.

Durabilidad. Una vez se ha completado la transacción, los resultados de la misma han de ser permanentes y sobrevivir a posibles caídas del sistema o la BD.

1.4 BD y sus componentes principales.

El volumen de información que puede almacenar una BD puede ir desde el de una agenda, un contador, o un libro de visitas, hasta el de una tienda en línea, un sistema de noticias, un portal, o la información generada en una red corporativa.

Las BD proporcionan la infraestructura requerida para los sistemas de apoyo a la toma de decisiones y para los sistemas de información estratégicos, ya que estos sistemas explotan la información contenida en las BD de la organización para apoyar el proceso de toma de decisiones o para lograr ventajas competitivas. Están formadas por cuatro componentes principales: Datos, Hardware, Software y Usuarios.

Datos: se refiere a que los datos en la BD serán tanto integrados (unificación de varios archivos que de otro modo serían distintos) como compartidos (las piezas individuales de datos en la base pueden ser compartidas entre diferentes usuarios, con lo que cada uno puede acceder a la misma pieza de datos con fines diferentes).

Hardware: se refiere a los dispositivos de almacenamiento donde reside la BD, así como a los dispositivos periféricos (unidad de control, canales de comunicación, etc.) necesarios para su uso.

Software: constituido por un conjunto de programas que se conoce como Sistema Manejador de BD (DBMS: Data Base Management System). Este sistema maneja todas las solicitudes formuladas por los usuarios a la BD.

Usuarios: Existen tres clases de usuarios relacionados con una BD:

- El programador de aplicaciones, quien crea programas de aplicación que utiliza la BD.
- El usuario final, quien accede a la BD por medio de un lenguaje de consulta o de programas de aplicación.
- El administrador de la BD (DBA Data Base Administrator), quien se encarga del control general del Sistema de BD.

1.5 Modelos de BD.

Un Modelo de Datos es básicamente una "descripción" del contenedor de datos (donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de BD; por lo general se refieren a algoritmos, y conceptos matemáticos. Entre los modelos de BD se encuentran:

Jerárquicas

Los datos se organizan en una forma similar a un árbol, donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se llaman hojas.

Este modelo no es utilizado porque una de sus principales limitaciones es: “la poca flexibilidad de este modelo puede obligar a la introducción de redundancia cuando es preciso instrumentar, mediante el modelo jerárquico, situaciones del mundo real que no responden a una jerarquía”.[7]

Red

Modelo ligeramente distinto del jerárquico, su diferencia fundamental es la modificación del concepto de nodo; éste permite que un mismo nodo tenga varios padres.

Gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; aún así, la dificultad que significa administrar la información en una BD de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales, por lo que este modelo no se utiliza.

Relacional

Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la BD. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información. [8]

Orientadas a Objetos

Se crearon para tratar de resolver algunas deficiencias que no resolvían los modelos tradicionales (jerárquico, red y relacional) cuando se trata de aplicaciones más complejas o sofisticadas, por ejemplo, el diseño y fabricación en ingeniería.

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la BD los objetos completos (estado y comportamiento).

Una BD orientada a objetos incorpora todos los conceptos importantes del paradigma de objetos:

- Encapsulación: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En el modelo orientado a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la BD.

El modelo de BD seleccionado fue el Relacional.

“Las BD Relacional son muy flexible ya que los elementos que la integran se pueden ingresar de modo independiente a la estructura que quiera formarse con ellos (o sea, primero las tablas luego sus relaciones) mientras que en los otros modelos la estructura ya esta definida”. [9]. Además en este modelo se puede realizar cualquier clase de búsqueda de datos entre tablas una vez que existan campos comunes entre estas.

1.6 BD desarrolladas en los EC.

Desde los inicios, se ha planteado la necesidad de una homogeneización del almacenamiento de los datos obtenidos como resultados de los EC a nivel mundial, pero, hasta el momento la idea que se ha materializado es el registro de todo ensayo clínico en una BD que esté al alcance del público. Las BD desarrolladas hasta la actualidad cuentan con poca cohesión entre ellas, siendo este el principal problema.

A fines del siglo pasado se crearon diversos registros públicos para ensayos clínicos. Destacan algunas iniciativas:

- En los Estados Unidos se creó un registro financiado con dineros públicos (www.clinicaltrials.gov). ClinicalTrials.gov es un registro de ensayos clínicos de la Biblioteca Nacional de Medicina de los Estados Unidos, el registro comprende ensayos federales y con financiación privada, que se realizan para una serie de enfermedades y condiciones médicas, brindando información actualizada sobre éstos EC. Contiene, actualmente, más de 36.100 estudios clínicos, patrocinado por el Instituto Nacional de Salud, otras agencias federales e industrias privadas. La lista de estudios en la base de datos se realizan en los 50 estados y en alrededor de 130 países. El sitio ClinicalTrials.gov recibe más de 12 millones de visitas por año y aproximadamente, 31.000 visitas diarias. [10]
- En Europa se estableció un metaRegistro. Recopilación internacional de ensayos clínicos completados o en desarrollo incluyendo los ensayos aleatorios de la *UK* Medical Research Council, la National Health Service (NHS) y el US Department of Veterans Affairs Co-operative Studies Program-, (www.controlled-trials.com) con el respaldo del Consejo de Investigación Médica (MRC) y del Programa de Investigación y Desarrollo del Sistema Nacional de Salud (NHS) del Reino Unido. Current Controlled Trials (CCT) es parte del Current Science Group (CGS) y es publicada por BioMed Central (publicación Open Acces Now). El CCT brinda información sobre ensayos clínicos de todos los países del mundo y en todas las áreas de la salud. En él las organizaciones registran un Ensayo, utilizando el Número Aleatorio Estándar Internacional para Ensayos Controlados (ISRCTN), un número de 8 dígitos, único para cada Ensayo, al cual se accede en línea de forma libre, pueden almacenar su Ensayo, ya registrado; el acceso al metaRegistro es libre para todos los usuarios. Por otra parte los usuarios pueden navegar con el ISRCTN el

metaRegistro, brindándole acceso a alrededor de 15.000 Ensayos registrados para cualquier área de la Salud. [10]

- La colaboración Cochrane también asumió un papel pionero con su Cochrane Central Register of Controlled Trials (CENTRAL). Registro que contiene alrededor de 430.000 informes de ensayos. Es una BD que contiene revisiones sistemáticas y otra información para ayudar a la toma de decisiones diagnósticas y terapéuticas. Contiene cuatro BD principales, entre las que se encuentra un registro de Ensayos Clínicos controlados. Su objetivo es promover, producir y diseminar revisiones sistemáticas y periódicas de EC controlados. [10].

En Cuba, por su parte, los pasos en la búsqueda de soluciones para el almacenamiento de los ensayos clínicos, se pueden calificar de lentos. Hasta el año 2005 se puede decir que los indicadores no eran los más favorables, por ejemplo:

- "En el año 2001, el Centro Nacional Coordinador de Ensayos Clínicos (CENCEC) desarrolló una plantilla en EXCEL con el propósito de crear un registro electrónico del Cuaderno de Recogida de Datos (CRD) de un ensayo que permite, además, llevar el control del registro de inclusión del estudio y el comportamiento de la no inclusión".[11]
- En el 2005 se diseñó una base de datos para almacenar la información referente a los pacientes incluidos o no en los 12 ensayos clínicos que se encontraban actualmente en ejecución en la provincia de Cienfuegos, debido a la necesidad de agilizar el trabajo de los coordinadores; el diseño se hizo con el programa Microsoft Access 2000 del paquete Office XP de Microsoft, contó con nueve tablas relacionadas entre si, cinco formularios y 4 sub-formularios interrelacionados de igual forma, permitiendo la captura de los datos pertenecientes a los pacientes; además tiene una serie de consultas, las cuales estratifican la información contenida en la base de datos hasta el nivel deseado, mostrando de forma dinámica y ágil sus reportes correspondientes logrando elevar los niveles de eficacia en el desarrollo de esta tarea; actualmente está concebida solo para Cienfuegos, pero puede aplicarse a cuanto ensayo clínico entre en vigor nacionalmente implementándose un clasificador de entidades en cada provincia. La BD logró recopilar la documentación generada en cada ensayo, de forma tal de tenerla disponible siempre que haga falta para hacer una investigación, aclarar dudas y enviar alguna información sin correr riesgos de pérdidas de datos partiendo siempre de una ganancia en economía de tiempo. [12]

Todos estos sistemas anteriormente mencionados, principalmente a nivel nacional poseen algunas limitaciones, pues se realizan en gestores no profesionales que no permiten la gran acumulación de información que se generarán posteriormente en los ensayos, por lo que se hacen necesario perfeccionarlos para un mejor control de los ensayos clínicos.

1.7 Sistemas Gestores de Base de Datos (SGBD).

El software de BD ha experimentado un auge extraordinario, además de rapidez y efectividad en los procesos. Con la existencia de múltiples entornos de desarrollo y la notable demanda de soluciones informáticas, han surgido multitud de gestores de BD.

Siendo los SGBD un tipo de software muy específico, dedicado a servir de interfaz entre la BD, el usuario y las aplicaciones que la utilizan y que prestan servicios para el desarrollo y el manejo de BD.

Dentro de los principales SGBD propietarios más potentes se encuentra Oracle y Microsoft SQL Server, y dentro del software libre, tenemos opciones muy completas como: MySQL y PostgreSQL.

Ventajas y Desventajas de los SGBD.

Ventajas:

- Eliminan las inconsistencias en los datos.
- Permiten compartir los mismos datos entre diferentes aplicaciones con distintas necesidades.
- Ahorran espacio de almacenamiento al no existir redundancia o ser ésta escasa.
- Mejoran la seguridad de los datos, pues normalmente incorporan mecanismos de seguridad en el propio SGBD.
- Permiten la creación de entornos de alta disponibilidad, pues con algunos SGBD es posible llegar a disponer de aplicaciones funcionando ininterrumpidamente
- Mejora en los servicios de copias de seguridad y de recuperación ante fallos.

Desventajas:

- Requieren una gran cantidad de espacio en disco y de memoria para trabajar de forma eficiente.

- Vulnerable a los fallos. El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse.

1.7.1 Oracle

Es un sistema de gestión de BD (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), que utiliza arquitectura cliente/servidor, fabricado por Oracle Corporation. Se considera como uno de los sistemas de BD más completos, destacando su: soporte de transacciones, estabilidad, escalabilidad y el ser multiplataforma.[13]

Ventajas:

- Es manejador de BD relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información.
- Puede ser usado tanto para el manejo de información personal, como para gigantescas bibliotecas multimedia, y corre en laptops, computadoras personales (PC), microcomputadoras, mainframes y computadoras con procesamiento paralelo masivo.
- Soporta la mayoría de los lenguajes de computación al igual que 26 idiomas diferentes.
- Corre automáticamente en más de 80 arquitecturas de hardware y software distinto sin tener la necesidad de cambiar una sola línea de código, esto se debe a que más del 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de sistemas.
- Soporta datos alfanuméricos ubicados en las tradicionales "filas y columnas" de las BD, sino que también soporta textos sin estructura, imágenes, audio y video.
- Incluye mejoras de rendimiento y de utilización de recursos.
- Soporta aplicaciones de procesamiento de transacciones online (OLTP) y de data warehousing mayores y más exigentes.
- Está disponible en múltiples plataformas como Windows, Linux, todas las versiones de Unix ofrecidas por diversas empresas como IBM, Sun, Digital, HP, Sequent, etc. y también en VAX-

VMS, así como en MVS. El ambiente multiplataforma de Oracle, lo convierte en una verdadera solución empresarial.

- Bloqueo y concurrencia: El lector verá los datos tal y como estaban antes de que el que escribe comenzará a cambiarlos (hasta que este haga commit).
- Disponibilidad. Soporta ambientes de cluster en modo activo-pasivo, es decir que un solo nodo utiliza la base de datos mientras el/los otro/s nodo/s está/n pendientes de entrar en funcionamiento en el momento que el servidor primario tenga una falla. Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal, esta funcionalidad se le denomina Oracle Standby Database. Las copias de la Base de Datos productiva pueden estar en modo de lectura solamente.

Desventajas:

- Es un software propietario, además su elevado precio (la BD más cara) hace que solo se vea en empresas muy grandes y multinacionales, por norma general.
- Elevado costo de soporte técnico
- Infinidad de cursos para poder manejar una sola línea (DBA, Desarrollo, Recursos Humanos, Minería de Datos, entre otras muchas).
- La seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos del 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre del 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Teniendo en cuenta que la investigación desarrollada es para la salud y esta está migrando a software libre, de acuerdo a todas las limitaciones antes mencionadas se decide no utilizar este sistema gestor de base de datos para el desarrollo de la investigación.

1.7.2 Microsoft SQL Server 2000

SQL Server es el Sistema manejador de Base de Datos Relacional ideal para un amplio espectro de clientes corporativos y productores independientes de software (ISV) inmersa en la creación de aplicaciones empresariales. Las necesidades y requisitos del cliente han dado lugar a innovaciones significativas en el producto SQL Server versión 2000, entre las que se incluyen la facilidad de uso, escalabilidad, fiabilidad, y almacenamiento de datos. SQL Server 2000 es la oferta completa de BD y análisis. Tanto por la capacidad para consultar la BD mediante un explorador como por la compatibilidad con el Lenguaje de marcado extensible (XML, Extensible Markup Language), SQL Server 2000 es la BD totalmente habilitada para Web.[14]

Ventajas:

- Soporta funciones definidas por el usuario. Esto soluciona los problemas de reutilización del código y da mayor flexibilidad al programar las consultas de SQL.
- Arquitectura de almacenamiento en disco permite la escalabilidad desde BD de equipos portátiles hasta bases de datos empresariales de tamaño de terabyte.
- Permite acceso y la realización de consultas desde URL a través de HTTP, donde el optimizador de consultas con múltiples fases busca el plan óptimo de consultas para mejorar el rendimiento de consultas complejas.
- Soporta consultas en múltiples servidores.
- Simplifica la configuración y gestión de un cluster de caídas. Permite que las BD permanezcan online durante la mayoría de las operaciones. Activa backups instantáneos.
- Estadísticas automáticas. Extrae estadísticas mediante el análisis rápido de una muestra, habilitando el optimizador de consultas para utilizar la información más reciente e incrementar la eficacia de las consultas.
- Los servicios de transformación de datos sirven para importar, exportar y transformar datos heterogéneos.
- Soporte enriquecido de XML. Los desarrolladores Web pueden acceder a los datos utilizando XML sin tener que emplear compleja programación, y los administradores de BD pueden manipular

datos fácilmente en formato XML mediante Transaction SQL (T-SQL) y procedimientos almacenados.

- Se instala con un alto nivel de seguridad por defecto, beneficiándose de la seguridad integrada de Microsoft Windows 2000. También presenta sofisticadas características de seguridad que incluyen seguridad de servidor de BD, de perfiles de aplicación potente y flexible basada en funciones; herramientas integradas para auditar la seguridad, soporte para la codificación de redes y archivos.
- Una nueva característica de búsqueda de texto completo (Full-Text Search (FTS)) permite que los usuarios consulten tanto documentos estructurados como no estructurados, incluyendo hojas de cálculo de Microsoft Excel, presentaciones de Microsoft PowerPoint, y documentos de Microsoft Word, potenciando la búsqueda online.
- Fiabilidad continúa. Permite actuar en modo Activo/Pasivo con un servidor como backup, o en modo Activo/Activo, que soporta varios servidores. El envío de logs totalmente integrado aporta una puesta en espera "amable" en varios servidores de backup moviendo automática y continuamente logs de una BD a otra. El soporte de backups diferenciales reduce significativamente el riesgo de pérdida de datos.[15]

Desventajas:

- Las funciones definidas por el usuario tienen algunas restricciones.
- No todas las sentencias SQL son válidas dentro de una función.
- Es un SGBD propietario.

1.7.3 MySQL

Es un SGBD cliente/servidor, multihilo y multiusuario. Se compone de un servidor SQL, varios programas clientes y bibliotecas, herramientas administrativas, y una gran variedad de interfaces de programación (APIs). Se puede obtener también como una biblioteca multihilo que se puede enlazar dentro de otras aplicaciones para obtener un producto más pequeño, más rápido, y más fácil de manejar. Es el servidor de BD relacionales más popular, desarrollado y proporcionado por MySQL AB. Esta es una empresa cuyo negocio consiste en proporcionar servicios en torno al servidor de BD MySQL.

MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual. Cualquier empresa o persona interesada puede descargar el software de MySQL de Internet y usarlo sin pagar por ello. Además, cualquiera que lo necesite puede estudiar el código fuente y cambiarlo de acuerdo a sus necesidades. Usa la licencia GPL (Licencia Pública General GNU), para definir qué es lo que se puede y no se puede hacer con el software para diferentes situaciones. Sin embargo, si no se está de acuerdo con la licencia GPL o tiene la necesidad de incorporar código de MySQL en una aplicación comercial es posible comprar una versión con una licencia comercial.[16]

Ventajas:

- Dispone de borrados multi-tablas. Al especificar múltiples tablas y la cláusula WHERE, se pueden agregar opciones ORDER BY y LIMIT a las consultas DELETE, para obtener un mejor control sobre cuántos registros son eliminados y el orden en el que son eliminados dichos registros.
- Mejores utilidades de administración (backup, recuperación de errores, etc.), contando con un sistema de replicación multihilo en los servidores esclavos. Si el servidor principal llega a fallar, es ahora mucho más probable que cada esclavo tenga los datos necesarios para hacer por sí mismo una recuperación de los datos y trabajar como si fuera el servidor maestro. Los registros de replicación ahora contienen los marcadores de transacción necesarios para asegurarse que las transacciones son replicadas apropiadamente.
- Soporta cinco tipos de tablas: MyISAM, ISAM, HEAP, BDB (Base de Datos Berkeley), e InnoDB. InnoDB y BDB son tablas transaccionales. Se puede utilizar la sentencia estándar BEGIN WORK seguida de varias consultas y finalizar con un COMMIT o ROLLBACK para completar la transacción ó se pueden correr en modo AUTOCOMMIT, así que cada consulta es efectivamente una transacción separada.
- Recuperación automática ante fallas. Si MySQL se cuelga o se da de baja de una forma anormal, no suele perder información ni corromper los datos, además InnoDB automáticamente completará las transacciones que quedaron incompletas.
- Integridad referencial. Ahora se pueden definir llaves foráneas entre tablas InnoDB relacionadas para asegurarse de que un registro no puede ser eliminado de una tabla si aún está siendo referenciado por otra tabla.

