Universidad de las Ciencias Informáticas

Facultad 3

"Diseño e implementación de la Base de Datos del proyecto Sistema de Informatización del Registro de la Cámara de Comercio".

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

Autor

Jorge Regalado Martínez

Tutor

Ing. José Sevila Hidalgo

Ciudad de la Habana, Cuba
2012
"Año 54 de la Revolución"

Declaración de autoría

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

Para que así conste firmo la presente a los _____días del mes de _____del año 2012.

_____	_____
Jorge Regalado Martínez	Ing. José Sevila Hidalgo

Agradecimientos

A MIS PADRES POR TODOS LOS SACRIFICIOS QUE HAN HECHO POR MÍ, POR SU PREOCUPACIÓN Y AMOR INCONDICIONAL. SÉ QUE CUMPLO SUS SUEÑOS HOY.

A PEPE, MI TUTOR, QUE MÁS QUE TUTOR HA SIDO MI AMIGO DESDE QUE ESTUDIÁBAMOS JUNTOS.

A TODA MI FAMILIA POR EL CARIÑO QUE SIEMPRE ME HA DADO.

A MI NOVIA LISANDRA POR COMPARTIR JUNTOS HERMOSOS MOMENTOS Y AYUDARME AÚN MÁS DE LO QUE CREE.

A LA FAMILIA ALMANZA POR ACEPTARME COMO UN HIJO. BESOS A ÑAÑA, DANIELA, CONSUELO Y ABUE. UN ABRAZO AL DANI Y AL CHARLIE.

A LISET Y MARÍA ELENA POR ACEPTARME IGUALMENTE EN SU FAMILIA.

A MIS AMIGOS EN LA UNIVERSIDAD: ALIANDRO, GABRIEL, HURSHEL, DAYRON, CARMEN, ODISLEYSI, ODALYS, ADRIÁN, JAVIER Y A TODOS LOS MAMAOS. A LEO, UN ABRAZO BIEN GRANDE PARA TI MI HERMANO.

A TODOS LOS QUE, DE UNA FORMA U OTRA, AYUDARON EN LA CONFECCIÓN DE ESTE TRABAJO. A JORRÍN Y ALAIN POR TODA SU AYUDA EN EL PROYECTO. A NELSON Y JOSÉ MANUEL POR COMPARTIRME SUS IDEAS.

A TODO EL TRIBUNAL POR LAS SUGERENCIAS, INCONFORMIDADES Y RECOMENDACIONES QUE ME HICIERON, PERMITIENDO LA CREACIÓN DE UN MEJOR TRABAJO DE DIPLOMA.

Dedicatoria

A MIS PADRES,

A MIS HERMANOS,

A MIS PRIMOS YOSBEL, YAN Y ALEJANDRO.

A TODA MI FAMILIA.

A MI HERMOSA LISANDRA.

DE MANERA ESPECIAL QUISIERA DEDICARLE ESTE TRABAJO A AQUELLOS COMPAÑEROS CON LOS QUE COMPARTÍ CINCO AÑOS DE CARRERA Y ME BRINDARON LAS MAYORES ENSEÑANZAS DE LA UNIVERSIDAD: TOLEDO, JOEL, MAIKEL, DANIEL, ALAIN, MALENA Y YAIDEL.

Resumen

La Cámara de Comercio de la República de Cuba tiene una responsabilidad social con el crecimiento de la sociedad y la economía cubana, por lo que las acciones que se acometen en su seno tributan al progreso y bienestar del pueblo cubano. Con bases en los principios de la sociedad cubana y con ciertas aspiraciones, la Cámara de Comercio enfrenta un proceso de modernización de los procesos que se desarrollan dentro de su marco de trabajo.

En el presente trabajo de diploma es expuesto el diseño e implementación de la base de datos del Sistema de Informatización del Registro de la Cámara de Comercio (SIRECC), aplicación informática que mejorará el control y gestión de los datos almacenados sobre los afiliados comerciales a la misma. Para la realización de este trabajo se empleó la metodología de desarrollo Rational Unified Process (RUP) y se emplearon Unified Modelling Language (UML) y E/R Studio como lenguaje y herramienta de modelado respectivamente. Se utilizó además PostgreSQL como Sistema Gestor de Bases de Datos para el desarrollo de la misma.

Tabla de contenidos

Índice de figuras

Índice de tablas

Introducción

El nombre Cámara de Comercio aparece por primera vez en la Historia en 1599 en Francia. Esta era una institución que tenía como objetivo asesorar al Estado sobre cuestiones económicas. En el siglo XVIII se crean las primeras cámaras en algunos países de la Europa Occidental y para mediados del siguiente siglo existían cámaras en casi todos los países del continente euroasiático.

No es hasta 1876 que se establece en La Habana la primera asociación que agrupaba a los comerciantes de la ciudad. Este gremio de comerciantes un año más tarde cambió su nombre a Junta General de Comercio, y posteriormente, ese mismo año, volvió a cambiar de nombre a Cámara Oficial de Comercio, Industria y Navegación. El nombre oficial siguió cambiando varias veces y en 1927 tomó el nombre de Cámara de Comercio de la República de Cuba. En 1963, se disolvió esta cámara y de acuerdo con la ley No. 1901 del 1ro de febrero de ese mismo año se creó la Cámara de Comercio que actualmente se encuentra en vigencia.

Como Cámara de Comercio se entiende a una institución civil que promueve la actividad de la libre empresa al amparo de la Constitución del Estado y demás normas vigentes. Esta reúne a empresas vinculadas al comercio, los servicios y la industria y consta de reconocimiento ante el Estado, permitiéndole orientar a las entidades pertenecientes, sobre las mejores alternativas para desarrollar la actividad comercial y empresarial. En el caso de Cuba, la Cámara de Comercio, representa un instrumento para la reincorporación de la economía en el ámbito de las relaciones económicas internacionales.

Dentro las funciones de la Cámara están el intercambio de información sobre las posibilidades de negocios e inversiones en Cuba y en el extranjero, la propuesta de ofertas de productos y servicios exportables de la isla, la sustitución de importaciones para beneficio de la economía nacional y la prestación de servicios a sus compañías cubanas asociadas y también a las sucursales de las compañías extranjeras y empresarios foráneos acreditados en territorio cubano. Además la Cámara de Comercio debe responder por los legítimos intereses de sus asociados, así como promover la libre empresa y facilitar la vinculación de oportunidades de negocios a sus asociados.

El punto de entrada para poder solicitar los servicios de la Cámara de Comercio es el proceso de afiliación a la misma. Este proceso puede ser solicitado por cualquier

empresa debidamente instituida y significa el registro de una serie de datos de contacto, así como el pago de los derechos de afiliación. Esto significa que cualquier institución empresarial que genere actividad económica en el país puede formar parte de la Cámara de Comercio. Esta inscripción es obligatoria para toda persona natural o jurídica que se dedique a ejercer el comercio, que se haya declarado como comerciante individual o constituido como comerciante social.

La Cámara de Comercio juega un importante papel en las aspiraciones de Cuba de convertirse en una potencia en el área del Caribe a través del incremento de la actividad turística, las inversiones extranjeras y el comercio. Sin embargo, el cumplimiento de este objetivo se ve afectado por la manera en que se realizan los procesos de gestión de los datos referentes a los afiliados a la Cámara de Comercio. Esta información es almacenada muchas veces de manera tradicional, o sea, manualmente; lo que no brinda siempre la misma fiabilidad que presentan otros formatos de almacenamiento de datos, ni aseguran que la información sea la correcta. La información almacenada de esta manera tiene además que enfrentar problemas de:

- Difícil acceso.
- Redundancia y desactualización.
- Mal cuidado o pérdida.
- Errores e inconsistencia.

Se emplean también herramientas de software ineficientes y de corto alcance como Microsoft Access y Microsoft Excel, que no son las más adecuadas para almacenar grandes volúmenes de datos y no permiten gestionar toda la información requerida. Esto imposibilita que el seguimiento de los datos y la realización de consultas a estos se ejecute con la celeridad y continuidad requerida por los distintos socios comerciales, e incluso se pueden presentar problemas de incongruencias y duplicación de la información. La información no se encuentra totalmente almacenada en un solo lugar por lo que se consume mucho tiempo en obtenerla. Consiguientemente se necesita crear una solución inmediata que permita una mejor gestión de los acuerdos y compromisos que se generan como producto de la actividad comercial, donde se empleen otras técnicas que posibiliten la centralización de estos datos, la actualización y seguimiento continuo de la información.