- Bloqueo a nivel de filas. Al usar tablas MyISAM, y tener consultas muy grandes que requieren de mucho tiempo, simplemente no se podían ejecutar más consultas hasta que terminaran las consultas que estaban en ejecución. En cambio, las tablas InnoDB usan bloqueo a nivel de filas para mejorar de manera impresionante el rendimiento.
- SELECTs sin bloqueo. El motor InnoDB usa una técnica conocida como multi-versioning que elimina la necesidad de hacer bloqueos en consultas SELECT muy simples.
- Registros de longitud fija y variable, sin límites en el tamaño, soportando hasta 32 índices por tabla y diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma flotante, doble precisión, carácter, fechas, enumerados, etc.
- Puede trabajar en distintas plataformas y S.O. distintos, además es multiproceso, es decir puede usar varias CPU si éstas están disponibles y consume muy pocos recursos, tanto de CPU como memoria.
- El servidor MySQL fue desarrollado originalmente para manejar grandes BD mucho más rápido que las soluciones existentes y está muy estandarizado el uso de plataforma de desarrollo de aplicaciones Web LAMP (Linux, Apache, MySQL y PHP).

Desventajas:

- No es viable para su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad.
- Carece de soporte para transacciones, rollback's y subconsultas.

1.7.4 PostgreSQL

Es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley.

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostGreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostGreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.[17]

Desventajas:

- “Consume bastantes recursos y carga con mucha facilidad el sistema.
- Velocidad de respuesta un poco deficiente al gestionar BD relativamente pequeñas, aunque esta misma velocidad la mantiene al gestionar BD realmente grandes”.[18]

Ventajas:

- DBMS Objeto-Relacional. Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia, y arrays.
- Cliente/Servidor. Usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.
- Altamente Extensible. Soporta los tipos de datos base, así como: tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. Además operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- Soporte SQL Comprensivo. Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.
- Integridad Referencial. Es utilizada para garantizar la validez de los datos de la BD.
- Lenguajes Procedurales. Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Además tiene habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.
- MVCC (Multi-Version Concurrency Control) Control de Concurrencia Multi-Versión. Es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios, es decir permite la lectura sin que sea bloqueada por los que escriben que están actualizando registros.
- Write Ahead Logging (WAL) Esta característica incrementa la dependencia de la BD al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en caso de que

la BD se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la BD desde el punto en que se quedó.

- Es un gestor magnífico bajo licencia Berkeley Software Distribution (BSD), que posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios Web que posean alrededor de 500.000 peticiones por día. Además por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.

Por todo lo anterior expuesto se decide utilizar este gestor de BD (PostgreSQL) ya que es Open Source bajo licencia BSD, es decir, sus potencialidades están en constante perfeccionamiento, permitiendo su uso y distribución sin costo; además continuamente se eliminan y mejoran cualquier hueco de seguridad que pueda aparecer, debido a que sus usuarios pueden acceder a su código y modificarlo a sus necesidades. En el caso que se esta trabajando con sistemas serios, la integridad referencial de los datos es muy buena, se pueden crear funciones complejas para validar datos. Es también utilizado porque es un SGBD potente y multiplataforma.

1.8 Metodologías

1.8.1 XP (Extreme Programing)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo y corto equipo. Consiste en una programación rápida o extrema, pues uno de los requisitos para llegar al éxito del proyecto es tener como parte del equipo al usuario final.

- “Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.
- Se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento”.[19]

1.8.2 Rational Unified Process (RUP)

La metodología de desarrollo empleada para la realización de esta investigación es RUP, llamada así por sus siglas en inglés Rational Unified Process, la cual organiza el proceso de desarrollo de software en cuatro fases y nueve flujos de trabajo. (Ver figura 1.1)

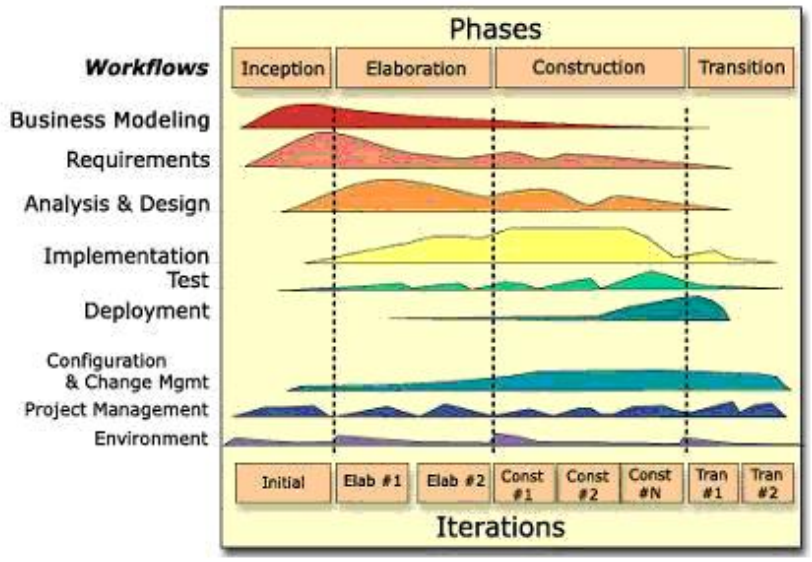

Figura 1.1 Fases e iteraciones de la metodología RUP

El ciclo de vida de esta metodología se caracteriza por:

- "Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).
- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura.

- Iterativo e Incremental: RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto".[19]

Se selecciona la metodología RUP porque es la que mejor se ajusta a la investigación del proyecto en cuestión. El rol que se desarrollará es el de diseñador principal de la BD del proyecto (responsable del diseño de la BD), jugando su papel principal en el flujo de trabajo de análisis y diseño en la fase de elaboración; construyendo el diagrama de clases persistentes y modelo de datos para el desarrollo de la BD, siendo esta metodología la que permite llegar a tan alto nivel. Es la más adaptable para proyectos a largo plazo, además de realizar análisis de riesgos y prever las acciones a realizar en cada caso.

1.9 Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering) son herramientas de software que automatizan tareas particulares del ciclo de vida del software.[20] Son la base para el proceso de análisis y desarrollo de software. Brindan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (Investigación Preliminar, Análisis, Diseño, Implementación e Instalación).

1.9.1 ERwin

ERwin es una herramienta CASE para el modelado y diseño de BD, que brinda productividad en su diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la BD diseñada, además ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la BD. Genera automáticamente las tablas y miles de líneas de procedimientos almacenados y disparadores para los principales tipos de base de datos.

ERwin hace fácil el diseño de una BD. Los diseñadores de BD sólo apuntan y pulsan un botón para crear un gráfico del modelo E-R (Entidad-Relación) de todos sus requerimientos de datos y capturar las reglas de negocio en un modelo lógico, mostrando todas las entidades, atributos, relaciones, y llaves importantes.

La migración automática garantiza la integridad referencial de la BD. ERwin establece una conexión entre una BD diseñada y una BD, permitiendo transferencia entre ambas y la aplicación de ingeniería reversa. Usando esta conexión, ERwin genera automáticamente tablas, vistas, índices, reglas de integridad referencial (llaves primarias, llaves foráneas), valores por defecto y restricciones de campos y dominios. Mantiene los recursos y mejora la precisión al sincronizar el modelo y la base de datos. [21]

ERwin soporta principalmente bases de datos relacionales SQL y bases de datos que incluyen Oracle, Microsoft SQL Server, Sybase. El mismo modelo puede ser usado para generar múltiples bases de datos, o convertir una aplicación de una plataforma de base de datos a otra.

Requerimientos Técnicos:

Mínimo 10 MB de espacio de disco duro, 16 MB RAM (32 MB RAM recomendado para modelos largos.)

1.9.2 Case Studio

Case Studio es una herramienta profesional para el diseño de base de datos. Soporta Oracle, MySQL, MS SQL, MaxDB, Firebird, PostgreSQL y otros sistemas [22]. El programa permite generar rápidamente diagramas gráficos de las bases de datos relacionales con las que se trabaja, simplificando mucho el trabajo del programador. Permite realizar Diagramas Entidad-Relación (DER) y Diagramas de Flujos de Datos (DFD) para distintos motores de base de datos

Su principal característica, es su potente sistema de ingeniería inversa, o sea, a partir del modelo de tablas llegar al modelo lógico. Permite identificar y estructurar BD ya existentes para poder trabajar con ellas sin problemas.

Su intuitiva interfaz gráfica hace fácil las tareas más complicadas. Mediante Case Studio se podrá realizar diagramas de flujo con muy poco tiempo y esfuerzo. Permite la generación de disparadores (triggers) para realizar validaciones en las entidades que formarán parte de las tablas de la BD.

Requisitos Mínimos

Win95/98/98SE/NT/ME/2000/XP
Procesador Pentium o Superior 64MB de RAM 16 MB de Disco Duro

Tipo de Licencia: Freeware

Tamaño: 7010 Kb

De las herramientas CASE mencionadas anteriormente, para el desarrollo de esta investigación se utiliza para el modelado de la BD la herramienta Case Studio 2, herramienta que aunque es nueva en el mercado soporta mayor número de SGBD a diferencia de Erwin, entre ellos, PostgreSQL, además trabaja en modelo físico, es decir, se elimina el modelo lógico utilizado en Erwin como punto de partida para el físico.

1.9.3 Rational Rose 2003

Para el desarrollo de esta investigación se decidió utilizar de la Suite del Rational 2003 la herramienta de modelado Rational Rose Enterprise Edition, para la realización del diagrama de clases persistentes.

Rational Rose 2003 provee a los desarrolladores de una de las metodologías estándar más utilizadas para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Además es reconocido como el líder tecnológico por su rol en el desarrollo del UML, logrado en gran parte por los esfuerzos de Grady Booch, Ivar Jacobson, y Jim Rumbaugh, los tres más importantes autores del UML. [23]

Abarca también todas las etapas de desarrollo de software. Permite además generación de código PHP mediante la utilización de RosePhpTool, plugins incorporado de manera independiente durante el desarrollo del proyecto. Permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Rational Rose acelera el diseño de bases de datos gracias a un sofisticado entorno de modelado visual.

1.10 Herramientas de desarrollo

1.10.1 SQL Manager 2005 for PostgreSQL

Se utiliza EMS SQL Manager for PostgreSQL ya que es una aplicación de alto desempeño para la administración y desarrollo de los servidores de BD para PostgreSQL. El programa trabaja con cualquier versión de PostgreSQL superior a la 8.0 y soporta las últimas características de PostgreSQL, incluyendo espacios de tablas, nombres de argumentos en funciones y más. Su interfaz gráfica es sumamente atractiva e incluye un modo guiado de trabajo.[24]

Características:

- Soporte completo para PostgreSQL superior a la versión 8.1

- Administración y navegación rápida de BD.
- Administración fácil de todos los objetos PostgreSQL.
- Administración efectiva de seguridad.
- Capacidades de exportación e importación de datos.
- Modo guiado para labores de mantenimiento.
- Interfaz atractiva.

1.10.2 EMS Data Generator 2005 for PostgreSQL

Es una herramienta potente para la generación de datos de prueba para BD PostgreSQL, brindando la posibilidad de generar datos para varias tablas a la vez. La aplicación asistente permite seleccionar las tablas y campos que serán llenados, permitiendo definir rango de valores, generar campos de tipo *char* utilizando máscara, además de cargar valores ya almacenados para campos *BLOB*, así como obtener los valores a almacenar de consultas SQL. Además permite definir plantillas de valores para su utilización futura, los cuales serán generados con solo cargar la misma. (Ver figura 1.2)

Características:

- El Asistente posee una interfaz de usuario amigable.
- Genera datos para varias BD diferentes, ubicadas en un mismo cliente.
- Soporta todos los tipos de datos que posee PostqreSQL, como: arreglos, direcciones de red y tipos geométricos.
- Genera datos de diferentes formas para cada campo, ya sea desde una lista de valores predefinida, valores aleatorios haciendo uso de caracteres previamente definidos o valores incrementales.
- Permite utilizar el resultado de consultas SQL como lista de valores para la generación de datos.
- Control automático sobre la integridad referencial durante la generación de datos para tablas relacionadas, validando el cumplimiento de la misma.
- Amplia variedad de parámetros de generación para cada tipo de campo.

- Permite generar valores nulos para un por ciento definido de los valores generados para cada campo.
- Realiza un almacenamiento de todos los parámetros degeneración en la sesión del asistente en uso.
- Permite la generación de datos haciendo uso de plantillas, a través de la utilidad *command-line*.[25]

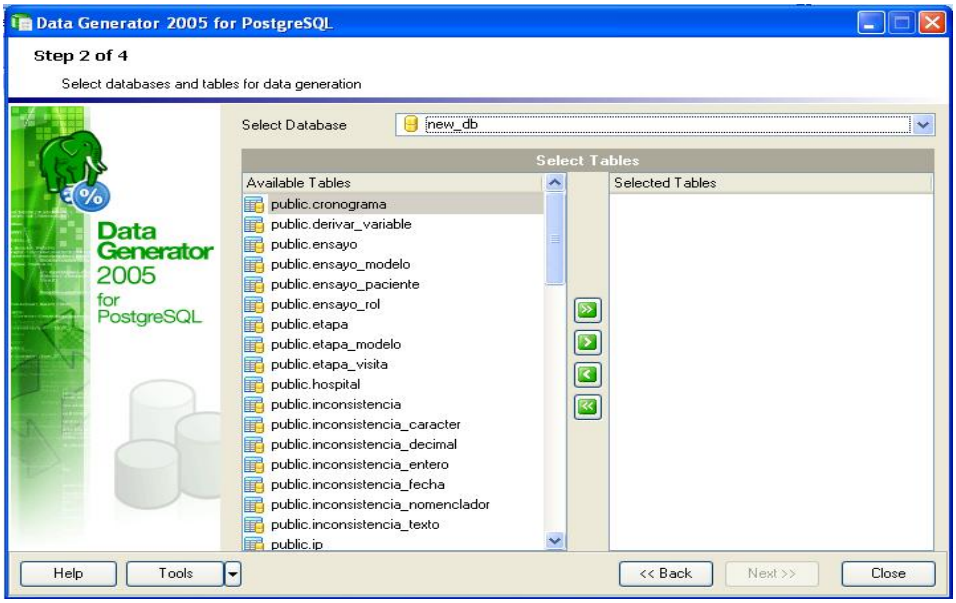

Figura 1.2 EMS Data Generator 2005 for PostgreSQL

1.10.3 EMS SQL Query 2005 for PostgreSQL.

Es una herramienta que permite, de manera rápida y sencilla, construir consultas SQL para BD PostgreSQL. Permite la construcción visual así como la edición directa de un texto de consulta. Su interfaz gráfica amigable permite la conexión a la BD, la selección de campos y tablas para una consulta y poner el criterio de selección. Se puede trabajar con varias consultas al mismo tiempo, revisar consultas y ver los resultados de la ejecución en diferentes modos y realizar cualquier otra operación que se necesite en el trabajo con las consultas a BD.

Características:

- Tiene la posibilidad de trabajo con varias consultas en ventanas separadas.

- Muestra historia de consultas usadas recientemente.
- Posee la capacidad de crear diagramas basados en consultas.
- Presenta la posibilidad de crear consultas con unión y subconsultas visualmente.[26]

Conclusiones

Después de realizar un estudio exhaustivo del tema de base de datos, así como algunos conceptos de interés necesarios para el entendimiento de esta investigación se concluye con la necesidad de Diseñar e implementar una base de datos para el almacenamiento de los Ensayos Clínicos en la que:

- Se ha escogido el modelo de base de datos relacional.
- Se escogió como metodología de desarrollo RUP y UML como lenguaje de modelado, para la realización del diagrama de clases persistentes.
- La herramienta CASE escogida para el diseño de la base de datos (diagrama entidad-relación) es Case Studio 2, y para la realización del diagrama de clases persistentes, de la suite de Rational 2003: Rational Rose Enterprise Edition.
- El SGBD a utilizar es PostgreSQL, utilizando para el desarrollo y administración de la BD: EMS SQL Manager for PostgreSQL, para la generación de datos de prueba EMS Data Generador for PostgreSQL y en la elaboración de consultas más frecuentes se utilizó EMS SQL Query 2005 for PostgreSQL.

Capítulo 2

Análisis y Descripción de la Solución Propuesta

En el presente capítulo se describe la solución propuesta, haciendo énfasis en los requisitos funcionales y no funcionales, en el diagrama de clases persistentes obtenido a partir del diagrama de clases del diseño, todas estas cuestiones referentes al trabajo del analista. Se hará una descripción de todas las clases, el diseño de la base de datos, incluyendo el diagrama entidad relación y la descripción de aquellas tablas importantes. También se hará una pequeña fundamentación de la descripción de la arquitectura propuesta.

2.1 Requisitos Funcionales y No Funcionales.

La definición de las necesidades del sistema es un proceso complejo, pues en él hay que identificar los requisitos que el sistema debe cumplir para satisfacer las necesidades de los usuarios finales y de los clientes.

El tratamiento de requisitos es el proceso mediante el cual se especifican y validan los servicios que debe proporcionar el sistema, así como las restricciones sobre las que se deberá operar. La importancia de esta fase es esencial puesto que los errores más comunes y más costosos de reparar, así como los que más tiempo consumen se deben a una inadecuada ingeniería de requisitos.

Los requisitos se dividen en dos grupos: Requisitos Funcionales y No Funcionales. Donde los primeros son capacidades o condiciones que el sistema debe tener y los segundos son propiedades o cualidades que el producto debe cumplir.

2.1.1 Requisitos Funcionales

A continuación se referencia una lista de requisitos funcionales, propuestos por los analistas, que representan lo que el sistema debe hacer para satisfacer las necesidades del usuario.

R1. Insertar Ensayo.

R2. Modificar datos del Ensayo Clínico

R3. Seleccionar ensayo(s) clínico(s).

R4. Seleccionar los roles asignados a un ensayo clínico.

R5. Insertar Sitio

R6. Modificar Sitio

R7. Eliminar sitio (si no se ha usado)

R8. Seleccionar sitio(s).

R9. Insertar hospital

R10. Modificar hospital

R11. Eliminar hospital (Se elimina un hospital en caso de que en él no se encuentre algún sitio que posea algún Ensayo Clínico activo).

R12. Seleccionar hospital(es).

R13. Insertar usuario

R14. Modificar datos del usuario

R15. Eliminar usuario (si no posee rol en algún ensayo).

R16. Seleccionar usuario(s)

R17. Seleccionar roles asignados a un usuario.

R18. Insertar asignación de rol a los usuarios en un sitio y grupos de IP dentro de un EC.

R19. Modificar asignación de rol

R20. Eliminar asignación de rol (si no esta asignado a un usuario).

R21. Seleccionar rol

R22. Seleccionar listado de roles asignados.

R23. Insertar Modelo

R24. Modificar Modelo

R25. Eliminar modelo (si no forma parte de un ensayo activo).

R26. Seleccionar modelos.

R27. Insertar submodelo

R28. Modificar submodelo

R29. Eliminar submodelo (si no forma parte de un ensayo clínico activo).

R30. Seleccionar submodelos.

R31. Seleccionar ensayos clínicos en los que se ha usado un submodelo

R32. Insertar variable

R33. Modificar variable.

R34. Eliminar variable (si no forma parte de un ensayo clínico activo).

R35. Seleccionar variable(s).

R36. Seleccionar valores de la variable

R37. Seleccionar nombres de una variable o los valores de un nomenclador.

R38. Seleccionar submodelos en los que se ha usado una variable

R39. Seleccionar ensayos clínicos en los que se ha usado una variable.

R40. Seleccionar diseños en elaboración

R41. Seleccionar modelos de un ensayo específico.

R42. Seleccionar cronograma de ejecución de un diseño.

R43. Insertar reglas de validación.

R44. Seleccionar visitas donde aparezca una variable determinada según el modelo y el submodelo al que pertenece.

R45. Insertar tiempo de duración del ensayo.

R46. Insertar visitas.

R47. Seleccionar visitas registradas.

R48. Insertar etapas del cronograma.

R49. Seleccionar etapas.

R50. Insertar paciente.

R51. Seleccionar pacientes de un ensayo

R52. Seleccionar inconsistencia por paciente (Id_paciente, resumen de inconsistencia).

R53. Seleccionar inconsistencia por modelo (Id_modelo, resumen de inconsistencia).

R54. Seleccionar queries por paciente (Mostrar listado de queries por estado, visita, monitor, descripción).

R55. Seleccionar queries por modelos (Mostrar listado de queries por estado, visita, monitor, descripción).

R56. Modificar paciente

R57. Seleccionar traza de auditoria.

R58. Seleccionar trazas por submodelo, fecha, hora, acción (creación, modificación, eliminación, consulta), variable).

R59. Mostrar listado de visitas de un ensayo por fecha, estado, inconsistencias, queries)

R60. Mostrar cronograma de ejecución del ensayo (mostrar los modelos, fechas de las visitas correspondientes).

R61. Mostrar cronograma de ejecución del ensayo para un paciente específico.

R62. Mostrar cronograma de ejecución del ensayo para un paciente específico.

R63. Insertar query por variable.

R64. Modificar query por variable.

R65. Seleccionar un usuario con el rol de monitor según(nombre, ensayo, sitio)

2.1.2 Requisitos no Funcionales

1. RNF de usabilidad

   El acceso a cualquier manipulación del sistema, tanto entrada como análisis de datos debe estar sometido a un proceso de autentificación por firma electrónica.

2. RNF Rendimiento

   El sistema será capaz de dar respuestas a las peticiones hechas al servidor de forma rápida. En caso de fallo o presentar algún problema poseerá rápido tiempo de recuperación.

   Los cálculos que se necesiten para dar acceso a los datos de los Ensayos Clínicos serán rápidos y eficientes, dando respuesta a las peticiones de los clientes con la mayor precisión posible.

   Los datos de los Ensayos Clínicos deberán estar almacenados durante 15 años como mínimo.

   Solo el administrador podrá eliminar algún dato pero con previa autorización.

3. RNF de Soporte

   Una vez terminada la aplicación se instalará en CIM para realizar pruebas piloto del software

   Una vez aprobada la aplicación y que este lista para comenzar su ejecución se instalará en los Centros del polo científico y hospitales donde se realicen ensayos Clínicos.

   Cada un tiempo previsto por los administradores del sistema se realizará el mantenimiento del SW

4. RNF de Portabilidad

   El sistema podrá ser usado en los sistemas operativos de Windows y GNU Linux.

5. RNF de Seguridad

   El acceso a cualquier manipulación del sistema, tanto entrada como análisis de datos debe estar sometido a un proceso de autentificación por firma electrónica.

   Cada usuario va a tener asignado uno o varios roles en el sistema

   Cada rol definido tendrá niveles de acceso al SW

   Todo cambio o modificación en el sistema tiene que ser atribuible (rastreable) a un usuario particular según su firma electrónica.

Paralelo a la base de datos primario se debe mantener una base de datos que registre todas las modificaciones hechas a la base de datos original, ordenadas cronológicamente y con una especie de firma temporal acuñada. Inclusive debe permitirse que el usuario explique el porqué de la modificación introducida y más aún debe guardarse la información previa a la modificación de modo que pueda ser inspeccionada posteriormente.

Los datos transmitidos a través de la red deben estar apropiadamente encriptados para asegurar la confidencialidad de los mismos.

La información obtenida del ensayo es propiedad del promotor y como tal solo puede autorizar la extracción e impresión de la misma por diferentes usuarios.

Todas las acciones de copia, extracción o impresión de información de la BD deberán ser registradas en la traza de auditoria.

Ninguna máquina que no pertenezca al centro donde se está usando el sistema tendrá acceso al sistema, y por tanto la información estará protegida de divulgación, corrupción, o inconsistencias.

Los mecanismos de seguridad no les impedirán a los usuarios obtener datos de los ensayos clínicos en un momento dado.

6. RNF de Software

   Para poder instalar este sistema se debe disponer con un sistema operativo superior o igual al de Windows 98, o GNU Linux. En las computadoras de los clientes también deberán existir las mismas restricciones de los Sistemas Operativos.

7. RNF de Hardware

   Las PC de los clientes deberán tener 40 GB o superior, microprocesador superior a 1.0 GHz de velocidad, 256 MB mínimo de RAM.

   En los grupos de la dirección de calidad se deberán montar más PCs con capacidad en las PCs Servidora de 60 GB o superior, microprocesador superior a 1.0 GHz de velocidad, 256 MB mínimo de RAM.

8. RNF Restricciones en el Diseño y la Implementación

Como herramienta CASE Rational Suite 2003 para el modelado de los artefactos que se generen en cada uno de los flujos de trabajo, y Case Studio.

Se utiliza UML como lenguaje de modelado.

Se usará como Gestor de Base de Datos PostgreSQL.

2.2 Estrategia de integración de la solución con otros módulos y sistemas.

El sistema de manejo de datos para ensayos clínicos esta estructurado en cuatro módulos, los cuales abarcan las funcionalidades principales para el desarrollo de un EC, encargados del proceso de recogida de datos de los CRD y almacenarlos en la BD, que luego utilizarán para el análisis estadístico. Estos tendrían una interacción de forma indirecta con la BD física, es decir, todas las peticiones se realizarían a través de clases de acceso a datos que posibilita separar el acceso a la base de datos desde la aplicación en sí.

2.3 Descripción de la arquitectura y fundamentación.

La BD propuesta contiene 56 tablas en modelo físico, las cuales serán utilizadas por los cuatro módulos del proyecto: Administrador, Diseñador, Publicador y Monitoreo, haciendo uso principalmente de las tablas *ensayo, paciente, modelo, submodelo, variable, sitio, usuario, hospital*; las cuales recogen la información de mayor peso del EC. Por otra parte el sistema se encuentra estructurado en tres capas siguiendo el patrón de arquitectura layers (capas), de ahí que exista una capa de acceso a datos encargada de gestionar la información.

2.4 Clases Persistentes. Diagrama de Clases Persistentes.

Las clases persistentes por lo general tienen como origen las clases clasificadas como entidad porque ellas modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso. Todas las clases identificadas en el dominio del análisis no son persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo.

El Diagrama de Clase se utiliza para modelar la estructura lógica de la BD, con clases representando tablas, y atributos de clase representando columnas. Las clases persistentes y sus atributos hacen referencia directamente a las entidades lógicas y a sus atributos.

Para un análisis del diagrama de clases persistentes remitirse al expediente del proyecto (Anexo 1), en este se representan las clases persistentes de la BD en su totalidad, las cuales son utilizadas por los módulos: Publicador y Monitoreo. Además se ha dividido el mismo en diagramas más pequeños que agrupan las clases persistentes relevantes para los restantes módulos: Administrador (Ver figura 2.1) y Diseñador (Ver figura 2.2).

Módulo: Administrador

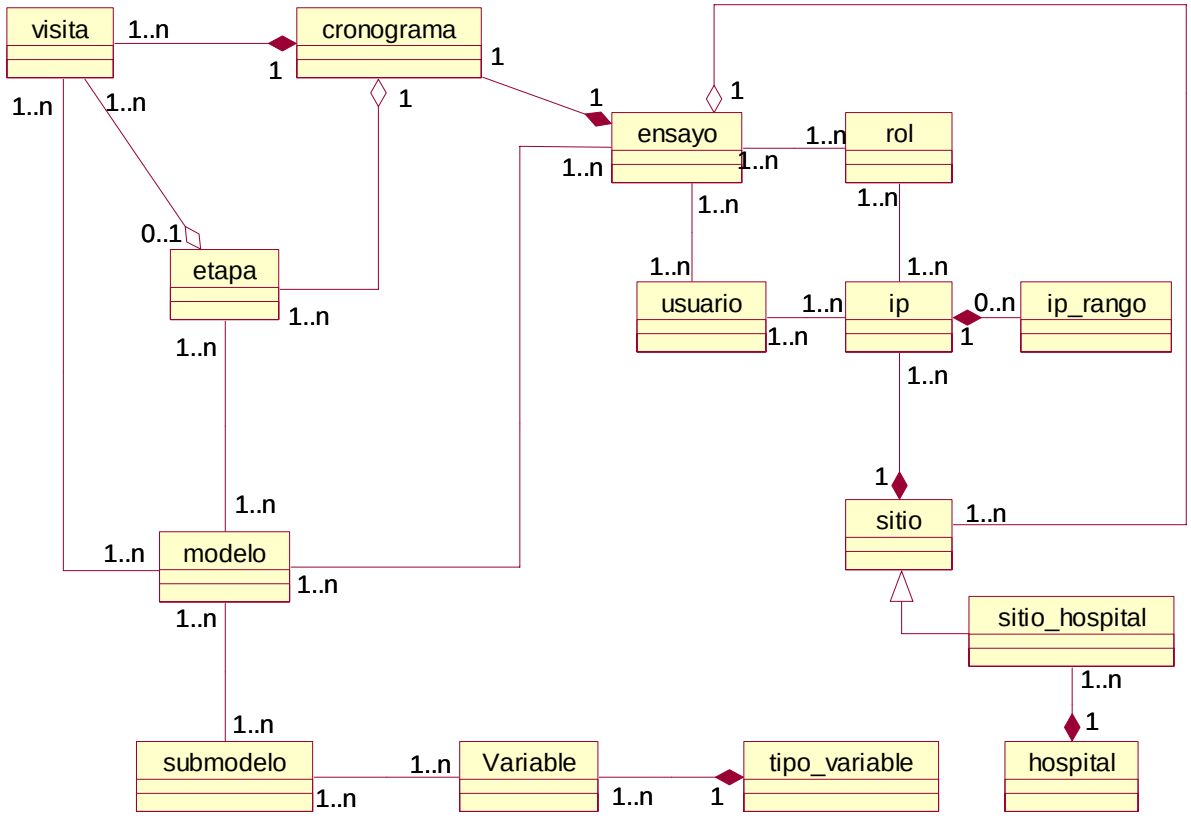

Figura 2.1 Diagrama de Clases Persistentes del Módulo Administrador.

Módulo: Diseñador

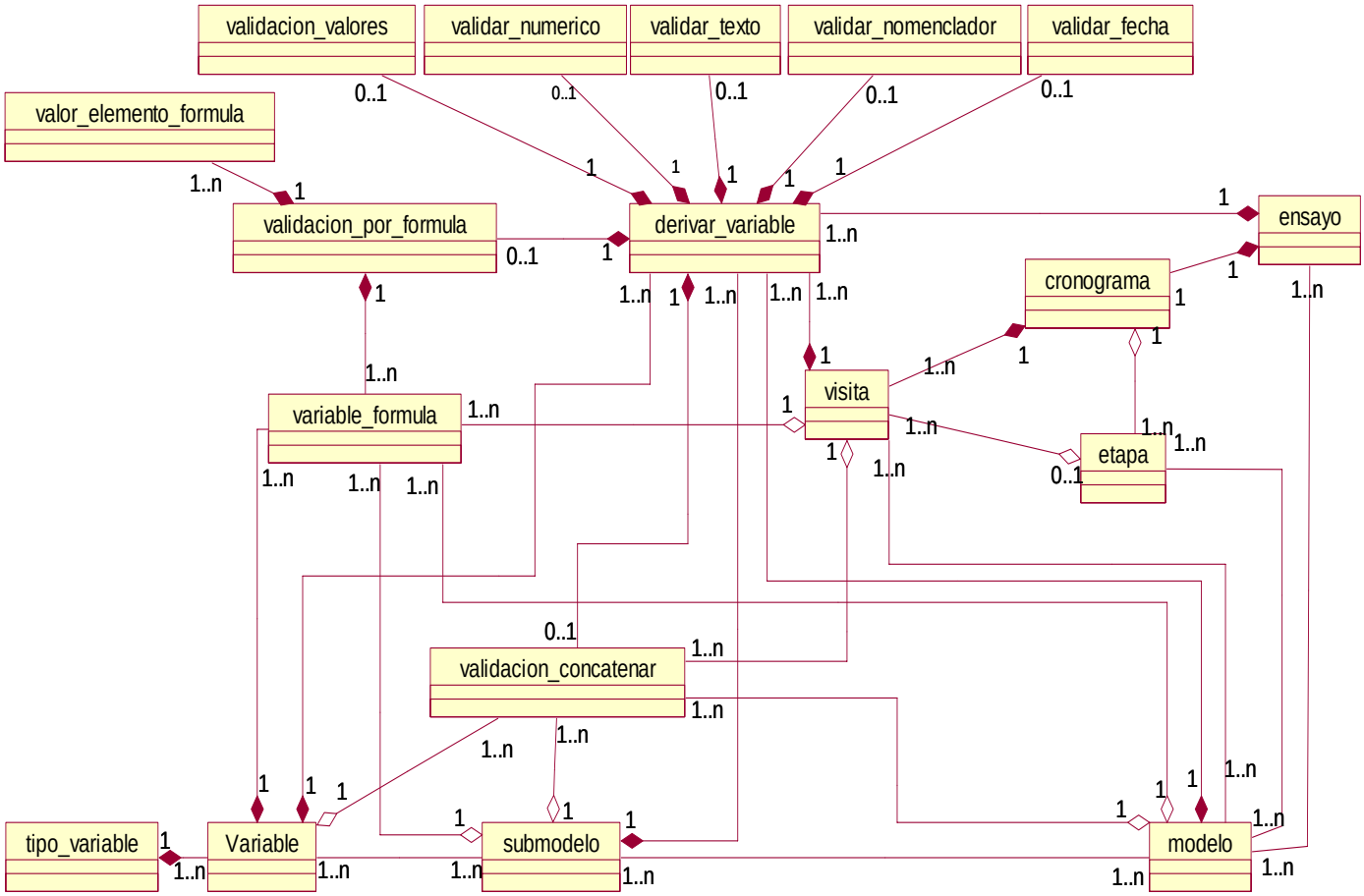

Figura 2.2 Diagrama de Clases Persistentes del Módulo Diseñador.

2.4.1 Descripción de las Clases Persistentes.

La descripción de las clases persistentes muestran los atributos con el tipo de dato que representan cada uno para un mejor entendimiento.

Nombre: ensayo	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_ensayo	Integer
nombre_ensayo	String
alias_ensayo	String

cod_ensayo	String
indicacion_ensayo	String
dia_creado_ensayo	Integer
mes_creado_ensayo	Integer
anno_creado_ensayo	Integer
dia_terminado_ensayo	Integer
mes_terminado_ensayo	Integer
anno_terminado_ensayo	Integer
centro_promotor_ensayo	String
producto_ensayo	String
estado_ensayo	String
fase_ensayo	String
descripción_ensayo	String

Nombre: modelo	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_modelo	Integer
estado_modelo	String
nombre_modelo	String
cant_submods	Integer
descripción_modelo	String

Nombre: modelo_datos	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_modelo_llenado	Integer
dia_llenado_modelo	Integer
mes_llenado_modelo	Integer
anno_llenado_modelo	Integer
estado_modelo	String
firma	String

Nombre: submodelo	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_submodelo	Integer

nombre_submodelo	String
tipo_submodelo	String
estado_submodelo	String
cant_variables	Integer
descripción_submodelo	String

Nombre: tipo_variable	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_tipo_var	Integer
tipo_variable	String

Nombre: variable	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_variable	Integer
nombre_variable	String
estado_variable	String
pertenencia	Integer
nomenclador_identificador	Boolean
descripción_variable	String

Se describieron las clases *var_entero, var_texto, var_decimal, var_fecha, var_caracter, var_nomenclador*, las cuales son una especialización de la clase *variable*, la clase *sitio_hospital*, que es una especialización de la clase *sitio* y las clases *query_variable y query_inconsistencia_valor*, las cuales son una especialización de la clase *query*. (Ver Anexo 1).

Nombre: sitio	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_sitio	Integer
nombre_sitio	String
telefono_sitio	String
direccion_sitio	String
provincia_sitio	String
tipo_sitio	String
descripcion_sitio	String

Nombre: hospital	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_hospital	Integer
cod_hospital	String
nombre_hospital	String
direccion_hospital	String
provincia_hospital	String
telefono_hospital	String
especificación_hospital	String

Nombre: usuario	
Tipo de clase: Entidad	
Atributo	**Tipo**
usuario_login	String
nombre_usuario	String
apellidos_usuario	String
contrasenna	String
correo_usuario	String
no_ci_usuario	Integer
telefono_usuario	String
afiliación_usuario	String
cargo_usuario	String
firma_usuario	String

Nombre: rol	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_rol	Integer
nombre_rol	String
descripción_rol	String

Nombre: ip	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_ip	Integer
direccion_ip	Inet
rango	Boolean

Nombre: ip_rango	
Tipo de clase: Entidad	
Atributo	**Tipo**
ip_final	Inet

Nombre: paciente	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_paciente	Integer
id_historia_clinica	String
iniciales_paciente	String
descripción_paciente	String

Nombre: etapa	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_etapa	Integer
nombre_etapa	String
inicio_etapa	Integer
fin_etapa	Integer
cant_visitas	Integer
descripción_etapa	String

Nombre: visita	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_visita	Integer
dia_visita_ensayo	Integer

hora_visita	Time
tipo_visita	String
intervalo_visita	Integer
duracion_visita	Integer
rango_llenado	Integer

Nombre: cronograma	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_cronograma	Integer
tiempo_duracion_cronograma	Integer
descripción_cronograma	String

Nombre: query	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_query	Integer
tipo_query	String
autor_query	String
dia_query	Integer
mes_query	Integer
anno_query	Integer
estado_query	String
historial_query	String
descripción_query	String

Nombre: derivar_variable	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_validacion_variable	Integer
estado_validacion	String

También se describen las clases que forman parte de la clase *derivar_variable*, es decir, los diferentes tipos en que se puede derivar o validar una variable. (Ver Anexo 2).