Las relaciones comerciales de Cuba con el mundo se han visto en aumento desde hace ya varios años, apoyadas por el prestigio moral que ha mantenido la isla siempre, en Latinoamérica fundamentalmente, y por el movimiento revolucionario que se ha

manifestado en algunos países del área como Venezuela, Bolivia, Brasil y Ecuador, sin dejar atrás a grandes socios comerciales como China y Rusia. En el marco del avance que se ha ido obteniendo en el desarrollo de la economía del país y las relaciones económicas y comerciales en incremento, la Cámara de Comercio de Cuba y el Centro de Gobierno Electrónico (CEGEL) de la Universidad de las Ciencias Informáticas (UCI), acordaron la creación de un sistema informático que permita el tratamiento automatizado de la información referente a sus afiliados comerciales y la correcta gestión de los datos que estos generan. Surge de esta manera el proyecto Sistema de Informatización del Registro de la Cámara de Comercio (SIRECC) cuyo principal objetivo es la creación de dicho sistema. Este sistema necesita un medio de almacenamiento que permita gestionar la información relacionada con los socios afiliados a la Cámara de Comercio.

A raíz de lo planteado, se reúnen las condiciones para expresar el **problema a resolver** que da origen al presente trabajo de diploma: ¿Cómo almacenar la información gestionada por el sistema SIRECC de manera que se permita la centralización de los datos, un mejor tratamiento y consulta de la información?

Teniendo en cuenta el problema a resolver planteado previamente se propone como **objeto de estudio:** diseño e implementación de BD relacionales, y como **objetivo general:** Realizar el diseño y la implementación de una base de datos para el sistema SIRECC, que permita la centralización de los datos, un mejor tratamiento y consulta de la información.

En este contexto queda definido como **campo de acción:** el diseño e implementación de la BD del SIRECC.

A partir del problema anteriormente expuesto se plantea que: con un apropiado diseño e implementación de la base de datos del sistema SIRECC, se logrará el almacenamiento de los datos permitiendo la centralización de estos y un mejor tratamiento y consulta de la información, siendo esta la **idea a defender** de la investigación**.**

Los **objetivos específicos** de la investigación son:

- Definir el marco teórico que sustenta la investigación.
- Diseñar e implementar la propuesta de base de datos para el proyecto SIRECC.
- Validar la solución propuesta.

Para garantizar el cumplimiento de los objetivos propuestos se proponen las siguientes **tareas** de la investigación:

- Estudio del estado del arte que fundamenta el objeto de investigación.
- Selección de la herramienta de modelado.
- Selección del gestor de base de datos.
- Estudio de los métodos de diseño de bases de datos.
- Realización el Modelo Entidad-Relación.
- Realización pruebas para comprobar el correcto funcionamiento de la base de datos.

Como **posible resultado** de la investigación se espera: el diseño e implementación de la base de datos del sistema SIRECC.

Entre los métodos empleados para la investigación científica están:

Métodos Teóricos:

- Histórico – lógico: utilizado para realizar el análisis histórico de los distintos sistemas de bases de datos.
- Analítico- sintético: utilizado para analizar la bibliografía referente al trabajo que se realizará.
- Modelación: utilizado para el modelado de los diagramas para el diseño de la base de datos.

Métodos empíricos:

- Observación: para identificar las actividades que se llevan a cabo para registrar los afiliados a la Cámara de Comercio.
- Medición: para detectar posibles errores en el diseño de la base de datos que impidan la ejecución de las funcionalidades descritas.

La investigación ha quedado estructurada en tres capítulos, donde se describe todo lo referente con la misma. El contenido de los mismos se plantea a continuación.

Capítulo 1: Fundamentación Teórica

En este capítulo se presenta información sobre el surgimiento, clasificación, características, gestores de las bases de datos así como los distintos modelos empleados en el diseño de BD. Se estudian además las principales herramientas y metodologías que se emplean en el desarrollo de un software, justificando las empleadas.

Capítulo 2: Descripción y análisis de la solución propuesta

En este capítulo se describe el Modelo Entidad – Relación correspondiente a la base de datos, teniendo en cuenta los requisitos funcionales y no funcionales trazados, y se describe brevemente algunas de las tablas del diseño y los campos que las componen.

Capítulo 3: Validación del diseño realizado

En este capítulo se ofrece información acerca de la validación del diseño de la base de datos propuesta, se muestran las reglas de integridad, la normalización y análisis de la redundancia para obtener un diseño con calidad. Se describen además los distintos tipos de pruebas realizadas para darle validez al diseño de la base de datos.

Capítulo 1: Fundamentación teórica.

En la actualidad las aplicaciones informáticas que gestionan información son cada vez más utilizadas, debido a que posibilitan suplir los procedimientos tradicionales de administración de la información por métodos automatizados que la almacenen con rapidez, calidad y seguridad, disminuyendo los costos y aumentando el control sobre la misma. Estas aplicaciones tienen sus propios mecanismos de almacenamiento y manejo de los datos, aspecto en el cual las bases de datos (BD) han demostrado su eficiencia y superioridad. En la actualidad representa no solamente un error tecnológico, sino además económico no emplear BD en sistemas de gestión.

En este capítulo se expone el marco teórico y conceptual que soportan la presente investigación, así como las tendencias actuales de las tecnologías más modernas utilizadas en la creación y tratamiento de BD al igual que de las herramientas que permiten la administración de las mismas, los Sistemas Gestores de Bases de Datos (SGBD). Se realiza un análisis histórico de las BD, sus antecedentes, tendencias, la importancia y ventajas de su uso para una organización. Se analizan además las metodologías de desarrollo que guían el proceso de creación del software, las herramientas y lenguaje de modelado más utilizados, enfatizando en los seleccionados para el diseño e implementación de la BD del proyecto SIRECC.

1.1. Metodologías de Desarrollo

Las metodologías de desarrollo de software constituyen un conjunto de técnicas, herramientas, procedimientos, y soportes documentales que sirven de ayuda a los desarrolladores para producir eficientemente un software de calidad. Estas garantizan una mayor transparencia y control sobre el proceso y la entrega en el tiempo establecido. Desarrollar software de calidad depende en gran medida de una variedad de actividades y etapas, por lo que la adopción de una metodología que se ajuste a las necesidades y al equipo de desarrollo es trascendental para el éxito del producto.

La implantación de una metodología en el proceso de desarrollo de un software brinda las siguientes capacidades:

- Guías para estimar costos.
- Manejo del proyecto en las tareas y entregas.
- Políticas y procedimientos para asegurar la calidad del software.

- Descripción de los roles.
- Medidas y métricas.

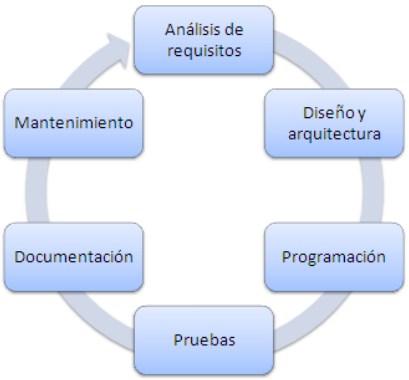


Fig. 1.1 Proceso de Desarrollo de Software.

1.1.1 Extreme Programming (XP)

Extreme Programming, o Programación Extrema en español, es la más destacada de las llamadas metodologías ágiles. El desarrollo con XP se hace de forma iterativa e incremental y dentro de sus características se encuentran:

- Pruebas unitarias continuas: Son realizadas frecuentemente, repetidas y automatizadas.
- Programación por parejas: Las tareas de desarrollo se recomiendan sean realizadas por parejas de desarrolladores.
- Corrección: Antes de añadir una nueva funcionalidad se corrigen todos los errores encontrados.
- Simplicidad en el código: Es más sencillo cambiar algo pequeño cambiar algo grande.
- Frecuente interacción: El cliente siempre es tenido en cuenta. Se recomienda que un representante del cliente trabaje junto con el equipo de desarrollo.
- Refactorización: Algunas partes del código son reescritas para aumentar su legibilidad y mantención.

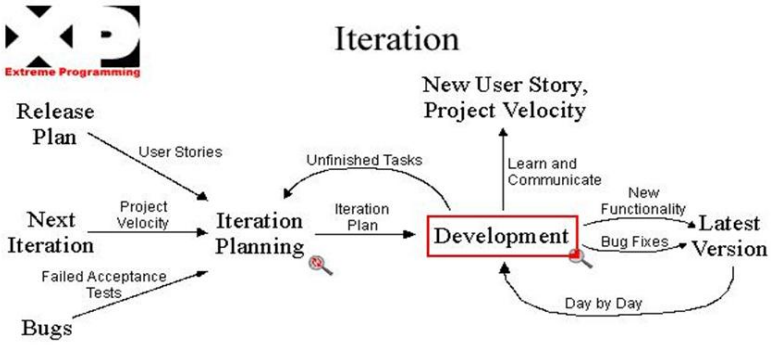


Fig. 1.2 Modelo Visual de XP.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. (Solis Álvarez, y otros, 2007)

1.1.2 Rational Unified Process (RUP)

RUP es una de las metodologías modernas que más auge tiene a nivel mundial en lo referente a producción de software, la cual ha sido refinada a través de los años por IBM Rational®. RUP, perteneciente a las llamadas metodologías pesadas por la gran documentación que genera, ha pasado a ser la metodología estándar en proyectos orientados a objetos.