Nombre: pertenencia_valor	
Tipo de clase: Entidad	
Atributo	**Tipo**
cant_pertenencias	Integer

Nombre: inconsistencia	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_inconsistencia	Integer
tipo_inconsistencia	String
dia_inconsistencia	Integer
mes_inconsistencia	Integer
anno_inconsistencia	Integer
hora_inconsistencia	Time
descripción_inconsistencia	String

Nombre: traza_auditoria	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_traza	Integer
dia_traza	Integer
mes_traza	Integer
anno_traza	Integer
hora_traza	Time
usuario_realiza_accion	String
rol_usuario	String
ensayo_bajo_accion	String
modelo_ensayo	String
submodelo_ensayo	String
paciente_ensayo	String
sitio_traza	String
hospital_traza	String
accion	String
resultado_accion	String
variable_bajo_accion	String
valor_anterior	String
valor_nuevo	String
descripción_accion	String

2.5 Diseño de la BD.

La BD para el Sistema de Manejo de Datos de Ensayos Clínicos (SIMDEC) es el medio para el almacenamiento de todos los datos relacionados con los EC (datos, imágenes, CRD, variables), y para el acceso controlado a los mismos.

2.5.1 Diagrama Entidad-Relación

El modelo conceptual más utilizado para el diseño de BD es el diagrama entidad-relación, capaz de mostrar una visión más amplia de como trabajará el sistema y la información a almacenar en el mismo, además expresa entidades relevantes para un sistema de información, sus interrelaciones y propiedades. Los diagramas entidad-relación son una necesidad para diseñar y mantener una buena BD relacional. Son una forma gráfica para representar una Base de Datos. [27]

El diagrama de la base de datos fue realizado en la herramienta Case Studio, versión 2.18, la cual genera de forma automática el modelo físico de la BD. Se decidió por cuestiones del proyecto generar el modelo en esta herramienta porque a la hora de generar el modelo de datos teniendo el diagrama de clases persistentes del Rational Rose no lo genera para PostgreSQL, que es el gestor con el cual se trabaja.

Para un mejor entendimiento del diseño de la BD remitirse al expediente del proyecto (Anexo 2) donde se muestra el diagrama E-R de la BD para el SIMDEC, el cual modela todas las entidades que utilizan los módulos: Publicador y Monitoreo. Además se ha dividido el Diagrama E-R, agrupando en cada sub-diagrama las tablas más importantes para los restantes módulos: Administrador (Ver figura 2.3) y Diseñador (Ver figura 2.4).

Módulo: Administrador

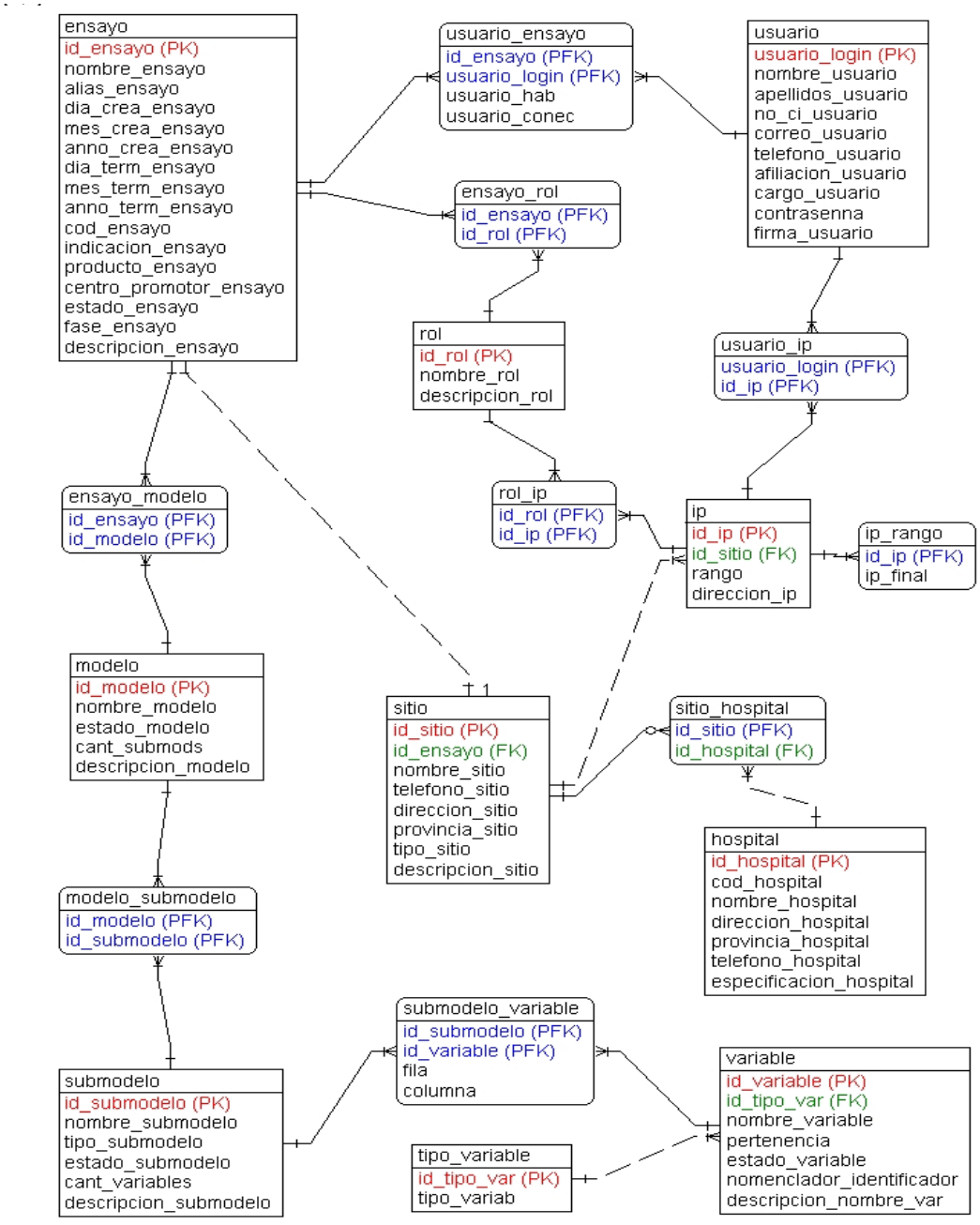

Figura 2.3 Diagrama E-R para el Módulo Administrador

Módulo: Diseñador

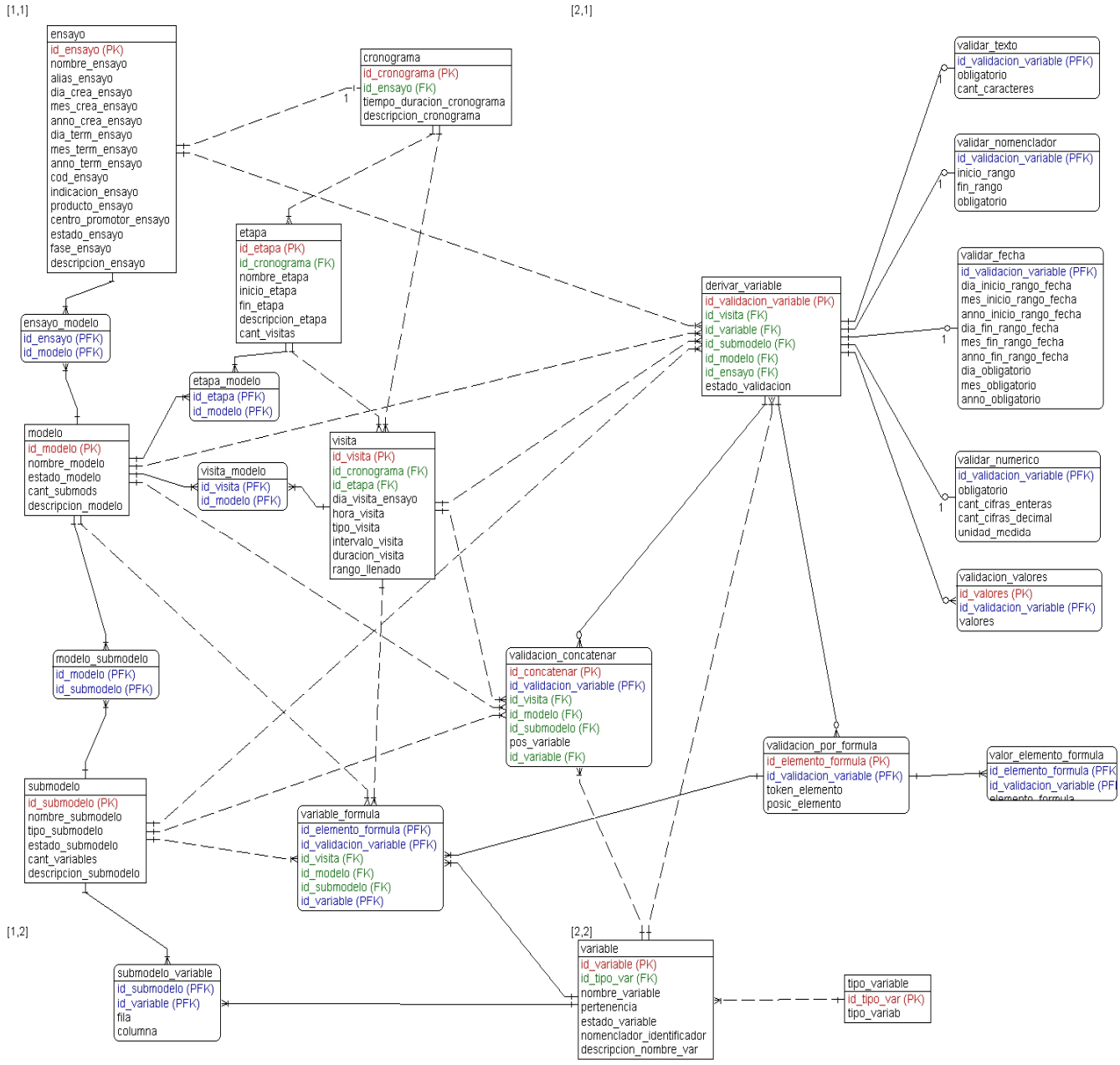

Figura 2.4 Diagrama E-R para el Módulo Diseñador

2.5.2 Descripción de las tablas de la base de datos EC.

Nombre: ensayo		
Descripción: Representa la entidad ensayo, donde se van a recoger datos relacionados con los ensayos clínicos		
Atributo	**Tipo**	**Descripción**
id_ensayo	Serial	Valor entero auto incremento, llave primaria
nombre_ensayo	Varchar	Surge de la unión del medicamento, parte del cuerpo y fase del Ensayo
alias_ensayo	Varchar(n)	Nombre más pequeño, puede ser null.
cod_ensayo	Varchar	Código único formado por letras y números.
indicacion_ensayo	Varchar	Lugar del cuerpo que se estudia.
dia_crea_ensayo	Integer	Dia de inicio de aplicación del EC
mes_crea_ensayo	Integer	Mes de inicio de aplicación del EC
anno_crea_ensayo	Integer	Año de inicio de aplicación del EC
dia_term_ensayo	Integer	Dia de cierre del EC, puede ser nulo
mes_term_ensayo	Integer	Mes de cierre del EC, puede ser nulo
anno_term_ensayo	Integer	Año de cierre del EC, puede ser nulo
fase_ensayo	Varchar	Fase del EC
producto_ensayo	Varchar	Producto asociado al EC
centro_promotor_ensayo	Varchar	Centro que elabora los medicamentos
estado_ensayo	Varchar	elaboracion, activo, interrumpido o terminado
descripción_ensayo	Varchar	Brinda una descripción del EC, puede ser null

Nombre: modelo		
Descripción: Representa el modelo con sus submodelos asociados.		
Atributo	**Tipo**	**Descripción**
id_modelo	Serial	Valor entero auto incremento, llave primaria
nombre_modelo	Varchar	Nombre del modelo, debe se rúnico
estado_modelo	Varchar	Puede estar en elaboración o finalizado
cant_submods	Integer	Cantidad de submodelos asociados.
descripcion_modelo	Varchar	

Nombre: modelo_datos		
Descripción: Representa el modelo lleno, es decir, con sus datos.		
Atributo	**Tipo**	**Descripción**
id_modelo_lleno	Serial	Valor entero auto incremento, llave primaria
estado_modelo	Varchar	Puede ser completo, incompleto, monitoreado, firmado.
dia_lleno_modelo	Integer	Día de llenado el modelo

mes_lleno_modelo	Integer	Mes de llenado el modelo
anno_lleno_modelo	Integer	Año de llenado el modelo
firma	Varchar	Cadena de caracteres auto generada que identifica de forma única a cada usuario
id_paciente	Integer	Lave foránea que la toma de la entidad paciente.
id_modelo	Integer	Llave foránea que la toma de la entidad modelo.
id_sitio	Integer	Llave foránea que la toma de la entidad sitio.

Nombre: submodelo		
Descripción: Es lo que forma un modelo, es decir, tablas, chekbox, textbox, todas las partes que forman un modelo.		
Atributo	**Tipo**	**Descripción**
id_submodelo	Serial	Valor entero auto incremento, llave primaria
nombre_submodelo	Varchar	Nombre del submodelo
tipo_submodelo	Varchar	Tipo: tabla o grupo de variables
estado_submodelo	Varchar(n)	Puede estar en: elaboración o finalizado
cant_variables	Integer	Cantidad de variables asociadas al submodelo.
descripcion_submodelo	Varchar	Descripción de lo que recoge el submodelo.

Nombre: variable		
Descripción: Representa todas las variables asociadas al submodelo		
Atributo	**Tipo**	**Descripción**
id_variable	Serial	Valor entero auto incremento, llave primaria
nombre_variable	Varchar	Nombre identificador de la variable
pertenencia	Integer	Variable que pertenece a otra variable.
estado_variable	Varchar	Puede estar: activa, inactiva
nomenclador_identificador	Boolean	
descripción_nombre_var	Varchar	Breve descripción del nombre de la var.
id_tipo_var	Integer	Llave foránea que la toma de la entidad tipo_variable.

Nombre: tipo_variable		
Descripción: Representa el tipo de variable a la cual está asignada el submodelo.		
Atributo	**Tipo**	**Descripción**
id_tipo_variable	Serial	Valor entero auto incremento, llave primaria
tipo_variab	Varchar(n)	Entero, decimal, nomenclador, caracter, texto, fecha.

Nombre: usuario		
Descripción: Tabla que representa los usuarios.		
Atributo	**Tipo**	**Descripción**
usuario_login	Varchar (n)	Nombre único identificador de cada usuario utilizado para acceder al sistema
nombre_usuario	Varchar	Nombre del usuario
apellidos_usuario	Varchar	Apellidos del usuario
contrasenna	Varchar	Contraseña para entrar al sistema.
correo_usuario	Varchar	Dirección de correo, puede ser null.
no_ci_usuario	Varchar(n)(11)	Número de carnét de identidad del usuario, el cual es único
telefono_usuario	Varchar	Numero de teléfono, puede no tener.
afiliación_usuario	Varchar	Institución a la que pertenece.
cargo_usuario	Varchar	Cargo que ocupa el usuario
firma_usuario	Varchar	Cadena de caracteres auto generada que identifica de forma única a cada usuario

Nombre: rol		
Descripción: Tabla que recoge los datos del rol que tendrá cada usuario.		
Atributo	**Tipo**	**Descripción**
id_rol	Serial	Valor entero auto incremento, llave primaria
nombre_rol	Varchar	Ejemplo: Diseñador, Publicador,….
descripción_rol	Varchar	Descripción de lo que debe hacer el rol.

Nombre: sitio		
Descripción: Porciones en las que se divide un hospital para aplicar Ensayos Clínicos		
Atributo	**Tipo**	**Descripción**
id_sitio	Serial	Valor entero auto incremento, llave primaria
nombre_sitio	Varchar	Cadena de caracteres que identifica un sitio.
telefono_sitio	Varchar	Número de teléfono del sitio, si tiene.

direccion_sitio	Varchar	Lugar donde se encuentra el sitio.
provincia_sitio	Varchar	Provincia a la que pertenece
tipo_sitio	Varchar	H(pertenece a un hospital) o F(no pertenece a un hospital)
descripción_sitio	Varchar	Descripción del sitio.
id_ensayo	Integer	Llave foránea que la toma de la entidad ensayo.

Nombre: sitio_hospital		
Descripción: Tabla que muestra si el sitio pertenece a un hospital dado, es una especialización de la tabla sitio.		
Atributo	**Tipo**	**Descripción**
id_sitio	Integer	Llave primaria foránea que la toma de la entidad sitio
id_hospital	Integer	Llave foránea que la toma de la entidad hospital

Nombre: hospital		
Descripción: Tabla que representa la entidad hospital.		
Atributo	**Tipo**	**Descripción**
id_hospital	Serial	Valor entero auto incremento, llave primaria
cod_hospital	Varchar(n)	Identificado principalmente por las iniciales del nombre del hospital. Debe ser único.
nombre_hospital	Varchar	Nombre del hospital
direccion_hospital	Varchar	Dirección del hospital
provincia_hospital	Varchar(n)(100)	Provincia a la que pertenece el hospital
telefono_hospital	Varchar	Número de teléfono de hospital, si tiene.
especificación_hospital	Varchar	

Nombre: paciente		
Descripción: Muestra las características principales que se recogen a la hora de atender a un paciente.		
Atributo	**Tipo**	**Descripción**
id_paciente	Serial	Valor entero auto incremento, llave primaria
id_historia_clinica	Varchar(n)	Número de la historia clínica, debe ser único
iniciales_Paciente	Varchar	Esta compuesta por las iniciales del nombre(s) y apellidos

descripción_Paciente	Varchar	Puede ser null

Nombre: ip		
Descripción: ip que van a tener asignados los usuarios en un determinado ensayo		
Atributo	**Tipo**	**Descripción**
id_ip	Serial	Valor entero auto incremento, llave primaria
direccion_ip	Inet	Dirección IP (10.x.x.x).
rango	Boolean	true(representa un rango) y false(representa un IP independiente)
id_sitio	Integer	Llave foránea que la toma de la entidad sitio.