RUP es un proceso de ingeniería de software que provee una metodología disciplinada para asignar tareas y responsabilidades dentro del desarrollo de una organización. Su meta es asegurar la producción de software de alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un cronograma y presupuesto predecible (Rational®, 1998).

Características de RUP:

- Iterativo e incremental.
- Centrado en la arquitectura.
- Guiado por casos de uso.

RUP enriquece la productividad en equipo, proveyendo a cada miembro del equipo con fácil acceso a una base de conocimientos con líneas de trabajo, plantillas y guías

de herramientas para todas las actividades críticas de desarrollo. Permitiendo el acceso de todos los miembros del equipo a la misma base de conocimientos, se hace irrelevante si se trabaja con requerimientos, diseño, pruebas, administración de proyecto o administración de configuración, pues se asegura que todos compartan un lenguaje común, proceso y vista de cómo desarrollar software (Rational®, 2001). Dentro de las mejores prácticas que se manejan actualmente para el desarrollo de software RUP agrupa algunas que le permite ser aplicable a una amplia variedad de proyectos. Entre ellas se encuentran:

- Desarrollar software iterativamente.
- Administración de requerimientos.
- Desarrollo basado en componentes.
- Modelado Visual.
- Verificación de la calidad del software.
- Control de cambios del software.

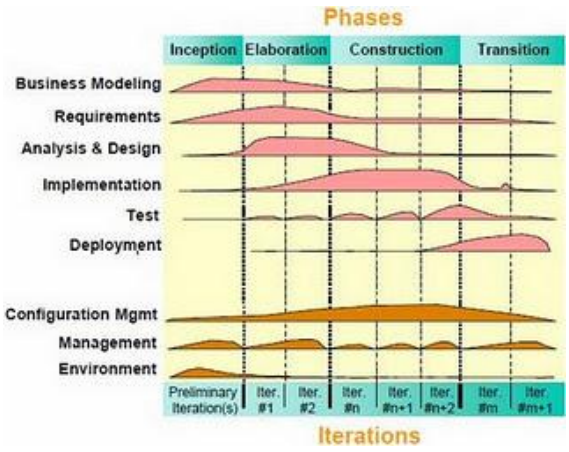


Fig. 1.3 Fases y flujos de trabajo de RUP.

RUP divide el ciclo de desarrollo del software en 4 fases consecutivas: Concepción, Elaboración, Construcción y Transición. Cada una de estas fases concluye con un hito bien definido, un punto en el cual cierta decisión crítica debe ser hecha y por lo tanto algunas metas claves deben ser logradas (Boehm, 1996). RUP define además como elementos de modelado: Flujos de Trabajo, Actividades, Artefactos y Roles.

Un rol define el comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. Una persona puede desempeñar diversos roles, así como un mismo rol puede ser representado por varias personas. Las responsabilidades de un rol son tanto el llevar a cabo un conjunto de actividades como el ser el “dueño” de un conjunto de artefactos (Rational®, 2001).

En el desarrollo de un software el rol del diseñador de la BD es el encargado de identificar los datos, las relaciones entre ellos y las restricciones que deben ser almacenadas en la BD por lo que debe tener amplios conocimiento de los datos que maneja la empresa así como de las reglas del negocio (Esakkirajan, 2007).

1.1.3 Selección de la metodología a emplear

Luego de haber analizado las metodologías se decidió utilizar RUP puesto que es una metodología de la que se tiene amplios conocimientos en la universidad y además porque es adaptable a los cambios de los requerimientos con pocas alteraciones. Permite también una fácil visualización de las interacciones entre el usuario y el sistema mediante una serie de modelos y además de que genera la documentación suficiente para la adecuada comunicación entre el cliente y el equipo de desarrollo, que hace énfasis en una buena captura de requisitos. Además uno de los pilares de XP es la presencia en todo momento de un representante del cliente dentro del equipo de desarrollo, pero en el proyecto SIRECC esto no es posible.

1.2 Base de Datos

Las tecnologías de la gestión de la información se han estado desarrollando desde la década del 50 hasta los poderosos sistemas de BD de la actualidad. El estado actual de las tecnologías de la información es resultado de la creciente demanda de información, convirtiéndose las BD en parte esencial de los sistemas informáticos a los que están integrados.

1.2.1 Surgimiento de las Bases de Datos

Anterior al surgimiento de las BD, los datos de los sistemas informáticos se almacenaban en el sistema de archivos que brindaban los sistemas operativos. Estos eran útiles siempre que el tamaño del archivo fuera manejable, puesto que los cambios se podían realizar sin mayores problemas. Sin embargo cuando el tamaño de este archivo crecía, surgían ciertos inconvenientes. Uno de ellos era debido a que el acceso a los datos se realizaba secuencialmente, por lo que para leer un área específica en un archivo demasiado grande, era necesario recorrerlo entero perdiendo tiempo en la búsqueda, además la introducción de datos nuevos se realiza siempre al final del archivo que nada más se puede abrir para escritura o lectura, no para ambas a la misma vez.

Según Esakkirajan y Sumathi dentro de los problemas que se encuentran en este método de almacenamiento están: (Esakkirajan, 2007)

- Redundancia e inconsistencia de datos: como resultado de que la información está almacenada en distintos archivos, un mismo registro [1] puede estar duplicado en varios de estos archivos, además surge el problema de que las distintas copias de este registro no coincidan.
- Dificultad en el acceso a los datos: como el acceso a los datos es realizado por las aplicaciones que usan estos datos, si surge alguna nueva necesidad de petición de datos es necesario escribir una nueva aplicación que satisfaga esta necesidad.
- Aislamiento de datos: debido a que los datos están en varios archivos, estos pueden que estén en distintos formatos por lo que crear una aplicación para recuperar estos datos es difícil.
- Problemas de integridad: la integridad de los datos se refiere a que los datos sean los correctos de acuerdo con ciertas reglas.
- Problemas de atomicidad: al realizar un cambio en un registro que afecte otro, debe realizarse de manera que se ejecuten ambos y no solo uno.
- Anomalías en el acceso concurrente: el acceso de múltiples usuarios a un mismo archivo puede traer problemas de concurrencia.
- Problemas de seguridad: no todos los usuarios de una base de datos deben poder acceder a todos los datos.

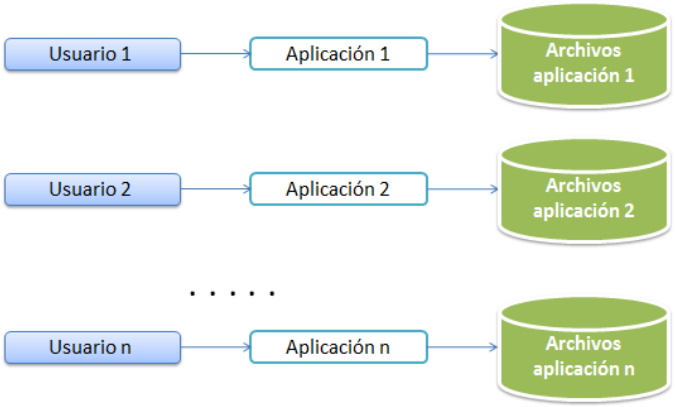


Fig. 1.4 Sistema basado en archivos.

[1] Véase más adelante.

Estas dificultades junto con el inherente problema del tamaño cada vez mayor, motivaron el desarrollo de mejores técnicas de almacenamiento de datos. Es así como surgen las BD a finales de los años 60, permitiendo la realización de operaciones más complejas sobre los datos y dándole solución a los problemas que tenían los sistemas orientados a ficheros.

De esta manera la información quedaría almacenada de manera integrada y compartida puesto que todos los datos quedan unificados en uno solo, de donde se elimina la redundancia y las aplicaciones que anteriormente tenían que acceder a distintos archivos deben acceder ahora a solamente uno.

Siguiendo este modelo los usuarios de la BD, solamente se deben preocupar por la definición externa de los datos, sin preocuparse como es que está definido el objeto internamente. Este nivel de abstracción le permite estar ajeno a cualquier cambio interno en la estructura de los datos, mientras tanto su definición externa no se vea alterada.

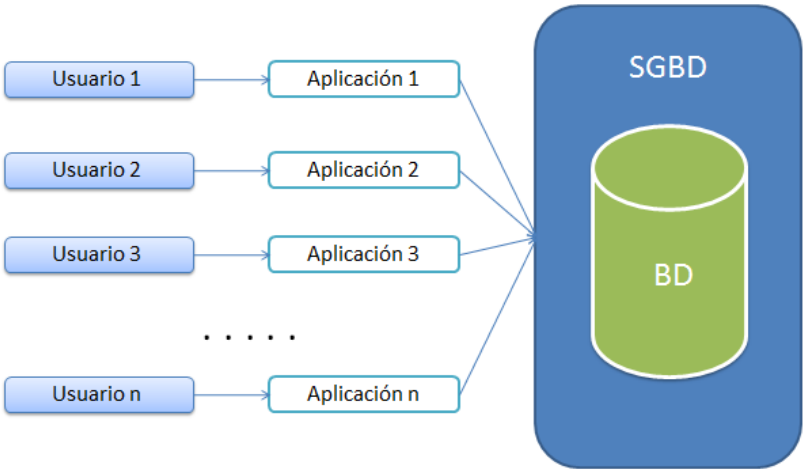


Fig. 1.5 Sistema basado en BD.

Con el avance de las tecnologías y el desarrollo de la computación, los sistemas de BD continuaron avanzando paralelamente a la tecnología, haciéndose cada vez más imprescindibles para el almacenamiento de grandes volúmenes de datos e incorporando características propias de la orientación a objetos más actualmente.

1.2.2 ¿Qué es una Base de Datos?

El término Base de Datos es usado frecuentemente sin saber su verdadero significado, que no es otro que la permanencia de un conjunto de información en cierto lugar

específico. En el mundo actual gracias a los avances tecnológicos esta información es almacenada electrónicamente, lo que permite un mayor grado de organización de la información así como que esta pueda ser accedida más rápidamente y por mayor número de usuarios.

Las BD son uno de los componentes más importante en cualquier sistema de información o gestión y cada vez cobran más importancia debido a la cantidad de información que se genera a cada minuto que necesita ser almacenada para su posterior consulta.

Un sistema de BD es básicamente un sistema computarizado para llevar registros. Los registros (llamados también tuplas o filas), representa un objeto único dentro de una tabla en la BD. En otros términos, una tabla de una BD se puede ver como un arreglo de filas y columnas, donde las filas o registros representan un conjunto de datos relacionados, con una estructura pareja para todas. Un registro representa un ítem único de datos implícitamente estructurados en una tabla.

De acuerdo con Date es posible considerar a la propia base de datos como una especie de armario electrónico para archivar; es decir, es un depósito o contenedor de una colección de archivos de datos computarizados. (Date, 2001)

Sobre la definición del concepto de BD se ha escrito y dicho mucho:

- Una colección de elementos de datos interrelacionados que pueden procesarse por una o más aplicaciones. (Date, 2001)
- Una base de datos es una colección bien estructurada de datos que están relacionados de manera significativa, la que puede ser accedida en diferentes órdenes lógicos. Los sistemas de bases de datos son sistemas donde la interpretación y almacenamiento de la información son de primera importancia. (Hansen, 2006)
- Llamamos base de datos justamente a esta colección de datos recopilados y estructurados que existe durante un periodo de tiempo. (Orallo, 2002)
- Una Base de Datos es un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo, un sistema de archivos electrónico. (García, 2005)
- Colección organizada de datos, relativa a un problema concreto, que puede ser compartida por un conjunto de usuarios/aplicaciones. (Zorrilla, 2003)

Entre las distintas definiciones que plantean distintos autores se puede resumir que una BD es una colección estructurada de datos que se relacionan entre sí de manera lógica y variable en el tiempo, localizada en servidores accesibles en tiempo real para la utilización de estos datos por distintos usuarios.

1.2.3 Componentes de una Base de Datos

Las BD almacenan grandes volúmenes de información, pero son mucho más que simples almacenes de datos.

Según Gary y James Hansen: un sistema de BD es algo más que simples datos o que los datos en combinación con un software de gestión de BD. En una organización, un sistema de BD completo está compuesto por cuatro componentes: el hardware, el software, los datos y las personas. (Hansen, 2006)

- *Hardware*: se refiere al conjunto de dispositivos físicos de almacenamiento donde radica la BD, así como los periféricos necesarios para su explotación.
- *Software*: incluye el conjunto de aplicaciones de propósito general conocido como SGBD, así como los programas de aplicación e incluso el sistema operativo.
- *Datos*: se refiere a la información en sí almacenada en la BD. Estos datos tienen que estar lógicamente estructurados de acuerdo con las reglas del negocio establecidas.
- *Usuarios*: los usuarios hacen uso de la BD, ya sea para crear los programas de aplicación que hacen uso de la BD (programador de aplicaciones), para consultar los datos de la BD mediante un lenguaje de consulta o programa de aplicación (usuario final), para controlar la BD en sí (administrador de la BD).

Esakkirajan y Sumathi definen además otro componente: *los procedimientos*. Los procedimientos son reglas que gobiernan el diseño y uso de las BD y pueden contener información sobre cómo autenticarse en un SGBD, iniciar y detener el SGBD, proceder sobre en cómo identificar el componente fallido, cómo recuperar la BD, cambiar la estructura de la tabla y mejorar el rendimiento. (Esakkirajan, 2007)

1.2.4 Modelos de Bases de Datos

Durante las décadas del 1960 y 1970 los desarrolladores de aplicaciones crearon sistemas de BD que solucionaban los problemas que tenían los sistemas basados en

ficheros y comenzaron a aparecer distintos modelos de datos para representar la estructura de la información almacenada.

Según Silberschatz un modelo de datos es una colección de herramientas conceptuales para la descripción de datos, relaciones entre datos, semántica de los datos y restricciones de consistencia. (Abraham Silberschatz, 2002)

Rivera define: un modelo de datos es un conjunto de conceptos que nos permiten describir los datos, las relaciones que existen entre ellos, la semántica y las restricciones de consistencia. (Rivera, 2011)

Pantaleón plantea que un modelo de datos es una "colección de herramientas conceptuales que se emplean para especificar datos, las relaciones entre ellos, su semántica asociada y las restricciones de integridad." (Zorrilla, 2009)

Un modelo de datos puede catalogarse como un conjunto de reglas y convenciones bien definidas matemáticamente que describen los datos del mundo real y las operaciones que permiten manipular dichos datos.

Generalmente un modelo de datos presenta dos sub – lenguajes que lo componen: el Data Definition Language[2] (DDL) o Lenguaje de Definición de Datos en español, que describe de manera abstracta la estructura de los datos, y el Data Manipulation Language[3] (DML) o Lenguaje de Manipulación de Datos en español enfocado a la recuperación de los datos, que se le conoce comúnmente como lenguaje de consulta.

Los modelos más usados en la década de 1960 eran el modelo jerárquico basado en árboles y el modelo de red basado en grafos. Posteriormente, a finales de esa década y principios de la década del 70 hicieron su aparición el modelo de datos relacional y el modelo de datos orientado a objetos.

1.2.4.1 Modelo jerárquico

El modelo jerárquico almacena la información en una estructura jerárquica, donde los datos se organizan en forma de árbol. Se consideran los registros de la BD como colecciones de otros y para crear los enlaces entre esos registros el sistema utiliza relaciones padre – hijo mediante el uso de árboles. A diferencia de otros modelos,

[2] El Data Definition Language (DDL) es usado para crear o modificar las tablas y otros objetos de la BD.
[3] El Data Definition Language (DML) es usado para trabajar con los datos almacenados en las tablas.

todas las relaciones están bien jerarquizadas, no permitiendo establecer relaciones entre hijos o entre distintas capas.

El modelo jerárquico describe determinado universo de discurso mediante un árbol en el que los nodos representan las entidades y los arcos que unen los nodos las relaciones entre las entidades.

Dentro de las principales ventajas de este modelo está que la navegación se realiza muy rápidamente y es muy fácil de ver la estructura de datos, pero tiene en detrimento que desaprovecha mucho el espacio, es necesario un conocimiento profundo de las relaciones entre los nodos y las operaciones de adición y eliminación son muy complejas.

De acuerdo con Catalá las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos. (Catalá Mallofré, 2006)

El modelo de datos jerárquico es usado todavía hoy, sobre todo en instituciones públicas para gestionar inventario y contabilidad.

1.2.4.2 Modelo de red

El modelo de datos en red se introdujo a finales de los años 1960 y fue novedoso en que introdujo la idea de punteros dentro de la BD. Estos punteros significaban que un registro podía referenciar a otro, por lo que cada registro además de los datos, contenía la dirección de otro registro. El modelo de red se puede considerar un punto intermedio entre el modelo jerárquico y el relacional, que se describirá más adelante, y su estructura es parecida a la del modelo jerárquico pero más compleja con lo que consigue evitar alguna de sus desventajas.

Según Rosa M. Mato, el término base de datos en red no se refería (al contrario de lo que se entiende actualmente) a que la base de datos estuviera almacenada en una red de ordenadores, sino por la manera en la que los datos se enlazaban con otros datos. Se llama, por tanto, modelo en red porque representa los datos que contiene en la forma de una red de registros y conjuntos (en realidad listas circulares llamadas sets) que se relacionan entre sí, formando una red de enlaces. (García, 2005)

Un modelo en red es una versión menos restrictiva de un modelo jerárquico. Frente a la relación “padre/unidad hijo”, la estructura en red permite que cualquier unidad se encuentre relacionada con cualquier otro conjunto de unidades mediante relaciones fijas. Esta flexibilidad permite una gran variedad de estructuras únicas. Aunque las unidades no forman ninguna estructura específica, se comportan como una red de conexiones.