Nombre: ip_rango		
Descripción: ip que van a tener asignados los usuarios en un determinado ensayo		
Atributo	**Tipo**	**Descripción**
ip_final	Inet	Dirección final del ip
id_ip	Integer	Llave foránea que la toma de la entidad IP

Nombre: pertenencia_valor		
Descripción: Es una tabla que surge de la unión de varias tablas para tomar su llave y se le agrega un atributo.		
Atributo	**Tipo**	**Descripción**
cant_pertenencias	Integer	
id_submodelo	Integer	Llave primaria foránea que la toma de la entidad submodelo.
id_modelo	Integer	Llave primaria foránea que la toma de la entidad modelo.
id_paciente	Integer	Llave primaria foránea que la toma de la entidad paciente
id_sitio	Integer	Llave foránea que la toma de la entidad sitio.
usuario_login	Varchar(n)	Llave primaria foránea que la toma de la entidad usuario.

Nombre: traza_auditoria		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_traza	Serial	Valor entero auto incremento, llave primaria
dia _traza	Integer	Dia de ocurrencia de la acción
mes_traza	Integer	Mes de ocurrencia de la acción
anno_traza	Integer	año de ocurrencia de la acción
hora_traza	Time	Hora de ocurrencia de la acción.
usuario_realiza_accion	Varchar	Usuario que realiza la acción
rol_usuario	Varchar	Rol que juega en el Ensayo
ensayo_bajo_accion	Varchar	Nombre del Ensayo
modelo_ensayo	Varchar	Nombre del modelo del ensayo.
submodelo_ensayo	Varchar	Nombre del submodelo del ensayo.
paciente_ensayo	Varchar	Nombre del paciente del ensayo.
sitio_traza	Varchar	Sitio que realiza la traza.
hospital_traza	Varchar	Hospital que realiza la traza
accion	Varchar	actualización, inserción, eliminación o consulta
resultado_accion	Varchar	Resultado que se obtiene
variable_bajo_accion	Varchar	Variable que se encuentra en el momento de la acción.
valor_anterior	Varchar	Valor anterior
valor_nuevo	Varchar	Valor nuevo
descripción_accion	Varchar	Descripción de la acción.

Nombre: var_entero		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_valor_var	Serial	Valor entero auto incremento, llave primaria
valor_entero	Integer	Valor entero que toma la variable en un submodelo X.
valor_validado	Boolean	false (posee inconsistencia) y trae (no posee inconsistencia).
estado_valor	Varchar(n)	incluido, monitoreado, limpio
dia_incluido_entero	Integer	Día de incluido el valor entero.
mes_incluido_entero	Integer	Mes de incluido el valor entero.
anno_incluido_entero	Integer	Año de incluido el valor entero.
hora_incluido_entero	Time	Hora de incluido el valor entero.
id_variable	Integer	Llave primaria foránea que la toma de la entidad variable

id_modelo	Integer	Llave foránea que la toma de la entidad modelo
id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo
id_sitio	Integer	Llave foránea que la toma de la entidad sitio
usuario_login	Varchar(n)	Llave foránea que la toma de la entidad usuario.
id_paciente	Integer	Llave foránea que la toma de la entidad paciente.

Nombre: var_decimal		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_valor_var	Serial	Valor entero auto incremento, llave primaria
valor_decimal	Real	Valor decimal que toma la variable en un submodelo X
valor_validado	Boolean	false (posee inconsistencia) y trae (no posee inconsistencia).
estado_valor	Varchar(n)	incluido, monitoreado, limpio
dia_incluido_decimal	Integer	Día de incluido el valor decimal.
mes_incluido_decimal	Integer	Mes de incluido el valor decimal.
anno_incluido_decimal	Integer	Año de incluido el valor decimal.
hora_incluido_decimal	Time	Hora de incluido el valor decimal.
id_variable	Integer	Llave primaria foránea que la toma de la entidad variable
id_modelo	Integer	Llave foránea que la toma de la entidad modelo
id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo
id_sitio	Integer	Llave foránea que la toma de la entidad sitio
usuario_login	Varchar(n)	Llave foránea que la toma de la entidad usuario.
id_paciente	Integer	Llave foránea que la toma de la entidad paciente.

Nombre: var_texto		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_valor_var	Serial	Valor entero auto incremento, llave primaria

valor_texto	Varchar	Valor texto que toma la variable en un submodelo X.
valor_validado	Boolean	false (posee inconsistencia) y trae (no posee inconsistencia).
estado_valor	Varchar(n)	incluido, monitoreado, limpio
dia_incluido_texto	Integer	Día de incluido el valor texto.
mes_incluido_texto	Integer	Mes de incluido el valor texto.
anno_incluido_texto	Integer	Año de incluido el valor texto.
hora_incluido_texto	Time	Hora de incluido el valor texto.
id_variable	Integer	Llave primaria foránea que la toma de la entidad variable
id_modelo	Integer	Llave foránea que la toma de la entidad modelo
id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo
id_sitio	Integer	Llave foránea que la toma de la entidad sitio
usuario_login	Varchar(n)	Llave foránea que la toma de la entidad usuario.
id_paciente	Integer	Llave foránea que la toma de la entidad paciente.

Nombre: var_fecha		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_valor_var	Serial	Valor entero auto incremento, llave primaria
dia_var_fecha	Integer	Día de la variable fecha
mes_var_fecha	Integer	Mes de la variable fecha
anno_variable_fecha	Integer	Año de la variable fecha
valor_validado	Boolean	false (posee inconsistencia) y trae (no posee inconsistencia).
estado_valor	Varchar(n)	incluido, monitoreado, limpio
dia_incluido_fecha	Integer	Día de incluido el valor fecha.
mes_incluido_fecha	Integer	Mes de incluido el valor fecha.
anno_incluido_fecha	Integer	Año de incluido el valor fecha.
hora_incluido_fecha	Time	Hora de incluido el valor fecha.
id_variable	Integer	Llave primaria foránea que la toma de la entidad variable
id_modelo	Integer	Llave foránea que la toma de la entidad modelo
id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo
id_sitio	Integer	Llave foránea que la toma de la entidad sitio

usuario_login	Varchar(n)	Llave foránea que la toma de la entidad usuario.
id_paciente	Integer	Llave foránea que la toma de la entidad paciente.

Nombre: var_nomenclador		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_valor_var	Serial	Valor entero auto incremento, llave primaria
valor_nomenclador	Varchar	Valor nomenclador que toma la variable en un submodelo X.
valor_validado	Boolean	false (posee inconsistencia) y trae (no posee inconsistencia).
estado_valor	Varchar(n)	incluido, monitoreado, limpio
dia_incluido_valornom	Integer	Día de incluido el valor nomenclador.
mes_incluido_ valornom	Integer	Mes de incluido el valor nomenclador.
anno_incluido_ valornom	Integer	Año de incluido el valor nomenclador.
hora_incluido_ valornom	Time	Hora de incluido el valor nomenclador.
id_variable	Integer	Llave primaria foránea que la toma de la entidad variable
id_modelo	Integer	Llave foránea que la toma de la entidad modelo
id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo
id_sitio	Integer	Llave foránea que la toma de la entidad sitio
usuario_login	Varchar(n)	Llave foránea que la toma de la entidad usuario.
id_paciente	Integer	Llave foránea que la toma de la entidad paciente.

Nombre: var_caracter		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_valor_var	Serial	Valor entero auto incremento, llave primaria
valor_caracter	Varchar	Valor caracter que toma la variable en un submodelo X
valor_validado	Boolean	false (posee inconsistencia) y trae (no posee inconsistencia).
estado_valor	Varchar(n)	Incluido, monitoreado, limpio

dia_incluido_caracter	Integer	Día de incluido el valor del caracter.
mes_incluido_caracter	Integer	Mes de incluido el valor del caracter.
anno_incluido_caracter	Integer	Año de incluido el valor del caracter.
hora_incluido_caracter	Time	Hora de incluido el valor del caracter.
id_variable	Integer	Llave primaria foránea que la toma de la entidad variable
id_modelo	Integer	Llave foránea que la toma de la entidad modelo
id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo
id_sitio	Integer	Llave foránea que la toma de la entidad sitio
usuario_login	Varchar(n)	Llave foránea que la toma de la entidad usuario.
id_paciente	Integer	Llave foránea que la toma de la entidad paciente.

Nombre: cronograma		
Descripción: Muestra el cronograma de ejecución de un ensayo.		
Atributo	**Tipo**	**Descripción**
id_cronograma	Serial	Valor entero auto incremento, llave primaria
tiempo_duracion_cronograma	integer	Tiempo destinado a durar el cronograma.
descripción_cronograma	Varchar	Descripción del cronograma.
id_ensayo	Integer	Llave foránea que la toma de la entidad ensayo

Nombre: etapa		
Descripción: Muestra las etapas definidas para el cronograma		
Atributo	**Tipo**	**Descripción**
id_etapa	Serial	Valor entero auto incremento, llave primaria
nombre_etapa	Varchar	Nombre de la etapa, debe ser único
inicio_etapa	Integer	Día de inicio de rango de etapa
fin_etapa	Integer	Día final de rango de etapa.
cant_visitas	Integer	Cantidad de visitas asociadas a la etapa.
descripción_etapa	Varchar	Descripción de la etapa.
id_cronograma	Integer	Llave foránea que la toma de la entidad cronograma.

Nombre: visita		
Descripción: Muestra las visitas asignadas a cada cronograma y etapas en un ensayo.		
Atributo	**Tipo**	**Descripción**
id_visita	Serial	Valor entero auto incremento, llave primaria
dia_visita_ensayo	Integer	Día de la visita en el ensayo
hora_visita	Time	Hora de la visita
tipo_visita	Varchar	D(día), H(hora), P(período)
intervalo_visita	Integer	Muestra el intervalo de la visita.
Duración_visita	Integer	Tiempo de duración de la visita.
rango_llenado	Integer	Define el tiempo que tienen para llenar el modelo.
id_cronograma	Integer	Llave foránea que la toma de la entidad cronograma
id_etapa	Integer	Llave foránea que la toma de la entidad etapa.

Nombre: inconsistencia		
Descripción: Muestra las inconsistencias que existen.		
Atributo	**Tipo**	**Descripción**
id_inconsistencia	Serial	Valor entero auto incremento, llave primaria
dia_inconsistencia	Integer	Día en que se vio la inconsistencia
mes_inconsistencia	Integer	Mes en que se vio la inconsistencia.
anno_inconsistencia	Integer	Año en que se vio la inconsistencia.
hora_inconsistencia	Time	Hora en que se vio la inconsistencia.
tipo_inconsistencia	Varchar	Caracter, texto, nomenclador, fecha, decimal, entero.
descripción_inconsistencia	Varchar	Descripción de la inconsistencia

Nombre: inconsistencia_entero		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_inconsistencia	Integer	Llave primaria foránea que la toma de la entidad inconsistencia.
Id_valor_var	Integer	Llave foránea que la toma de la entidad var_entero.
Id_variable	Integer	Llave foránea que la toma de la entidad var_entero.

Nombre: inconsistencia_texto		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_inconsistencia	Integer	Llave primaria foránea que la toma de la entidad inconsistencia.
Id_valor_var	Integer	Llave foránea que la toma de la entidad var_texto
id_variable	Integer	Llave foránea que la toma de la entidad var_texto.

Nombre: inconsistencia_decimal		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_inconsistencia	Integer	Llave primaria foránea que la toma de la entidad inconsistencia.
Id_valor_var	Integer	Llave foránea que la toma de la entidad var_decimal.
Id_variable	Integer	Llave foránea que la toma de la entidad var_decimal.

Nombre: inconsistencia_fecha		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_inconsistencia	Integer	Llave primaria foránea que la toma de la entidad inconsistencia.
Id_valor_var	Integer	Llave foránea que la toma de la entidad var_fecha.
Id_variable	Integer	Llave foránea que la toma de la entidad var_fecha.

Nombre: inconsistencia_caracter		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_inconsistencia	Integer	Llave primaria foránea que la toma de la entidad inconsistencia.

Id_valor_var	Integer	Llave foránea que la toma de la entidad var_caracter.
Id_variable	integer	Llave foránea que la toma de la entidad var_caracter.

Nombre: inconsistencia_nomenclador		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_inconsistencia	Integer	Llave primaria foránea que la toma de la entidad inconsistencia.
Id_valor_var	Integer	Llave foránea que la toma de la entidad var_nomenclador
id_variable	Integer	Llave foránea que la toma de la entidad var_nomenclador

Nombre: querie		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_querie	Serial	Valor entero auto incremento, llave primaria
dia_querie	Integer	Día de la querie.
mes_querie	Integer	Mes de la querie.
anno_querie	Integer	Año de la querie.
tipo_query	Varchar	variable, libre o valor
autor_querie	Varchar	Autor de la querie.
estado_querie	Varchar	confirmado, asignado, corregido
historial_querie	Varchar	Historial de la querie.
descripción_querie	Varchar	Descripción de la querie.
id_ensayo	Integer	Llave foránea que la toma de la entidad ensayo

Nombre: query_variable		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_querie	Integer	Llave primaria foránea que la toma de la entidad querie.

id_variable	Integer	Llave foránea que la toma de la entidad var_entero.

Nombre: query_inconsistencia_valor		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_querie	Integer	Llave primaria foránea que la toma de la entidad querie.
Id_inconsistencia	Integer	Llave foránea que la toma de la entidad inconsistencia.

Nombre: derivar_variable		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_validacion_variable	Serial	Valor entero auto incremento, llave primaria
estado_validacion	Varchar(n)	El estado puede ser: valores, concatenar, formula, asignada, repetir.
Id_visita	Integer	Llave foránea que la toma de la entidad visita.
Id_variable	Integer	Llave foránea que la toma de la entidad variable.
Id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo
id_ensayo	Integer	Llave foránea que la toma de la entidad ensayo.
Id_modelo	Integer	Llave foránea que la toma de la entidad modelo.

Nombre: validación_valores		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_valores	Serial	Valor entero auto incremento, llave primaria
id_validacion_variable	Integer	Llave primaria foránea que la toma de la entidad derivar_variable.
Valores	Varchar	Valores a validar.

Nombre: validación_por_formula		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_elemento_formula	Serial	Valor entero auto incremento, llave primaria
posic_elemento	Integer	Ubicación del elemento en la fórmula.
token_elemento	Varchar	variab, operador (logico, aritmético, comparación), operación (ln, concatenar, raiz cuadrada), paréntesis y número.
id_validacion_variable	Integer	Llave primaria foránea que la toma de la entidad derivar_variable.

Nombre: valor_elemento_formula		
Descripción:		
Atributo	**Tipo**	**Descripción**
elemento_formula	Varchar	
id_elemento_formula	Integer	Llave primaria foránea que la toma d ela entidad validación_por_formula
id_validación_variable	Integer	Llave primaria foránea que la toma de la entidad validación_por_formula.

Nombre: variable_formula		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_elemento_formula	Integer	Llave primaria foránea que la toma d ela entidad validación_por_formula
id_validación_variable	Integer	Llave primaria foránea que la toma de la entidad validación_por_formula.
id_variable	Integer	Llave primaria foránea que la toma de la entidad variable
id_visita	Integer	Llave foránea que la toma de la entidad visita
id_modelo	Integer	Llave foránea que la toma de la entidad modelo.
id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo.

Nombre: validación_concatenar
Descripción:

Atributo	**Tipo**	**Descripción**
Id_concatenar	Serial	Valor entero auto incremento, llave primaria
id_validacion_variable	Integer	Llave primaria foránea que la toma de la entidad derivar_variable.
pos_variable	Integer	Lugar que ocupa en la cadena de valores a generar.
id_variable	Integer	Llave foranea que la toma de la entidad variable
id_visita	Integer	Llave foránea que la toma de la entidad visita
id_modelo	Integer	Llave foránea que la toma de la entidad modelo
id_submodelo	Integer	Llave foránea que la toma de la entidad submodelo.

Nombre: validar_numerico		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_validacion_variable	Integer	Llave primaria foránea que la toma de la entidad derivar_variable.
obligatorio	Boolean	true (valor obligatorio) , false (valor opcional).
cant_cifras_enteras	Integer	Cifras antes de la coma
cant_cifras_decimal	Integer	Cifras después de la coma.
unidad_medida	Varchar(n)	Unidad de medida (cm, kg, litro, entre otros).

Nombre: validar_texto		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_validacion_variable	Integer	Llave primaria foránea que la toma de la entidad derivar_variable.
obligatorio	Boolean	true (valor obligatorio) , false (valor opcional).
cant_caracteres	Integer	Número de caracteres máximo de la cadena.

Nombre: validar_nomenclador		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_validacion_variable	Integer	Llave primaria foránea que la toma de la entidad derivar_variable.

inicio_rango	Varchar	Valor de inicio del rango del nomenclador.
fin_rango	Varchar	Valor final del rango nomenclador
obligatorio	Boolean	true (valor obligatorio) , false (valor opcional).

Nombre: validar_fecha		
Descripción:		
Atributo	**Tipo**	**Descripción**
id_validacion_variable	Integer	Llave primaria foránea que la toma de la entidad derivar_variable.
dia_inicio_rango_fecha	Integer	Inicio del rango permisible para la fecha
mes_inicio_rango_fecha	Integer	
anno_inicio_rango_fecha	Integer	
dia_fin_rango_fecha	Integer	Fin del rango permisible para la fecha.
mes_fin_rango_fecha	Integer	
anno_fin_rango_fecha	Integer	
dia_obligatorio	Boolean	true (valor obligatorio) , false (valor opcional).
mes_obligatorio	Boolean	true (valor obligatorio) , false (valor opcional).
anno_obligatorio	Boolean	true (valor obligatorio) , false (valor opcional).

Se hace una pequeña descripción de las tablas del diagrama Entidad-Relación que se generan a partir de relaciones, a las cuales se le agregan atributos necesarios para el trabajo. (Ver Anexo 3)

2.6 Análisis de Optimización de Consultas (Querys).

Las consultas son peticiones que se realizan a la BD, para almacenar información y/o visualizar o actualizar la información existente en la misma. Una consulta puede expresarse de diferentes maneras, donde cada una sugiere una estrategia para encontrar la respuesta y por lo tanto algunas pueden ser más óptimas que otras.