Este modelo, aunque considerado más flexible que el jerárquico, tiene sin embargo algunas desventajas también puesto que para seguir todos los enlaces de los registros en BD muy grandes consumía mucho tiempo, además de que las aplicaciones tenían que tomar generalmente la responsabilidad de mantener los punteros a medida de que los registros eran agregados o eliminados. Otra desventaja es que no venía acompañado de un lenguaje de consulta de alto nivel por lo que para escribir alguna consulta se requería de gran esfuerzo.

Al igual que el modelo jerárquico, cayó en desuso con el paso de los años, y ambos se consideran como puentes para la creación del modelo relacional.

1.2.4.3 Modelo relacional

La teoría de las BD cambió notablemente en 1970 con la publicación de una serie de informes donde se planteaba que los sistemas de BD deberían presentarse en estructuras llamadas relaciones, definiendo el modelo relacional y las formas no procedimentales de consultar los datos, naciendo las BD relacionales.

Este informe presentado por E. F. Codd, presentaba la idea de que “los datos deberían relacionarse mediante interrelaciones naturales, lógicas, inherentes a los datos, más que mediante punteros físicos” (Codd, 1970).

La simplicidad del modelo relacional y la posibilidad de ocultar completamente los detalles de implementación al programador fueron realmente atractivas (Esakkirajan, 2007). Este modelo se basa en la percepción del mundo real, representando los objetos como una colección de objetos básicos y las relaciones entre estos.

Según Codd:

“Los futuros usuarios de grandes bancos de datos deben estar protegidos de tener que conocer como los datos son organizados en la máquina (la representación interna). […] Las actividades de los usuarios en las terminales y la mayoría de los programas de aplicación deben permanecer sin afectaciones cuando la representación interna de

los datos es cambiada e incluso cuando algunos aspectos de la representación externa son cambiados. Los cambios en la representación de los datos serán frecuentemente necesarios como resultado de cambios en las *querys*[4]*, updates*[5] y reportes de tráfico y el crecimiento natural en los tipos de información almacenada" (Codd, 1970)

Los modelos de datos se expresan en términos de entidades, relaciones y atributos. En la **Fig. 1.6** se muestran los bloques constructivos del modelo relacional.

Las *entidades*, representadas en el modelo relacional por un rectángulo, significan un objeto del mundo real que se distingue de todos los demás objetos. Una entidad es algo sobre lo que el sistema necesita guardar información. (Riordan, 1999)

Los *atributos* son propiedades de las entidades que las describen y en el modelo relacional se representan mediante elipses. Los atributos describen propiedades que posee cada miembro de un conjunto de entidades. La designación de un atributo para un conjunto de entidades expresa que la BD almacena información similar concerniente a cada entidad del conjunto de entidades; sin embargo, cada entidad puede tener su propio valor para cada atributo. (Esakkirajan, 2007)

Las *relaciones* representan asociaciones entre distintas entidades y se representa mediante un rombo. La mayoría de las relaciones son binarias, o sea, entre 2 entidades, pero se pueden encontrar también relaciones ternarias y reflexivas.

Según Quiroz la estructura fundamental del modelo relacional es la relación, es decir una tabla bidimensional constituida por filas (tuplas) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la BD. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representan las propiedades de la entidad. (Quiroz, 2003)

Matthews y Stones plantean que el modelo relacional enfatiza en la integridad de los datos mucho más que cualquiera de los modelos anteriores. La integridad referencial se refiere a asegurar que los datos en la base de datos tengan siempre sentido. (Matthew, y otros, 2005)

[4]Consultas a la BD
[5]Actualizaciones de la BD

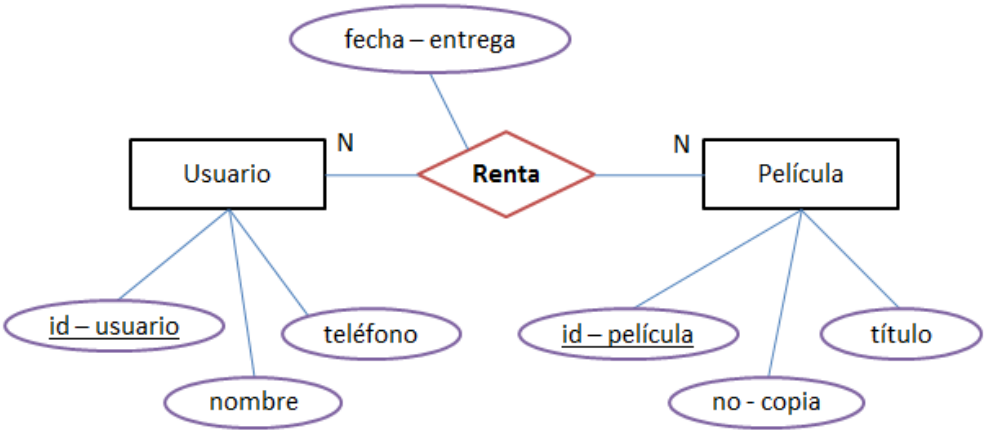


Fig. 1.6 Modelo relacional.

1.2.4.4 Bases de Datos Orientadas a Objetos

La programación orientada a objetos ha llegado hasta los sistemas de gestión que utilizan BD. Aquí surge el conflicto de conectar la orientación a objetos con las consultas y acceso a los datos que se realizan en la BD; además, los datos devueltos por las consultas de la BD son retornados en un formato incomprensible para el lenguaje orientado a objeto. Las clases utilizadas en un determinado lenguaje de programación orientado a objetos son las mismas clases que serán utilizadas en una Base de Datos Orientada a Objetos (BDOO); de tal manera, que no es necesaria una transformación del modelo de objetos para ser utilizado por un Sistema Gestor de Base de Datos Orientada a Objeto (SGBDOO). De forma contraria, el modelo relacional requiere abstraerse lo suficiente como para adaptar los objetos del mundo real a tablas.

Surge entonces el modelo orientado a objetos en las BD, de reciente creación, que define la BD en términos de objetos, propiedades y operaciones. Los objetos que tienen la misma estructura pertenecen a la misma clase que se organiza en una jerarquía de clases.

Las BDOO nacen para evitar los problemas que surgen al tratar de representar cierta información, aprovechar las ventajas del paradigma orientado a objetos en el campo de las bases de datos y para evitar transformaciones entre modelos de datos. (Alberca Manzaneque, y otros, 2008)

Una BDOO es una BD que incorpora en su diseño, conceptos específicos de la orientación a objetos como son:

1. Encapsulación: propiedad que permite la ocultación de la información al resto de los objetos.
2. Herencia: propiedad mediante la cual los objetos heredan propiedades y comportamientos siguiendo una jerarquía de clases.
3. Polimorfismo: propiedad por la cual un mensaje puede tener distintos comportamientos de acuerdo al objeto.

En este modelo la información sobre una entidad se almacena como un objeto persistente y no como una fila en una tabla. Esto, en principio, lo hace más eficiente en términos de requerimientos de espacio y asegura que los usuarios puedan manipular los datos sólo de las maneras en las que el programador haya especificado. También es más eficiente en el uso de espacio de disco requerido para las consultas, ya que en vez de almacenar la consulta, simplemente se construye una serie de índices (punteros) a los objetos seleccionados. A esto hay que sumar las ventajas derivadas del modelo orientado a objetos, ya explotadas en sus lenguajes de programación, la mayor expresividad y su adecuación para almacenar muchos tipos de datos diferentes. (García, 2005)

1.2.5 Importancia del empleo de Bases de Datos

La utilización de las BD como plataforma para el desarrollo de aplicaciones ha crecido en los últimos años. El empleo de una BD en un proyecto o empresa significa el empleo de tecnologías y personal capacitado para manejarlas, así como el gasto de capitales por concepto de pago de licencias y compra de hardware apropiado para la instalación de la BD. Además la implantación del sistema de BD es difícil, lo que no brinda beneficios palpables a corto plazo.

Siendo así: ¿cuáles serían las ventajas que reportaría el uso de una BD para solucionar los problemas de almacenamiento de información?

De acuerdo con Date, dentro de las ventajas que tiene el empleo de una BD sobre el almacenamiento tradicional de la información están: (Date, 2001)

- Compactación: No hay necesidad de archivos en papeles voluminosos.
- Velocidad: La máquina puede recuperar y actualizar datos más rápidamente que un ser humano.
- Menos trabajo laborioso: Las tareas mecánicas siempre las realizan mejor las máquinas.

- Actualidad: En el momento que la necesitemos, tendremos a nuestra disposición información precisa y actualizada.

Con respecto a los sistemas basados en ficheros se puede mencionar que las BD las aventajan en:

- Reducción de la redundancia: En los sistemas de ficheros, las aplicaciones tienen sus datos privados, lo que provoca alta redundancia y se desaprovecha el espacio en disco.
- Compartimentación de los datos: Las nuevas aplicaciones que necesiten acceder a los datos ya existentes, no tienen que almacenar nuevos datos.
- Restricción de la seguridad: Solamente los usuarios autorizados pueden acceder a los datos y realizar operaciones sobre ellos.
- Recuperación ante fallas: Los sistemas de BD presentan técnicas para recuperarse ante cualquier falla técnica.
- Integridad de los datos: La integridad de los datos consiste en garantizar que los datos almacenados no entren en contradicción entre ellos mismos.

Como consecuencia de las bondades que brindan los sistemas de BD, su uso ha aumentado considerablemente en las últimas décadas, surgiendo además nuevas demandas, técnicas y herramientas para darle solución a estas necesidades cada vez más creciente de los usuarios. Uno de los fenómenos que más ha influenciado en el desarrollo de las BD ha sido el crecimiento de la Internet. El enlace de la BD con la web se ha ido desarrollando de manera tal que anteriormente se utilizaban herramientas específicas para la realización de esa tarea y actualmente los SGBD proporcionan módulos definidos solamente para eso.

De este modo los sistemas de BD se han transformado en el cimiento sobre los que se basan los sistemas de información, en especial los sistemas de gestión.

1.3 Sistemas Gestores de Bases de Datos

Para poder realizar un buen trabajo de gestión de datos es necesario contar con las herramientas que faciliten y optimicen esta tarea y sirvan como una capa intermedia entre los datos almacenados en la BD y los usuarios. Es aquí donde se introducen los Sistemas Gestores de Bases de Datos (SGBD) o *Data Base Management System* (DBMS) en inglés.

Según Bachmann el primer SGBD fue el *Integrated Data Store* (IDS), creado en los primeros años de la década de 1960 por él. Este se basaba en el modelo de datos en red que fue ampliamente usado por la industria (Bachmann, 2009). Posteriormente surge el *Information Management System* (IMS) bajo el concepto del modelo de datos jerárquico. Con el advenimiento del modelo de datos relacional y posteriormente del modelo de datos orientado a objetos, se fueron creando cada vez más potentes SGBD hasta los que se encuentran actualmente. Estos SGBD se accedían mediante lenguajes de programación como Cobol usando interfaces de bajo nivel haciendo que las tareas de creación de aplicaciones y mantenimiento de los datos fuesen controlables, pero aún complejas.

Existen varios conceptos de SGBD:

- EL Dr. Rogelio Silverio Castro plantea que un SGBD es “un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y los sistemas que la utilizan. Sus funciones principales son la creación, mantenimiento y eliminación de bases de datos, el control de accesos, la manipulación de información de acuerdo a las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad” (Silverio Castro, 2005).
- Según Silberschatz “un sistema gestor de bases de datos consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente” (Abraham Silberschatz, 2002).
- El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos. (García, 2005)

En resumen, un SGBD es un sistema computacional de administración de BD que controla el almacenamiento, organización, recuperación, integridad y seguridad de los datos en una BD y maneja todas las solicitudes de acceso a la BD.

Los SGBD se dicen transaccionales si son capaces de mantener la integridad de los datos al realizar transacciones[6] siempre que estos no queden en estados no definidos. Para garantizar la integridad de estos datos se necesita que el SGBD mantenga las propiedades ACID (Atomicity, Consistency, Isolation y Durability) para las transacciones:

- *Atomicidad*: Las operaciones de la transacción se realizan adecuadamente todas o no se realiza ninguna de ellas, lo que significa que la transacción no puede ser subdividida
- *Consistencia*: Esta propiedad garantiza que solo se puedan realizar aquellas transacciones que no rompan las reglas de integridad de la BD.
- *Aislamiento*: En el SGBD distintas transacciones son ejecutadas concurrentemente. Estas transacciones deben realizarse aisladas cada una de la otra y no deben afectarse mutuamente asegurando que las transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- *Durabilidad*: Esta propiedad garantiza que una vez que una transacción es terminada, el resultado de sus operaciones sobreviven la BD aun cuando exista una falla posterior.

1.3.1 Arquitectura de los SGBD

La mayoría de los programas de aplicación son dependientes de los datos, en el sentido de que no se puede cambiar la estructura interna de los datos ni las maneras de acceder a ellos sin afectar la aplicación. Dentro de las características propias de los SGBD está precisamente la independencia de los datos de los programas de aplicación, así como el uso de un catálogo para almacenar el esquema de la BD y el manejo de múltiples vistas por los usuarios.

En 1975 el comité *American National Standard Institute – Standards Planning and Requirements Committee* (ANSI/SPARC) propone una arquitectura de 3 capas que resulta muy útil para la consecución de estas características, en especial de la independencia de los datos. Esta arquitectura es empleada actualmente por la

[6] Una transacción es una colección de operaciones que se ejecutan en una unidad lógica de tiempo de un programa que accede y posiblemente actualiza varios datos. Según Gray “una transacción es una transformación de estado que tiene las propiedades de atomicidad, durabilidad y consistencia” (Gray, 1981)

mayoría de los SGBD comerciales aunque no todos los sistemas la implementan. Según este comité: “La independencia de los datos es la capacidad de un sistema para permitir que a las referencias a los datos almacenados, especialmente en los programas y en sus descriptores de los datos, estén aislados de los cambios y de los diferentes usos en el entorno de los datos, como pueden ser la forma de almacenar dichos datos, el modo de compartirlos con otros programas y como se reorganizan para mejorar el rendimiento del sistema de Bases de Datos”. (ANSI/SPARC, 1975)

De acuerdo con Esakkirajan y Sumathi: “la independencia de los datos significa que la estructura interna de la BD debe permanecer sin afectaciones debido a cambios en aspectos del almacenamiento físico. Debido a la independencia de los datos, el administrador de la BD puede cambiar las estructuras de almacenamiento de los datos sin afectar la vista de los usuarios” (Esakkirajan, 2007).

La arquitectura *ANSI/SPARC* está compuesta por tres niveles, como se muestra en la **Fig. 1.7**:

- El nivel externo o de vista se ubica primero entre los usuarios finales de la BD y las demás capas de abstracción. Este nivel describe varios esquemas externos o *vistas* de usuario. Cada vista está hecha según los requerimientos y privilegios de cada usuario y limita lo que un usuario puede percibir de la BD.
- El nivel lógico o conceptual es el nivel medio de abstracción. Se define por el Esquema Conceptual, en el que se describen las entidades con sus atributos y las relaciones entre ellas, ocultando los detalles de las estructuras de almacenamiento y definiendo las restricciones de integridad y confidencialidad.
- El nivel físico o interno es el responsable del almacenamiento físico de la información, describiendo en detalles las complejas estructuras de datos de bajo nivel: direcciones físicas, relaciones, índices, los archivos de configuración, etc. De este nivel solo tiene conocimiento el diseñador y administrador de la BD.

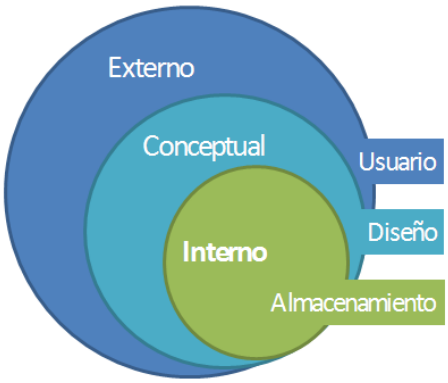


Fig. 1.7 Arquitectura en 3 niveles.

Hay que destacar que los tres esquemas no son más que descripciones de los mismos datos pero con distintos niveles de abstracción.

1.3.2 Componentes de los SGBD

Los SGBD están formados por una serie de componentes que le permiten la realización de cada una de sus funciones:

- Procesador DDL. Convierte las definiciones de datos (esquemas externos) en su forma objeto correspondiente y las almacena en el diccionario de datos.
- Procesador DML. Se encarga de manejar las peticiones para la manipulación de los datos.
- Procesador del lenguaje de consultas. Se encarga de manejar las consultas de alto nivel ingresadas por el usuario.
- Optimizador de consultas. Define las operaciones de los usuarios de manera que estas puedan ser ejecutadas de manera más eficiente.
- Procesador en tiempo real. Maneja el acceso a la BD en tiempo real. Recibe las operaciones y las lleva a la BD.
- Manejador de almacenamiento. Es responsable de almacenar, obtener y actualizar los datos. Está compuesto por:
    - Administrador de autorización e integridad. Chequea las constantes de integridad y las autorizaciones de los usuarios para acceder a los datos.
    - Administrador de transacciones. Su función es la administración de las peticiones de manipulación de la BD y que el acceso concurrente a los datos no resulten en conflicto.

- Administrador de archivos. Se encarga de la localización de los archivos en disco y se encarga de definir y mantener las estructuras e índices[7] definidos en el esquema interno.
- Administrador de Buffer. Se encarga de recuperar los datos desde el disco hacia la memoria principal.

- Metadatos o diccionario de datos. Contiene el esquema de la BD, los usuarios y sus permisos. Son datos sobre los datos que permiten la traducción a través de los distintos niveles de la arquitectura ANSI/SPARC.