Después de enviar la consulta, la BD debe producir su correspondiente plan de ejecución. El primer paso en este proceso es traducir la consulta desde SQL a un árbol lógico en álgebra relacional, proceso comúnmente llamado **parser**. El próximo paso es traducir el árbol de la consulta en el plan de ejecución. Generalmente existe un gran número de planes que implementan al árbol de la consulta; el proceso de encontrar el mejor de estos planes se le denomina **optimización de consultas**.

Finalmente el optimizador envía el plan óptimo al motor de ejecución. El motor de ejecución ejecuta el plan usando como entrada las relaciones almacenadas en las BD y produce una tabla con los datos solicitados como salida. (Ver figura 2.5) [28]

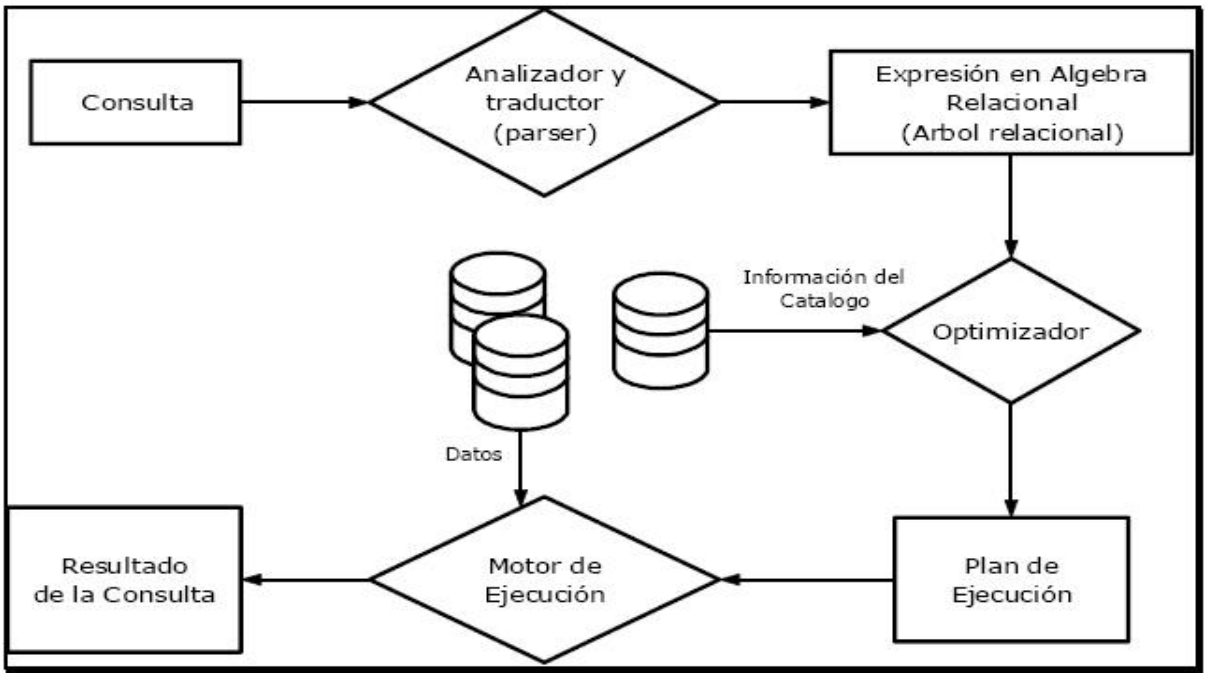

Figura 2.5 Pasos en el procesamiento de una consulta SQL

Para la selección del plan óptimo de implementación durante la optimización de las consultas, el procesador de consultas elige:

- Los índices que se van a utilizar, en su caso.
- El orden de ejecución de las combinaciones.
- El orden de aplicación de las restricciones, como las cláusulas **WHERE**.

Por lo cual deben tenerse en cuenta algunos criterios, que pueden ser útiles, por ejemplo:

1. Es preferible no usar la cláusula **HAVING** si la condición deseada se puede expresar en la cláusula **WHERE**.
2. Se crean índices en tantos campos como sea posible.

3. Es importante conocer las particularidades del manejador, esto se refiere a conocer como han sido implementadas las rutinas de procesamiento de las consultas. Por ejemplo, es importante saber, al existir una condición disyuntiva (con **OR**) en la consulta si se usan o no los índices existentes.
4. Las condiciones de **JOIN** se pueden evaluar más eficientemente contra un índice primario. Y en general, en términos de rendimiento es preferible evaluar una condición de igualdad numérica que una condición de igualdad sobre cadenas (strings) de caracteres.
5. 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.

En ocasiones no se hace el uso más adecuado de las potencialidades que brinda el lenguaje SQL apareciendo así consultas que pudieran ser optimizadas si se implementarán de otro modo, el orden en que se hacen los **INNER JOIN**, aunque no se vea así, puede afectar el tiempo de ejecución de la consulta y con ella el rendimiento de la aplicación al ejecutarla. Por ejemplo, para la BD del SIMDEC, se realizará una consulta para obtener todas las variables de tipo nomenclador que se usen en algún ensayo creado entre los años 2000 y 2007 que se encuentre en estado activo.

1. Consulta versión 1 para el resultado esperado:

```
– SELECT
        public."variable".nombre_variable
  FROM
      public.submodelo_variable  INNER JOIN public."variable"
      ON (public.submodelo_variable.id_variable = public."variable".id_variable)
      INNER JOIN public.modelo_submodelo
      ON (public.submodelo_variable.id_submodelo = public.modelo_submodelo.id_submodelo)
      INNER JOIN public.ensayo_modelo
      ON (public.modelo_submodelo.id_modelo = public.ensayo_modelo.id_modelo)
```

```
INNER JOIN public.ensayo ON (public.ensayo_modelo.id_ensayo = public.ensayo.id_ensayo)
INNER JOIN public.tipo_variable
ON (public."variable".id_tipo_var = public.tipo_variable.id_tipo_var)
WHERE
  (public.ensayo.anno_crea_ensayo >= 2000) AND
  (public.ensayo.anno_crea_ensayo <= 2007) AND
  (public.tipo_variable.tipo_variab = 'nomenclador') AND
  (public.ensayo.estado_ensayo = 'activo')
GROUP BY nombre_variable
```

2. Consulta versión 2 para el resultado esperado:

```
– SELECT
    public."variable".nombre_variable
  FROM
    public.submodelo_variable INNER JOIN public."variable"
    ON (public.submodelo_variable.id_variable = public."variable".id_variable)
    INNER JOIN public.modelo_submodelo
    ON (public.submodelo_variable.id_submodelo = public.modelo_submodelo.id_submodelo)
    INNER JOIN public.ensayo_modelo
    ON (public.modelo_submodelo.id_modelo = public.ensayo_modelo.id_modelo)
    INNER JOIN public.ensayo ON (public.ensayo_modelo.id_ensayo = public.ensayo.id_ensayo)
    INNER JOIN public.tipo_variable
    ON (public."variable".id_tipo_var = public.tipo_variable.id_tipo_var)
```

```
WHERE
        (public.ensayo.anno_crea_ensayo BETWEEN 2000 AND 2007) AND
        (public.tipo_variable.tipo_variab = 'nomenclador') AND
        (public.ensayo.estado_ensayo = 'activo')
GROUP BY nombre_variable
```

Para ambos caso, el resultado fue el mismo de 400 variables, pero la primera consulta se ejecutó en 0,33 segundos, mientras que la segunda lo hizo en 0,29, aunque a primera vista la diferencia se marca en la utilización de operadores de comparación <= (menor o igual que), >= (mayor o igual que) o **BETWEEN**, que no es mucha, cuando analizamos el volumen de información sobre el cual se puede ejecutar una de las consultas antes mencionadas y el número de repeticiones, se necesita por cada vez ejecutada la segunda, 0,04 segundos más para ejecutar la primera, o lo que es lo mismo, 4 segundos por cada 100 veces que se ejecuten.

Conclusiones

En este capítulo se ha mostrado el diagrama de clases persistentes, analizando todas las clases persistentes encontradas, definiendo un total de 45 clases, de las cuales se ha explicado brevemente cada uno de sus atributos y su tipo.

Además se ha mostrado el diagrama entidad relación para la BD, explicando brevemente cada una de las entidades representadas y sus atributos, en el cual se definieron 56 entidades, lo que facilita el entendimiento del diagrama y de cada una de sus partes, mostrando la descripción de las entidades con sus atributos, el tipo y una pequeña descripción de los más importantes. Se mostraron, además, varias consultas, analizando cual resulta más óptima a la hora de su ejecución.

Capítulo 3

Validación del Diseño Realizado

En este capítulo se brinda información sobre la validación realizada al diseño descrito en el capítulo 2, mostrando en detalle las reglas a tener en cuenta para un diseño de óptima calidad, así como los detalles de una validación funcional a través de la selección de consultas para un llenado voluminoso e inteligente de la BD.

3.1 Validación teórica del diseño

Para realizar un buen diseño de BD hay que tener en cuenta varios aspectos que son importantes velar como: la integridad de los datos, la normalización del diseño, la redundancia de información, la trazabilidad de las acciones que se realizan en la BD y sobre todo la seguridad, la cual es base principal de los aspectos anteriormente mencionados, permitiendo controlar los accesos a los datos y que los mismos no se corrompan como resultado de acciones no controladas.

3.1.1 Integridad

La integridad de los datos se refiere a la corrección y completitud de los datos en una BD. Al modificar el contenido de la BD con sentencias **INSERT**, **DELETE** o **UPDATE**, la integridad de los datos puede verse afectada añadiendo datos no válidos, modificando datos existentes tomando un valor incorrecto o eliminando datos que al hacerlo violan alguna regla. La integridad de datos presenta varias restricciones que posibilitan la declaración y comprobación de condiciones para expresar la consistencia, corrección y exactitud de datos almacenados, estas son: Integridad de dominio, de entidad, referencial, entre otras.

Integridad de dominio: Se restringen los valores que puede tomar un atributo respecto a su dominio, en el caso de la BD para el manejo de datos de los ensayos clínicos se definió la fecha por los campos: día, mes y año, en el que los días tienen que ser >0 y <32, restringiendo su dominio a un valor entre 1 y 31, en el caso del mes sucede algo parecido el cual puede tomar valores entre 1 y 12, y en el caso del año que su dominio estaría en los enteros de 4 dígitos entre 1000 y 9999, rango que aunque es amplio, valida cualquier fecha permisible de la realidad. Otro ejemplo de integridad de dominio existe en si el valor de una cadena de caracteres es de no más de 15 caracteres, se usa un varchar(n)(15), ejemplo de

ello sería el carné de identidad de un usuario que es un tipo de dato varchar y solo acepta cadenas de longitud 11, su dominio sería varchar(n)(11). Cuando el tipo de dato es varchar, acepta números, letras o caracteres, pero existen campos que es necesario validar los datos que pueden almacenar, por ejemplo en el caso anterior con el número del carné de identidad de un usuario que solo admite números o el número de teléfono, ya sea en la tabla *sitio*, *hospital* o en otras donde este presente, sólo acepta números y el caracter (-). Para realizar estas validaciones PostgreSQL presenta los "obligadores" (constrains), que se encargan de chequear (**CHECK**) el cumplimiento de una o varias condiciones que deben cumplir el o los campos de una tabla y disparan excepciones (errores), en caso de ser violadas. También existen los "disparadores" (triggers) , los cuales, pueden validar campos de más de una tabla a la vez; como su nombre lo indica disparan una excepción (error), en caso de no cumplirse alguna(s) de las condiciones que valida. Estos hacen uso de funciones especiales, las cuales validan la(s) condiciones que debe cumplir una tupla para ser aceptada como correcta e insertarla en la BD o retornan valor nulo (**NULL**), en caso de violarse alguna condición, además pueden ser programadas para dispararse antes o después de ocurrir eventos de **INSERT**, **UPDATE**, o **DELETE** de una tulpa en la tabla. Un ejemplo implementado de los "obligadores" , en este caso un **CHECK**, para que el número del carné de identidad sólo acepte números haciendo uso de expresiones regulares, sería:

– no_ci_usuario **VARCHAR**(11) **CHECK**(no_ci_usuario SIMILAR **TO** '%(1|2|3|4|5|6|7|8|9|0)')

En el caso de los "disparadores" (triggers) un ejemplo sería para darle cumplimiento a algún **RF** de eliminación de datos de alguna tabla, siendo necesario validar la integridad referencial, ya que como buenas prácticas de diseño se hace mención a que en la mayoría de las ocasiones es bueno no dejarle todas las validaciones de integridad referencial al SGBD, porque el mismo podría reaccionar ante estos de una forma no esperada obtener un funcionamiento incorrecto o incluso que se cuelgue, por ello, se crean "disparadores" antes del evento **DELETE** en las tablas, validando que la tupla de la tabla que se elimina no viola la integridad referencial propia de la tabla. Un ejemplo de situación donde es necesaria la implementación de un "disparador" (trigger) que valide esto, sería, al eliminar un sitio hay que chequear que no se encuentre realizando en él ningún EC en estado "activo", es decir, que este en uso.

Otro tipo de integridad de dominio es definir el conjunto de valores lo más específico que se pueda, ejemplo de ello es el estado de un *modelo_datos* (modelo lleno), que puede ser: incompleto, completo,

monitoreado o firmado. Esto trae consigo que el usuario de acuerdo al modelo que está analizando no cometa un error que pueda afectar el sistema y viole su integridad.

Integridad de entidad: la clave primaria de una entidad no puede tener valores nulos y siempre deberá ser única. Un ejemplo de las buenas prácticas lo constituye la elección del campo de la tabla que será o formará parte de la llave primaria, partiendo de la obligación de que todos los valores del atributo son únicos y nunca nulos (**NULL**).

Integridad referencial: las claves ajenas de una tabla hija se tienen que corresponder con la clave primaria de la tabla padre con la que se relaciona. Por ejemplo, en la tabla *sitio_hospital* se necesita el id (llave primaria) del sitio para poder insertarlo, ya que id_sitio es una llave primaria foránea de la tabla *sitio_hospital*, por lo cual es necesario validar la existencia del sitio y del hospital, antes de insertar o actualizar alguna tupla en la tabla *sitio_hospital*. Otro ejemplo, en la tabla *etapa* existe como llave foránea el id_cronograma al que pertenece, por tanto al insertar una etapa debe existir previamente el cronograma al que se inserta la misma. Aunque PostgreSQL valida que se cumpla la integridad referencial, en la BD se hace uso de “disparadores” (triggers) para el tratamiento de errores, que permiten validar la existencia previa de las llaves foráneas e insertar información válida en la BD.

Datos requeridos: Establece que en una tabla al insertar una tupla no exista una columna determinada con valor nulo. Se define efectuando la declaración de una columna es **NOT NULL** cuando la tabla que contiene la columna se crea por primera vez. Ejemplo de ello es que en la tabla *ensayo* el nombre del ensayo es un dato requerido, por lo que se declara **NOT NULL**, validando que no existan tuplas con esa columna con valor nulo (**NULL**).

Chequeo de validez: Cuando se crea una tabla cada columna tiene un tipo de datos y el DBMS asegura que solamente los datos del tipo especificado sean ingresados en la tabla. Además el diseñador puede hacer chequeo de validez, por ejemplo, el caso del número del carné de identidad, mencionado anteriormente.

3.1.2 Normalización de la BD.

La normalización de la BD es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles

de mantener. Son una serie de reglas que sirven para ayudar a los diseñadores de BD a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede.

Además ayuda a prevenir errores lógicos en la manipulación de datos, facilitando también agregar nuevas columnas sin romper el esquema actual ni las relaciones.

Los propósitos de la normalización son:

- Reducir o eliminar el almacenaje de datos duplicados
- Organizar los datos en una estructura eficiente y lógica

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

Existen varios niveles de normalización: [29]

- Primera Forma Normal (1NF)
- Segunda Forma Normal (2NF)
- Tercera Forma Normal (3NF)
- Forma Normal Boyce-Codd
- Cuarta Forma Normal
- Quinta Forma Normal o Forma Normal de Proyección-Unión
- Forma Normal de Proyección-Unión Fuerte
- Forma Normal de Proyección-Unión Extra Fuerte.
- Forma Normal de Clave de Dominio.

Cada nuevo nivel o forma nos acerca más a hacer una BD verdaderamente relacional. Cada una tiene sus propias reglas. Cuando una BD se conforma a un nivel, se considera normalizada a esa forma de normalización. No siempre es una buena idea tener una BD conformada en el nivel más alto de normalización, puede llevar a un nivel de complejidad que pudiera ser evitado si estuviera en un nivel más bajo de normalización.

Primera Forma Normal

Incluye la eliminación de todos los grupos repetidos, es decir, establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas. Es decir, prohíbe los atributos multivaluados, los atributos compuestos y sus combinaciones. Esto indica que todos los atributos deben tener valores atómicos.

Poner la BD en la Primera Forma Normal resuelve el problema de los encabezados de columna múltiples. Muy a menudo, los diseñadores de BD inexpertos harán algo similar a la tabla no normalizada. Una y otra vez, crearán columnas que representen los mismos datos. La normalización ayuda a clarificar la BD y a organizarla en partes más pequeñas y más fáciles de entender.

Segunda Forma Normal

Asegura que todas las columnas que no son llave sean completamente dependientes de la llave primaria (PK), es decir, establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un término que describe a aquellos datos que no dependen de la llave primaria de la tabla para identificarlos.

Una vez alcanzado el nivel de la Segunda Forma Normal, se controlan la mayoría de los problemas de lógica. Podemos insertar un registro sin un exceso de datos en la mayoría de las tablas.

Tercera Forma Normal

Elimina cualquier dependencia transitiva. Una dependencia transitiva es aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave. Una tabla está normalizada en esta forma si todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas.

Cuando las tablas están en la Tercera Forma Normal se previenen errores de lógica cuando se insertan o eliminan registros. Cada columna en una tabla está identificada de manera única por la llave primaria, y no debe haber datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir.

La BD para el SIMDEC está normalizada hasta la Tercera Forma Normal (3NF) ya que provee suficiente nivel de normalización, es decir, inicialmente se parte del cumplimiento de la Primera Forma Normal (1NF), ya que todos los valores de los atributos de cada tabla son atómicos y no presenta valores

multivaluados o atributos compuestos, por ejemplo, en la tabla *ensayo* existe una fecha de creado para el diseño y fecha de terminado las cuales se recogen de forma atómica, como: día, mes y año de creado y lo mismo para la fecha en que se termina el diseño, es decir, fecha:creado: {dia_creado, mes_creado, anno_creado} y fecha:terminado: {dia_terminado, mes_terminado, anno_terminado}.