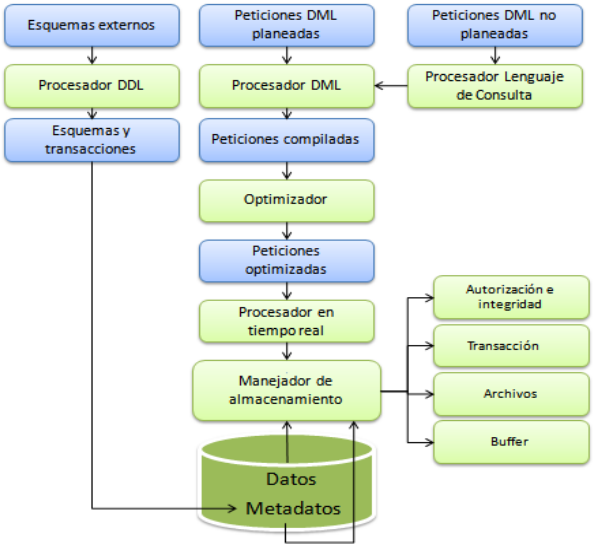


Fig. 1.8 Componentes de los SGBD.

1.3.3 Clasificación

Generalmente los SGBD se clasifican atendiendo al modelo lógico en el que se basan. Atendiendo a los modelos lógicos, actualmente se pueden clasificar en 2 tipos de modelo lógicos empleados con mayor frecuencia en los SGBD comerciales:

- SGBDR. Para los SGBD basados en el modelo relacional.
- SGBDOO. Para los SGBD basados en el modelo orientado a objetos.

Además los SGBD pueden ser clasificados como pasivos y activos. (Esakkirajan, 2007)

[7] Los índices son una lista de valores numéricos que dan el orden de los registros cuando son ordenados por un campo específico o columna de una tabla. (Esakkirajan, 2007)

- SGBD pasivos: los SGBD pasivos son guiados por programas. Los SGBD tradicionales son pasivos en el sentido que son explícita y sincrónicamente invocados por el usuario o programa de aplicación. Las aplicaciones mandan solicitudes de operaciones para ser realizadas por el SGBD y esperan por que este le devuelva algún posible resultado. Las operaciones pueden ser actualizaciones y definiciones de esquemas así como consultas y actualizaciones de datos.
- SGBD activos: los SGBD activos son guiados por datos o eventos. En un SGBD activo el usuario especifica la información necesaria. Si esta información está disponible el SGBD monitorea la llegada de esta información y se la entrega al usuario. El alcance de las consultas en los SGBD pasivos se limita a los datos pasados y presentes mientras que en un SGBD activo incluye además los datos futuros. Un SGBD activo revierte el flujo de control entre la aplicación y el SGBD, en vez de solamente la aplicación invocar al SGBD, este puede invocar a la aplicación también.

1.3.4 Objetivos y Funciones

Los principales objetivos de los SGBD son: (Esakkirajan, 2007)

- Disponibilidad de los datos. Los datos están disponibles para una variedad de usuarios en un formato significativo a un costo razonable de manera que los usuarios puedan acceder fácilmente a estos datos.
- Integridad de los datos. Se refiere a la corrección de los datos en la BD, o sea que los datos sean fiables.
- Seguridad de los datos. Los datos puedan ser accedidos solamente por los usuarios autorizados. La seguridad puede ser reforzada mediante contraseñas.
- Independencia de los datos. Los SGBD deben esconder ciertos detalles de cómo los datos son almacenados, proveyendo una vista abstracta de las complejas estructuras que representan los datos.

Dentro de las funciones de las que se encargan los SGBD están principalmente:

- Definir la BD mediante el DDL, el cual permite definir las estructuras de datos así como las restricciones a estos.
- Permitir la consulta, inserción y eliminación de los datos mediante el DML.
- Permitir la realización de copias de seguridad de la BD, así como la recuperación de estas ante cualquier fallo del sistema.

- Minimizar o eliminar la redundancia de datos y así reducir espacio en disco.
- Permitir una fácil administración de los datos.
- Controlar la concurrencia.
- Minimizar el tiempo de respuesta para hallar o actualizar los datos.

1.3.5 Ventajas

La utilización de una tecnología está justificada por las ventajas que esta brinda. Los SGBD tienen varias ventajas como son:

- Administración centralizada de datos.
- Integridad del sistema.
- Reutilización de datos y programas.
- Control sobre la redundancia de datos y consistencia de estos.
- Compartición de datos.
- Aumento en la seguridad, integridad y accesibilidad de los datos.
- Las copias de seguridad y la recuperación ante fallos son mejoradas.
- Establecimiento de prioridades de requerimientos.

1.3.6 Desventajas

No obstante el conjunto de ventajas que presentan los SGBD tienen también una serie de inconvenientes que deben ser tenidos en cuenta. Estos son:

- La complejidad inherente del sistema.
- El coste económico del SGBD.
- Coste del hardware necesario para soportarlo.
- Vulnerabilidad a fallos.

1.4 SGBD actuales

Actualmente puede observarse el cambio en los SGBD. Inicialmente se trataba de un software muy caro, en ordenadores muy costosos. En la actualidad existen SGBD para ordenadores personales, algunos de ellos muy económicos o incluso gratuitos. Este abaratamiento y reducción del tamaño físico difiere con la mayor capacidad y potencia de los SGBD. La mayoría de los SGBD actuales son relacionales y el mercado se encuentra controlado por tres compañías: Oracle, IBM y Microsoft.

1.4.1 Microsoft SQL Server

Microsoft SQL Server es un SGBD creado por Microsoft Corporation basado en el modelo relacional y bajo la licencia privativa EULA[8]. SQL Server constituye la alternativa creada por Microsoft para enfrentar a otros potentes SGBD como son Oracle, MySQL o PostgreSQL. SQL Server abarca tanto el área de diseño como la de administración. Es capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización.

Dentro de sus características se pueden nombrar:

- Seguridad y estabilidad.
- Arquitectura de almacenamiento en disco, permite la escalabilidad desde BD de equipos portátiles hasta BD empresariales de tamaño de terabyte.
- Soporta procedimientos almacenados.
- Presenta un potente entorno gráfico de administración.
- Soporte de transacciones.
- Facilidad de uso.

Según Microsoft en su sitio web: “Microsoft SQL Server es un servidor de base de datos comprensivo y una plataforma de información que ofrece un conjunto de tecnologías para la empresa y herramientas que permiten obtener lo más valioso de la información al menor costo. Disfrute de altos niveles de rendimiento, disponibilidad y seguridad; emplee más herramientas de administración y desarrollo […]” (Microsoft Corporation, 2012)

Cabe destacar como desventajas de este SGBD que no maneja la compresión de sus datos por lo que las BD pueden llegar a ocupar mucho espacio en disco y que solo puede instalarse en servidores con sistema operativo Microsoft Windows.

1.4.2 Oracle

Oracle es una herramienta cliente/servidor para la gestión de BD objetos-relacional creado por Oracle Corporation y considerado como uno de los SGBD más completos existentes en la actualidad. Es un SGBD muy potente que brinda muchas posibilidades

[8]End User License Agreement (EULA) es una licencia que solamente permite al usuario final de cierta aplicación o software, la explotación y empleo de la misma.

pero debido a su elevado precio de su licencia y del soporte técnico es mayormente usado solo por grandes empresas. Dentro de las funcionalidades que nos aporta este SGBD destacan el soporte y tratamiento de grandes cantidades de datos y permite trabajar en un entorno multi-usuario, distribuido y portable.

Ventajas:

- Estabilidad.
- Escalabilidad.
- Soporte de transacciones, cumple con las normas ACID.
- Es multiplataforma.
- Soporta la mayoría de los lenguajes de programación.

Desventajas:

- Su mayor inconveniente es su alto precio.
- Ha sido criticado por algunos especialistas por problemas de seguridad de la plataforma, y las políticas de suministro de parches de seguridad.

Sus últimas versiones han sido certificadas para poder trabajar bajo GNU/Linux debido a la gran competencia que tiene hoy con los demás SGBD de software libre. Oracle es sin duda alguna uno de los SGBD más potentes aunque sus altos precios no lo hacen muy accesible a todos.

1.4.3 MySQL

MySQL es de los SGBD de software libre más populares en la actualidad, licenciado bajo *GNU GPL*[9], actualmente subsidiada por la corporación Oracle. A diferencia de otros proyectos como Apache, donde el encargado de desarrollar software es la comunidad, MySQL está patrocinada por una empresa privada, la que brinda soporte y servicios.

MySQL proporciona un servidor de BD muy rápido, multi-hilo, multi-usuario, multiplataforma y robusto. El servidor MySQL se diseñó para entornos de producción críticos, con abundante carga de trabajo y para integrarse en software distribuido. Dentro de sus características están además:

- Cumple con las normas ACID para las transacciones.

[9]GPL es una licencia creada por la Fundación de Software Libre, orientada a la protección de la libre distribución, modificación y uso de software.

- Replicación.
- Conexión segura.
- Transacciones y llaves foráneas.
- Triggers.
- Consume pocos recursos.