Otro ejemplo en la tabla *paciente* tenemos el atributo número de inclusión del paciente (no_inclusion_paciente), y una fecha de inclusión, atributos que dependen de la llave del paciente, del ensayo al que se haga mención y del sitio en que se este realizando el mismo, por lo cual son atributos de la tabla *ensayo_paciente* que se generaría al relacionar el sitio con el paciente de muchos a muchos, ya que en la tabla *sitio* la llave de ensayo es llave foránea permitiendo ubicarse en el ensayo; en el caso de la fecha de inclusión, sucedería como el ejemplo anterior de las fechas de creado y terminado del ensayo.

Además cumple con la Segunda Forma Normal (2NF), ya que todas sus tablas están en 1FN y sus atributos no llave dependen únicamente de la llave de la tabla, permitiendo que la ocurrencia de cada fila en la tabla sea única y quede representada por la llave.

Por último la BD cumple con la Tercera Forma Normal (3NF), pues se encuentra en 2NF y todos sus atributos no llave, son independientes entre sí, eliminado cualquier dependencia transitiva.

Se ha normalizado hasta la 3NF, debido a que utilizar un nivel de normalización muy elevado conduce a una BD más relacional, pero más compleja, es decir, normalizar demasiado puede conducir a tener una BD ineficiente y obtener un esquema demasiado complejo para trabajar.

3.1.3 Análisis de redundancia de información

La redundancia de datos es aquella información duplicada que genera inconsistencia en la BD, requiriendo más espacio en disco; aunque en proyectos grandes es imposible evitarla al 100%, lo que a veces es deseable por cuestiones de rendimiento. Tener presente que antes con el almacenamiento de los EC en gestores de BD no profesionales se realizaba una entrada doble de los datos, lo que traía consigo gran redundancia de información. Con la realización de la BD para el SIMDEC gran parte de esta redundancia de datos se verá solucionada, principalmente los problemas de entrada doble de datos que serán sustituidos por las validaciones propias de las interfaces de cada diseño y en un segundo momento por las validaciones de las variables en cada uno de los modelos y submodelos en la BD.

Teniendo como punto de partida el diseño de la BD se debe mencionar la existencia de tablas donde se recogen los valores de las variables, que además deben recoger el modelo, submodelo, paciente, sitio y usuario a los que están asociados los valores de las variables, dado que aunque crean información redundante son necesarios para tener un control de los valores introducidos para cada una de las variables.

Además se tiene la tabla *variable_formula*, la cual almacena las variables que se utilizan en las fórmulas para validar otras variables, recogiendo además la visita, modelo y submodelo, que aunque están presentes en la tabla *derivar_variable*, no son los mismos valores, además es necesario recogerlos porque los valores utilizados de estas variables para validar otras depende de los modelos, submodelos y visitas en los que se hayan recogido anteriormente.

Otro ejemplo lo constituye la tabla *sitio* en la que se almacena la información de los lugares donde se realizan los ensayos clínicos, ya sea, dentro de los hospitales o fuera de ellos; aunque los datos recogidos por los sitios en muchas ocasiones se asemejan a la de los hospitales, esta varia en al alguno de los valores recogidos por lo cual se hace necesario almacenarla en una nueva tabla con el nombre *sitio*, la cual sirve además, como se hizo mención anteriormente, para registrar aquellos sitios que no estén ubicados dentro de ningún hospital

3.1.4 Análisis de seguridad de la BD.

La información que almacena una BD, está en constante riesgo de sufrir ataques que puedan provocar su modificación o pérdida, por ello es de vital importancia velar la seguridad de la misma, protegiéndola, en un primer momento, contra accesos no autorizados o cualquier acción que puedan violar la integridad de los datos o la confidencialidad de los mismos. En un segundo momento es necesario proteger los datos que se almacenan en la BD, para lo cual se deben realizar salvas.

Aunque evitar el acceso mal intencionado a la BD se hace difícil, se pueden tomar medidas que permitan mantener la consistencia de la información y que sea tan alto el costo de violar la seguridad que frustre casi cualquier intento de acceso no autorizado a la BD.

“Los sistemas de bases de datos ofrecen múltiples características que permiten securizarlas”.[30]

Como punto de partida se deben presentar los datos solo a quien esté autorizado. PostgreSQL, permite realizar configuraciones para controlar los permisos de los usuarios, utilizando tres niveles de acceso. En

un primer nivel (nivel 0) se configuran los permisos de conexión para los host y los usuarios a la o las BDs, datos que se recogen en el archivo pg_hba.conf [31], en el mismo se define que PCs (dirección o direcciones IP) tendrán acceso, además a cuál o cuales BD y el modo en que podrán conectarse, que puede ser: conexión sin contraseña, validando el usuario y la contraseña para conectarse, o que rechace cualquier conexión desde el IP o rangos IP y usuarios seleccionados. En un segundo nivel (nivel 1), PostgreSQL permite configurar a que BD pueden acceder determinados usuarios, utilizando las opciones del archivo pg_ident.conf [31]. Por último existe un tercer nivel (nivel 2), que permite configurar dentro de las BD los accesos a las tablas, utilizando los comandos GRANT y REVOKE para permitir o denegar los permisos, respectivamente.

En la BD propuesta se deben establecer controles de seguridad para todos los datos almacenados, garantizando que solo los usuarios autorizados puedan efectuar operaciones correctas sobre algunas tablas o sobre toda la BD.

Dentro de la BD el registro de los usuarios sigue un orden que permite controlar los accesos de los mismos. Inicialmente la inserción de cada usuario lo realiza un administrador, el cual cuenta con los permisos necesarios para realizar operaciones de administración en la BD como, inserción de ensayos, modelos, submodelos y variables, permitiendo esto controlar las acciones que se realizan en la BD. Una vez registrado un usuario, luego de introducir sus datos, la contraseña introducida se encripta antes de ser almacenada, de modo que solo el usuario tiene conocimiento de la misma, además se genera una firma para cada usuario que lo identifica de forma única, permitiendo validar determinadas acciones que se realizan en el sistema, de las cuales se recoge constancia de quien las realiza y que solamente las puede realizar el usuario que desempeña un determinado rol, y que una vez finalizada debe introducir su firma como constancia. Como se mencionó anteriormente en la BD se almacenan los roles que se le asignan a los usuarios por direcciones IPs, validando así las máquinas que pueden conectarse a la BD y desde la PC, un usuario, con que rol lo puede hacer. Los usuarios registrados tienen los permisos restringidos dentro de la BD, es decir, acceden a las tablas según el rol que desempeñan.

Para validar que no exista pérdida de información en caso de corrupción de los datos, PostgreSQL, permite la realización de salvas (backups) mediante el volcado (dump) de la BD, los cuales pueden realizarse de forma automática o dejarlos a elección del cliente.

Para el volcado de la BD se utiliza el comando pg_dump el cual indica al SGBD que se va a realizar la salva de la BD, además cuenta con una serie de parámetros adicionales que permiten indicar el nombre de la BD, el usuario y contraseña para conectarse a la misma, el nombre que tomará el archivo de salva y donde se desea localizar la misma. El volcado de la BD puede realizarse en caliente, es decir, con el servicio postgres corriendo, lo cual constituye una ventaja, además se puede realizar a una o varias BD, además mantiene compatibilidad entre versiones, sus desventajas están marcadas por la lentitud que alcanza con BD de gran volumen y el consumo de recursos del sistema.

En caso de desear restaurar una salva para su análisis o uso, se utiliza el comando pg_restore indicándole la salva que deseamos restaurar, además de algún otro parámetro de los mencionados anteriormente.

3.1.5 Trazabilidad de las Acciones

La Trazabilidad es la cualidad que permite que todas las acciones realizadas sobre un sistema de tecnología de la información sean asociadas de modo inequívoco a un individuo o entidad. Además es la capacidad que tiene una organización o sistema para rastrear, reconstruir o establecer relaciones entre objetos monitoreados, para identificar y analizar situaciones específicas o generales en los mismos.[32]

Para llevar un control de las acciones realizadas por los usuarios que interactúan con la BD se creó una tabla *traza_auditoria*, en la cual se almacenará el historial de las actividades de los usuarios en todos los momentos, es decir, recogerá los datos necesarios para auditar el momento (fecha y hora) en que se llevó a cabo, la acción que se realiza, usuario que inició la acción, que pueden ser de consulta, inserción, actualización o eliminación, y el resultado de la misma.

3.2 Validación Funcional.

3.2.1 Revisión y selección de herramientas para pruebas de carga intensiva.

Para el llenado voluminoso de BD existen varias herramientas. En ocasiones se utiliza scripts encontrados en Internet para llenar sus BD y realizarles pruebas, aunque no son recomendables debido a que se debe tener conocimientos de programación para esto.

Para el llenado de la BD se utilizó la herramienta EMS Data Generador For PostgreSQL 2005, la cual permite la generación de datos para una o varias tablas a la vez, definiendo para cada campo el rango de

valores admisibles, dependiendo del tipo, además de la cantidad de tuplas que se desean generar, validando de forma automática la integridad referencial.

Se generaron para las pruebas a la BD un volumen de 10000 datos para las tablas *ensayo, modelo, submodelo y hospital*, los cuales se insertaron en un tiempo no mayor de 1.10 minutos. Para las tablas *cronograma, etapa, visita* y *variable* se generaron alrededor de 1000 datos y para las restantes alrededor de 500 datos invirtiendo en ello tiempos de 40 y 35 segundos respectivamente por cada tabla, debido a que estas tablas incluyen llaves foráneas y el software debe hacer una búsqueda en el catálogo de llaves para realizar las validaciones.

Se mostrarán algunas consultas hechas a la BD, utilizando la herramienta SQL Query 2005 for PostgreSQL, con el objetivo de probar el funcionamiento correcto de la BD, así como los tiempos de ejecución de la misma.

Las consultas generadas para la realización de pruebas, se escogen de acuerdo a las operaciones que se esperan sean las más utilizadas una vez implantada la BD, por ejemplo:

- Obtener los cronogramas de los ensayos que se encuentran activos:

```
SELECT
        public.etapa.nombre_etapa, public.visita.dia_visita_ensayo,
        public.modelo.nombre_modelo
FROM
        public.visita INNER JOIN public.cronograma
        ON (public.visita.id_cronograma = public.cronograma.id_cronograma) INNER JOIN public.etapa
        ON   (public.cronograma.id_cronograma = public.etapa.id_cronograma) AND
        public.visita.id_etapa = public.etapa.id_etapa) INNER JOIN public.visita_modelo
        ON (public.visita.id_visita = public.visita_modelo.id_visita) INNER JOIN public.modelo
        ON (public.visita_modelo.id_modelo = public.modelo.id_modelo)
        INNER JOIN public.ensayo
```

```
    ON (public.cronograma.id_ensayo = public.ensayo.id_ensayo)
WHERE
    (public.ensayo.estado_ensayo = 'activo')
ORDER BY
        public.etapa.nombre_etapa,
        public.visita.dia_visita_ensayo,
        public.modelo.nombre_modelo
```

El resultado de esta consulta fue de 550 filas, que representan los cronogramas de los ensayos activos y se realizó en un tiempo de 3,56 segundos.

Otro ejemplo puede ser, todos los usuarios y ensayos en los que juegan algún rol que se encuentren realizándose en sitios donde el rango de direcciones IP este entre 10.24.0.1 y 10.18.8.254:

```
SELECT
    public.ensayo.nombre_ensayo,
    public.usuario.nombre_usuario,
    public.usuario.apellidos_usuario
FROM
    public.usuario_ensayo INNER JOIN public.ensayo
  ON public.usuario_ensayo.id_ensayo = public.ensayo.id_ensayo) INNER JOIN public.usuario
  ON  (public.usuario_ensayo.usuario_login = public.usuario.usuario_login)
   INNER JOIN public.ensayo_rol ON (public.ensayo.id_ensayo = public.ensayo_rol.id_ensayo)
   INNER JOIN public.rol ON (public.ensayo_rol.id_rol = public.rol.id_rol)
   INNER JOIN public.rol_ip ON (public.rol.id_rol = public.rol_ip.id_rol) INNER JOIN public.ip
  ON (public.rol_ip.id_ip = public.ip.id_ip) INNER JOIN public.usuario_ip
  ON (public.usuario.usuario_login = public.usuario_ip.usuario_login) AND
```

(public.usuario_ip.id_ip = public.ip.id_ip), public.ip_rango

WHERE

(public.usuario_ensayo.usuario_hab = true) **AND**

(public.ip.direccion_ip **BETWEEN** '10.24.0.1' AND '10.128.8.254')

Para el ejemplo anterior el resultado obtenido fue de 218 filas, en un tiempo de 1,41 segundos.

Las pruebas realizadas a una BD, nunca pueden tomarse como resultados reales, aunque dan un aproximado al mismo, dependiendo del grado de acercamiento que se utilice a los procesos de la vida real. Aún así, la implantación y utilización de una BD, está marcada por varios factores entre ellos: cantidad de usuarios a conectarse y el grado de concurrencia de las solicitudes hechas por lo mismo, los cuales hay que tener en cuenta siempre. Además, es necesario velar el cumplimiento de todos los requerimientos no funcionales propuestos para la BD, ya que validan el funcionamiento correcto y el máximo rendimiento de la misma. De aquí que, aunque las pruebas realizadas brinden resultados que puedan parecer alentadores, no pueden ser motivo de confianza, todo lo contrario, la mejor prueba que se le puede realizar a cualquier resultado informático lo constituye la interacción directa del usuario, de aquí los procesos de mantenimiento de software y la realización de versiones del mismo buscando mejorar o resolver, posibles errores que se detecten durante su vida útil.

Luego de las pruebas realizadas se obtuvo como resultado el cumplimiento de los requisitos funcionales propuestos por los analistas.

Conclusiones

En el capítulo se han analizado una serie de aspectos y consideraciones que se deben tener en cuenta para realizar un diseño eficiente de una BD, como son: la integridad del diseño de la BD, la redundancia de información presente en la misma, además de la normalización, la cual en la BD propuesta se llevó hasta la tercera forma normal (3NF). También se analizó la seguridad que debe presentar la misma, realizando un control de acceso a la BD, haciendo uso en primer lugar de los archivos de configuración del SGBD seleccionado: PostgreSQL, y por otra parte velando el cumplimiento de los requisitos funcionales propios del sistema, de validar los accesos de acuerdo al IP desde donde se realiza la conexión y el rol asignado al usuario en la misma. Para tener constancia de las acciones realizadas por los usuarios en la BD en cada momento, se hizo un análisis de la trazabilidad de las acciones, almacenándose los datos necesarios para realizar auditorias de control.

Se realizó un llenado de la BD con la herramienta EMS Data Generador For PostgreSQL 2005, seleccionándose algunas consultas que se valoran con mayor grado de ocurrencia, utilizándose para la realización de pruebas y la validación del cumplimiento de los requerimientos funcionales.

Conclusiones

El gran volumen de información que se genera a través de los EC, debe almacenarse en los CRD, constituyendo un problema hoy en día, a la hora de escoger, cual SGBD facilita el trabajo para resolver las necesidades de los usuarios finales.

Mediante este trabajo se dio cumplimiento a los objetivos propuestos:

- Se realizó una validación teórica del diseño de la BD, en la que se analizó la integridad, normalización, seguridad y redundancia de información. Además se analizó la trazabilidad de las acciones a realizarse en la BD.
- Una vez finalizada la validación teórica, se diseñó e implementó una BD relacional, utilizando el SGBD PostgreSQL debido a las facilidades de alta escalabilidad que nos brinda, permitiendo almacenar y manejar el gran volumen de información referente a los EC desarrollados en el CIM y los diferentes centros del polo. El diseño obtenido cuenta con 56 tablas, que almacenan la información necesaria para el trabajo correcto de los cuatro módulos en los que se divide el Sistema de Manejo de Datos de los Ensayos Clínicos.
- Se realizó una validación funcional, mediante la cual se realizaron pruebas a la BD a través de la utilización de herramientas para el llenado de BD y algunas consultas seleccionadas, obteniendo como resultado el cumplimiento de todos los requisitos funcionales planteados por los analistas.

Por todo lo antes expuesto se concluye que todos los objetivos propuestos en la investigación se cumplieron satisfactoriamente, incluyendo una serie de recomendaciones para el trabajo futuro.

Recomendaciones

A lo largo de todo el proceso de desarrollo del trabajo queda claro que esta propuesta es sólo la primera fase de un proyecto que puede ser tan ambicioso como se desee. Por lo que se hacen llegar las siguientes recomendaciones:

- Diseñar e Implementar dos BD, una para almacenar la información de cada ensayo clínico que se confeccione en el CIM y otra para la información manejada en los módulos de administradores y diseñadores, la cual resulta común para todos los ensayos clínicos.
- Que se aplique y se haga extensivo a todos los hospitales del país.

Referencias Bibliográficas

1. Jiménez, G.y.P., Maria A y López, Isabel Sistema de reporte periódico de la marcha de los Ensayos Clínicos en la red nacional de coordinación de ensayos clínicos en Cuba. 2004 [cited 2006]; Available from: http://www.informaticamedica.org/I04/papers/jimenezrivero_51.pdf.
2. ClinicalTrials. What is a clinical trial? 2007 [cited; Available from: http://clinicaltrials.gov/ct/info/whatis#whatis.
3. OJD. La OMS promueve el registro de ensayos clínicos. 2006 [cited 2006]; Available from: http://www.elmundo.es/elmundosalud/2006/05/18/industria/1147976599.html.
4. CIM. Centro de Inmunología Molecular 2000-2007 [cited; Available from: http://www.cim.sld.cu/.
5. León, K.y.R., Mayra (2006) "Informatización de los Ensayos Clínicos e integración en la Red Nacional de hospitales" **Volume**,
6. salud, B. Las fases de los ensayos clínicos 2000-2007 [cited; Available from: http://www.buenasalud.com/lib/ShowDoc.cfm?LibDocID=3325&ReturnCatID=348.
7. Ruiz, F. El modelo de datos jerárquico. 2006 [cited; Available from: http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001_MAMoraga.pdf.
8. Moreno Ortiz, A. Bases de datos: Modelos de datos. 2000 [cited; Available from: http://elies.rediris.es/elies9/4-2-3.htm.
9. Franco, R.: Base de datos relacionales. 2006 [cited; Available from: http://www.google.com.cu/search?hl=es&q=Modelos+BD+de+red&meta=.
10. Cuervo, L.G.y.V., América y Clark, María Luisa. El registro internacional de ensayos clínicos. 2006 [cited; Available from: http://www.who.int/ictrp/Editorial_Cuervo_2006.pdf.
11. Jiménez, G.y.D., Mara. El Registro Electrónico del Proceso de Recolección en un Ensayo Clínico. 2002 [cited 2007; Available from: http://www.cecam.sld.cu/pages/rcim/revista_4/articulos_html/gladys.htm.

12. Fernández, A.y.M., Anabely y Pérez, Liesel (2005) BASE DE DATOS PARA ENSAYOS CLÍNICOS EN LA PROVINCIA DE CIENFUEGOS. **Volume**,

13. Masip, D. Qué es Oracle. 2002 [cited; Available from: http://www.desarrolloweb.com/articulos/840.php.

14. Características de SQL Server. 2001 [cited; Available from: http://www.sqlmax.com/caracter.asp.

15. Microsoft. ¿Por qué actualizar? 2006 [cited 2007 21 febrero]; Available from: http://www.microsoft.com/spain/sql/2000/productinfo/whyup.aspx.

16. Dubois, P. Las principales características de MySQL. 2007 [cited; Available from: http://dev.mysql.com/doc/refman/5.0/es/features.html.

17. Group, P.G.D. PostgreSQL. 2007 [cited; Available from: http://www.postgresql.org/.

18. Pecos, D. PostGreSQL. 2005 [cited 2007]; Available from: http://www.netpecos.org/docs/mysql_postgres/x15.html.

19. Mendoza, M.A. Metodologías De Desarrollo De Software. 2004 [cited 2006 22 febrero]; Available from: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

20. Vara, J.M. Análisis y Diseño asistido por ordenador:CASE. 2007 [cited; Available from: http://kybele.escet.urjc.es/documentos/HC/HC4GL2007-T1-CASE-Compl.pdf.

21. Guevara Injoque, M.A.y.F.N., César R. Manual de Erwin. 2002 [cited; Available from: http://www.jp.unlugar.com/Erwin.pdf.

22. Softonic, E.d. CASE Studio 2. 2005 [cited; Available from: http://case-studio.softonic.com/.

23. Rational Rose. 2005 [cited; Available from: http://www.indudata.com/1rational_rose.htm.

24. EMS SQL Manager 2005 for PostgreSQL 3.6. 2004-2007 [cited; Available from: http://www.download3000.com/download_5711.html.

25. Solutions, E.D.M. EMS Data Generator for PostgreSQL. 2007 [cited; Available from: http://www.sqlmanager.net/en/products/postgresql/datagenerator.

26. Solutions, E.D.M. EMS SQL Query for PostgreSQL. 2007 [cited; Available from: http://www.sqlmanager.net/en/products/postgresql/query.

27. Garcia Chavez, C.A. Diseño de base de datos relacionales. 2001 [cited; Available from: http://www.mailxmail.com/curso/informatica/disenobasesdatosrelacionales/capitulo7.htm.

28. CISTERNA NEIRA, M., METODOS DE OPTIMIZACION DE CONSULTAS PARA EL LENGUAJE SQL. 2002.

29. Eduardo. Normalización de Bases de Datos. . 2003 [cited; Available from: http://www.mysql-hispano.org/page.php?id=16&pag=1.

30. Piñero, R. Las Bases de Datos. 2007 [cited; Available from: http://www.techtear.com/2007/04/30/las-bases-de-datos/.

31. John Worsley, J.D. PostgreSQL Práctico. 2001 [cited; Available from: http://www.sobl.org/traducciones/practical-postgres/node40.html.

32. Cano, J.J. Trazabilidad de las Operaciones Electrónicas 2005 [cited; Available from: http://www.isaca.org/Template.cfm?Section=Home&CONTENTID=24655&TEMPLATE=/ContentManagement/ContentDisplay.cfm.

Bibliografía

- Arsys.es, *Bases de datos PostgreSQL*. 2007. [Disponible en: http://arsys.es/ayuda/guias/postgresql.htm] (23/3/2007)
- Carmona Ibáñez, G. *Seguimiento y control de los Ensayos Clínicos. Elaboración de una Base de Datos*. [Disponible en: http://www.sefh.es/revistas/vol19/n5/295_298.PDF] (27/3/2007).
- ClinicalTrials.gov *Resource Information*. 2005. [Disponible en: http://www.clinicaltrials.gov/ct/info/resources;jsessionid=66BEC502516F0D21A62D3577E20BE589] (12/1/2006).
- Current Control Trials. *Welcome to Current Controlled Trials*. 2005. [Disponible en: http://www.controlled-trials.com/]
- Download3000.com, EMS SQL Query 2005 for PostgreSQL 2.2. 2007. [Disponible en: http://www.download3000.com/download_13105.html (5/5/2007)
- DownloadJunction, *EMS Data Generator 2005 for PostgreSQL*. 2007. [Disponible en: http://www.downloadjunction.com/product/store/30302/index.html] (5/5/2007).
- Ferraggine, Viviana. *ERWIN Herramienta CASE para el modelado y desarrollo de una Base de Datos*. 2006. [Disponible en: http://www.exa.unicen.edu.ar/catedras/dbases1/Apuntes/Erwin.pdf] (23/1/2007).
- FRANCO, R.: *Base de datos relacionales*. 2006. [Disponible en: http://www.google.com.cu/search?hl=es&q=Modelos+BD+de+red&meta=
- García, Ignacio. *Base de Datos Modelo en Red General*. [Disponible en: http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001_IGarcia.pdf (12/3/2007)
- IBM, *Rational Rose Data Modeler.* [Disponible en: http://www-306.ibm.com/software/info/ecatalog/es_AR/products/H106683S95271V02.html] (28/4/2007).
- Jacobson Ivar, Booch Grady y Rumbaugh James. *El proceso unificado de desarrollo de software Volumen 1*. Félix Varela. La Habana, Cuba. 2004.
- Legislación vigente de medicamentos huérfanos, *Ensayos Clínicos*. 2001. [Disponible en: http://www.ub.es/legmh/ereensay.htm#introduccion] (23/1/2007).

- Márquez Andrés, Maria M. *Ventajas e inconvenientes de los Sistemas de Base de Datos*, 2001. [Disponible en: http://www3.uji.es/~mmarques/f47/apun/node7.html] (25/5/2007).
- Mato García, Rosa María. *Sistemas de Bases de Datos*. Pueblo y Educación. Ciudad habana, Cuba. 2005.
- Medrano González, Francisco. *Medicina basada en evidence*. 2005. [Disponible en: http://personal.telefonica.terra.es/web/fmedranog/MBe3.htm#315] (19/11/2006).
- Padrón Torres, Liudmila. Tono: *Revista técnica de la empresa ETECSA*. Página 54. Ciudad Habana, Cuba. 2006
- PostgreSQL. *Kit de Prensa de PostgreSQL 8.2*. 2007. [Disponible en: http://www.postgresql.org/about/press/presskit82.html.es] (10/5/2007).
- Presuman, Roger S*. Ingenieria de software Un enfoque práctico Parte 1*. Félix Varela. Página 239. Ciudad Habana, Cuba. 2005.
- Roberts L., Aluned I. *Ensayo Clínico*. 2006.[Disponible en: http://www.usal.es/~sabus/site%20med/descargas/del%20pino.pdf] (24/11/2006).
- Ruiz, F. *El modelo de datos jerárquico*. 2006 [Disponible en: http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T0001_MAMoraga.pdf
- SQL Manager.Net, *EMS Data Generator for PostgreSQL*. 2007. [Disponible en: http://www.sqlmanager.net/products/postgresql/datagenerator] (5/5/2007).
- Brookins, Andrew. *Características de PostgreSQL.* 2001. [Disponible en: http://www.sobl.org/traducciones/practical-postgres/node19.html].

Anexos

Anexo 1: Descripción de las clases que son especialización de las clases *variable*, *sitio* y *query*.

Nombre: var_entero	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_valor_var	integer
valor_entero	Integer
estado_valor	String
valor_validado	Boolean
dia_incluido_entero	Integer
mes_incluido_entero	Integer
anno_incluido_entero	Integer
hora_incluido_entero	Time
descripción	String

Nombre: var_decimal	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_valor_var	integer
valor_decimal	Integer
estado_valor	String
valor_validado	Boolean
dia_incluido_decimal	Integer
mes_incluido_decimal	Integer
anno_incluido_decimal	Integer
hora_incluido_decimal	Time
descripción	String

Nombre: var_texto	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_valor_var	integer
valor_texto	Integer

estado_valor	String
valor_validado	Boolean
dia_incluido_texto	Integer
mes_incluido_texto	Integer
anno_incluido_texto	Integer
hora_incluido_texto	Time
descripción	String

Nombre: var_fecha	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_valor_var	Integer
dia_var_fecha	Integer
mes_var_fecha	Integer
anno_var_cheha	Integer
estado_valor	String
valor_validado	Boolean
dia_incluido_fecha	Integer
mes_incluido_fecha	Integer
anno_incluido_fecha	Integer
hora_incluido_fecha	Time

Nombre: var_nomenclador	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_valor_var	integer
valor_nomenclador	String
estado_valor	String
valor_validado	Boolean
dia_incluido_valornom	Integer
mes_incluido_valornom	Integer
anno_incluido_valornom	Integer
hora_incluido_valornom	Time

Nombre: var_caracter
Tipo de clase: Entidad

Atributo	**Tipo**
id_valor_var	Integer
valor_caracter	String
valor_validado	Boolean
estado_valor	String
dia_incluido_caracter	Integer
mes_incluido_caracter	Integer
anno_incluido_caracter	Integer
hora_incluido_caracter	Time
descripción_caracter	String

Nombre: sitio_hospital	
Tipo de clase: Entidad	
Atributo	**Tipo**

Nombre: query_variable	
Tipo de clase: Entidad	
Atributo	**Tipo**

Nombre: query_inconsistencia_valor	
Tipo de clase: Entidad	
Atributo	**Tipo**

Anexo 2. Descripción de las clases que forman parte de los diferentes tipos en que se puede derivar o validar una variable.

Nombre: validación_valores	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_valores	Integer
valores	String

Nombre: validación_por_formula	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_elemento_formula	Integer
posic_elemento	Integer
token_elemento	String

Nombre: valor_elemento_formula	
Tipo de clase: Entidad	
Atributo	**Tipo**
Elemento_formula	String

Nombre: variable_formula	
Tipo de clase: Entidad	
Atributo	**Tipo**

Nombre: validación_concatenar	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_concatenar	Integer
pos_variable	Integer

Nombre: validar_numerico	
Tipo de clase: Entidad	
Atributo	**Tipo**
obligatorio	Boolean
cant_cifras_enteras	Integer
cant_cifras_decimales	Integer
unidad_medida	String

Nombre: validar_texto	
Tipo de clase: Entidad	
Atributo	**Tipo**
obligatorio	Boolean
cant_caracteres	Integer

Nombre: validar_nomenclador	
Tipo de clase: Entidad	
Atributo	**Tipo**
inicio_rango	String
fin_rango	String
obligatorio	Boolean

Nombre: validar_fecha	
Tipo de clase: Entidad	
Atributo	**Tipo**
dia_inicio_rango_fecha	Integer
mes_inicio_rango_fecha	Integer
anno_inicio_rango_fecha	Integer
dia_fin_rango_fecha	Integer
mes_fin_rango_fecha	Integer
anno_fin_rango_fecha	Integer
dia_obligatorio	Boolean
mes_obligatorio	Boolean
anno_obligatorio	Boolean

Anexo 3. Descripción de las tablas del modelo Entidad-Relación que se generan a partir de relaciones, agregándoseles atributos.

Nombre: ensayo paciente		
Descripción: Esta entidad surge de la relación entre la tabla paciente y sitio, a la que se agregan atributos referentes a los pacientes.		
Atributo	**Tipo**	**Descripción**
id_sitio	Integer	Llave primaria foránea tomada de la entidad sitio
id_paciente	Integer	Llave primaria foránea tomada de la tabla paciente
no_inclusion_paciente	Integer	Número de inclusión del paciente
dia_inclusion_paciente	Integer	Fecha de inclusión del paciente
mes_inclusion_paciente	Integer	
anno_inclusion_paciente	Integer	
estado_paciente_ensayo	Varchar	Mal_incluido, Mal_incluido_waiver, Incluido, En_curso_Tratamiento, En_curso_Seguimiento, Interrupción_Tratamiento, Interrupción_Completada, Estudio_Completado

Nombre: submodelo_variable		
Descripción: Esta entidad surge de la relación entre la tabla paciente y sitio, a la que se agregan atributos referentes a los pacientes.		
Atributo	**Tipo**	**Descripción**
id_submodelo	Integer	Llave primaria foránea que la toma de la entidad submodelo.
id_variable	Integer	Llave primaria que la toma de la entidad variable.
fila	Integer	Parámetro para obtener la posición a la hora del diseño desde la base de datos.
columna	Integer	Parámetro para obtener la posición a la hora del diseño desde la base de datos.

Nombre: usuario_ensayo		
Descripción: Esta entidad surge de la relación entre la tabla usuario y ensayo, a la que se agregan atributos referentes a los usuarios.		
Atributo	**Tipo**	**Descripción**
id_ensayo	Integer	Llave primaria foránea tomada de la entidad ensayo
usuario_login	Varchar	llave primaria foránea tomada de la entidad usuario
usuario_hab	Boolean	true si el usuario puede acceder al ensayo y false en caso contrario
usuario_conec	Boolean	true si el usuario habilitado se encuentra trabajando en el ensayo y false si el usuario habilitado no se encuentra trabajando en el ensayo.

Glosario de Términos

ACID: (Atomicity, Consistency, Isolation, Durability). Propiedades importantes de las BD relacionales.

Atomicity (atomicidad): Propiedad en la que el trabajo de transacciones se realiza en su totalidad o no se realiza en ningún caso.

BD: (Base de Datos). Colección de datos interrelacionados que se puede utilizar por uno o más programas de aplicación, puede considerarse una colección de datos variables en el tiempo.

CIM: (Centro de Inmunología Molecular). Centro dedicado a la comercialización de los Ensayos Clínicos.

Consistency (consistencia): Propiedad de la BD que después de las transacciones los datos deben quedar consistentes.

CRD: (Cuaderno de Recogida de Datos). Cuadernos que son capaces de almacenar todas las variables recogidas durante la ejecución del ensayo, siendo capaz de registrar todos los datos que se recogen en el protocolo.

Cronograma: Constituye la distribución de los modelos que serán llenados al paciente en las diferentes visitas programadas.

Datos primarios: Son todos aquellos documentos de los cuales se obtiene información que se registra en la Historia Clínica del paciente. Ej. Rayos X, entre otros.

DBMS: (Sistema Manejador de Base de Datos).

Durability (durabilidad): una transacción debe ser guardada permanentemente con independencia de cualquier tipo de fallo del sistema.

EC: (Ensayos Clínicos). Estudios clínicos en que se evalúan nuevos fármacos o tratamientos médicos con un protocolo de investigación estrictamente controlado. Permite determinar si un tratamiento o medicamento contribuirá a prevenir, detectar o tratar una enfermedad.

EMS (Electronic MicroSystems): es una Compañía de Tecnología de la Información, uno de sus campos de actividades es el desarrollo de software. EMS se fundó en 1993 e inicialmente se especializó en el desarrollo de aplicaciones de red, BD corporativas y en la construcción de herramientas de automatización de negocios en arquitectura cliente-servidor multi-capa.

En curso de tratamiento: Representa un estado del paciente. Un paciente esta en este estado cuando se le esta aplicando el tratamiento y recogiendo datos.

En curso seguimiento: Representa un estado del paciente. Un paciente esta en este estado cuando ya se le termino de aplicar el tratamiento y se le siguen recogiendo datos.

EPINFO: Es un programa de dominio público de especial utilidad para la Salud Pública. Tiene un sistema fácil para construir bases de datos. Es un conjunto de programas de microcomputadora para manejar datos en formato de cuestionario y para organizar los resultados en texto que puede formar parte de informes escritos.

Estado completo: Se refiere al estado del modelo. Cuando se registran todos los datos.

Estado Firmado: Ya se registraron los datos y el IP. Se ha insertado la firma dejando el modelo en formato de solo lectura

Estado Incompleto: Cuando le faltan datos por ser registrados.

Estado Monitoreado: Cuando se han registrado todos lo datos. Hay que tener presente si hay datos inconsistentes o no.

Estado vacío: Se encuentra en este estado cuando no se le han entrado datos.

Estudio completado: Representa un estado del paciente. Un paciente esta en este estado cuando se le han recogido todos los datos requeridos por el estudio.

Incluido: Representa un estado del paciente. Esta en estado incluido cuando se le han llenado los modelo de inclusión y exclusión del ensayo y cumple con los criterios.

Inconsistencia: Es la calificación que se le da a una variable que no cumple con las reglas de validación. Dicha inconsistencia pudo surgir por descuido del coordinador al entrar los datos a un modelo del paciente o porque realmente ese sea el valor que tiene para dicho paciente.

Interrupción Completada: Representa un estado del paciente. Un paciente esta en este estado cuando por alguna causa debe ser sacado del ensayo.

Interrupción Tratamiento: Representa un estado del paciente Un paciente esta en este estado cuando por algún evento se le debe dejar de aplicar el tratamiento, pero se le sigue evaluando y recogiendo datos.

Isolation (aislamiento): una transacción no puede usar datos de otra transacción que todavía no ha concluido.

Mal incluido: Representa un estado del paciente. Esta en estado mal incluido cuando se le han llenado los modelo de Inclusión y exclusión del ensayo y no cumple con los criterios.

Mal incluido Waiver: Representa un estado del paciente. Esta en este estado cuando a pesar de estar mal incluido se le sigue evaluando y recogiendo datos.

Parser: Viene de parseador, que es el programa que se encarga del parseo de una consulta. Proceso de traducir la consulta SQL a un árbol lógico en álgebra relacional.

Queries: Constituyen una especie de intercambio entre el Monitor y el Coordinador de la Investigación Clínica o el Gerente de Datos, su objetivo es comunicar y establecer un debate sobre alguna deficiencia encontrada relacionada con los pacientes, los modelos o el ensayo en general.

Script: Se refiere a un archivo de texto que recoge todo el código de la BD, que muestra un guión y conjunto de instrucciones. Es ejecutado por el PostgreSQL para su uso y crear todas las tablas.

SGBD: Conjunto coordinado de programas, procedimientos, lenguajes. Que suministra tanto a usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos, manteniendo su integridad, confidencialidad y seguridad.

Traza Auditoria: El sistema registra todas las acciones que se realizan sobre un ensayo.