Según las propias palabras del fabricante: “MySQL se ejecuta en más de 20 plataformas, incluyendo Linux, Windows, Mac OS, Solaris, IBM AIX, brindando el tipo de flexibilidad que te pone en control. Sin importar si eres nuevo en tecnología de bases de datos o un desarrollador experimentado o administrador de bases de datos, MySQL ofrece una variedad de herramientas para la base de datos, soporte, entrenamiento y servicios de consultas para su éxito.” (Oracle Corporation, 2012)

1.4.4 PostgreSQL

PostgreSQL se puede considerar actualmente como uno de los SGBD objeto-relacional de código abierto más avanzado. Publicado bajo la licencia BSD [10] y desarrollado en la Universidad de California por el Departamento de Ciencias de la Computación Berkeley, lleva más de 15 años de desarrollo y consta de una probada arquitectura basada en la fiabilidad e integridad de los datos.

PostgreSQL es un SGBD objeto-relacional que utiliza un modelo cliente/servidor y usa multiprocesos en vez de multi-hilos para garantizar la estabilidad del sistema. Es multiplataforma pudiendo ejecutarse en distintos sistemas operativos como Linux, Mac OS X, Windows, etc.

PostgreSQL puede ser comparado favorablemente con otros SGBD. Contiene casi todas las características que se pueden encontrar en otras bases de datos comerciales u open-source, y algunas extras que no se pueden encontrar en más ningún otro. (Matthew, y otros, 2005)

Dentro de las características de PostgreSQL se pueden encontrar:

- Cumple las normas ACID para las transacciones.
- Multi-Version Concurrency Control (MVCC). Esta tecnología permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

[10]La licencia Berkeley Software Distribution (BSD) es una licencia de software libre que tiene menos restricciones que la GPL, sí permitiendo el uso de código en software propietario.

- Soporte para llaves foráneas, vistas, sub-consultas y procedimientos almacenados, integridad referencial.
- Herencia de tablas.
- Gran variedad de tipos nativos (Arrays, direcciones IPv4 e IPv6, direcciones MAC, texto de largo ilimitado).
- Soporta almacenamiento de grandes objetos binarios como sonido, imágenes y video.

Los usuarios pueden además extenderlo añadiendo:

- Nuevos tipos de datos.
- Funciones y operadores.
- Funciones agregadas.
- Lenguajes procedurales.

En el sitio web del fabricante se plantea: “Lo mejor de todo es que el código fuente PostgreSQL está disponible bajo la licencia open source: la licencia PostgreSQL. Esta licencia permite la libertad para usar, modificar y distribuir PostgreSQL de cualquier manera que se desee, código abierto o propietario. Cualquier modificación, mejora o cambios que realices son suyos para hacer como desees. Por consiguiente, PostgreSQL no es solamente una bases de datos poderosa capaz de ejecutarse por empresas, es una plataforma de desarrollo en la que se puede desarrollar productos comerciales, web y domésticos que requieran de un SGBD objeto-relacional capaz.” (PostgreSQL Global Development Group, 2012)

1.4.5 SGBD empleado

Luego de realizar un estudio de los principales SGBD utilizados en la actualidad, se escogió como manejador de la BD a PostgreSQL en su versión 9.1.3. La razón principal de porque se escogió este gestor es que su licencia es de software libre por lo que se puede hacer uso de él y comercializar el producto sin necesidad de pagar licencias, como sí hay que hacerlo con Oracle y SQL Server. No es solo eso, sino que es uno de los SGBD más potente que existe en la actualidad, presenta características muy atractivas como la orientación a objetos y excelentes prestaciones así como una excelente seguridad. Es además multiplataforma y consume pocos recursos.

1.5 Metodología del diseño de una Base de Datos

El diseño de BD es un proceso complejo por lo que se necesitan tomar ciertas decisiones. Para controlar la complejidad del diseño se separa el problema en otros más sencillos, resolviéndose cada uno de estos de manera independiente. Así el diseño de una BD se descompone en el diseño conceptual, lógico y físico.

Según Pantaleón las fases del diseño de la BD son: (Zorrilla, 2003)

- *Fase inicial*: Análisis de requisitos. Descripción de la información a gestionar y sus procesos. Entrevistas con usuarios y expertos.
- *Diseño conceptual*: Traducción del análisis de requisitos al esquema conceptual. Representación generalmente gráfica de las entidades y sus relaciones.
- *Implantación en el gestor:*
    - Diseño lógico: traducción del modelo conceptual al DDL del gestor correspondiente.
    - Diseño físico: determina la organización de archivos y las estructuras de almacenamiento interno.

Durante la fase inicial se analizan los requisitos que debe cumplir la BD. En esta fase describe la información que se requiere gestionar y los procesos que deberá realizar el sistema. Como resultado de esta fase se obtiene una especificación de la información que se va a almacenar, también se obtienen los requisitos funcionales que son las especificaciones de las operaciones que se realizan con los datos.

Durante el diseño conceptual se traducen los requisitos anteriormente identificados y se traducen al esquema conceptual, obteniéndose una representación gráfica de las entidades que conforman la BD. Es en esta fase donde se realiza el modelo entidad-relación. Este diseño es completamente independiente de los aspectos de implementación, como puede ser el SGBD, los lenguajes de programación, el hardware o cualquier otra consideración física.

Durante el diseño lógico se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de la BD en el que se basa el SGBD utilizado, como puede ser el modelo relacional u objeto – relacional. La normalización es una técnica utilizada para comprobar la validez del esquema lógico, ya que asegura que las relaciones obtenidas no tengan datos redundantes.

El diseño conceptual y el lógico son etapas clave para conseguir que un sistema funcione correctamente. Si estos no son una representación fiel del negocio, es casi imposible definir todas las vistas de usuario y mantener la integridad de la BD.

El diseño físico utiliza el esquema lógico y da como resultado un esquema físico. Este esquema es una descripción de la implementación de una BD en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para el acceso a los datos. El diseño físico depende del SGBD utilizado y se expresa mediante su DDL. Entre el diseño físico y el lógico debe existir una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico, pueden afectar la estructura del esquema lógico.

1.6 Patrones de diseño de BD.

Un patrón es una plantilla que ya ha sido evaluada como la responsable de resolver un problema, es una guía para apoyarse en realizar el trabajo. Los patrones de diseño de bases de datos permiten crear una base de datos más fortalecida, ofreciendo una guía que especifica cómo debe ser la base de datos. El diseño y construcción de una bases de datos requiere del mayor esfuerzo y análisis posible ya que a partir de este diseño es que se crea la bases de datos, en la actualidad las bases de datos suelen ser muy grandes y el trabajo con los patrones de diseño hace que el trabajo sea más fácil, además asegura un resultado correcto (Blaha, 2010).

Dentro de los patrones de diseño más conocidos se encuentran:

- Árbol fuertemente codificado.
- Árbol simple
- Árbol cambiante
- Máquina de estado
- Entidad – Atributo – Valor
- Llaves Subrogadas

Los patrones empleados en el diseño de la BD del sistema SIRECC serán explicados con detenimiento en el Capítulo 2.

1.7 Diseñador de Bases de Datos

El diseñador de la BD es el encargado del diseño lógico y físico de la BD. Este se encarga de identificar los datos, las relaciones entre ellos y sus restricciones, además de decidir cómo será instrumentada a nivel físico la BD.

El diseñador de BD es responsable del mapeo del modelo lógico de datos en un conjunto de tablas y restricciones de integridad, seleccionar la estructura específica de almacenamiento y diseñar las medidas de seguridad requerida para los datos. (Esakkirajan, 2007)

1.8 Herramientas y lenguajes de modelado

El desarrollo de la informática ha puesto al alcance de todos, herramientas que facilitan el desarrollo de productos de software. Estas contribuyen a controlar con mayor efectividad los procesos permitiendo realizar actividades de manera rápida. Para la realización del diseño e implementación de una BD se utilizan herramientas que permiten modelar correcta y rápidamente la BD. Entre las distintas herramientas y lenguajes que se pueden utilizar se encuentran:

1.8.1 UML

Como lenguaje de modelado se escogió Unified Modelling Language (UML) en su versión 2.0 ó Leguaje Unificado de Modelado como se le conoce en español. UML es un lenguaje que permite visualizar, construir y documentar los artefactos de un sistema de software. UML además es gratuito y por consiguiente accesible a todos. Conforma la colección de las mejores técnicas de ingeniería que han probado ser un éxito en el modelado de sistemas grandes y complejos.

UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. (Larman, 2004)

Aunque no garantiza el éxito de los proyectos, sí mejora sustancialmente el desarrollo de los mismos, permitiendo una la integración entre las herramientas y los procesos.

1.8.2 Visual Paradigm

Visual Paradigm para UML es una herramienta ampliamente utilizada en el mundo del software que permite a los profesionales modelar sus diseños. Posibilita la integración de aplicaciones empresariales a las bases de datos. Es capaz de generar código e ingeniería inversa para lenguajes como el PHP. Permite manejar grandes estructuras de manera eficiente, solo requiere una configuración de escritorio común. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegues. (Visual Paradigm, 2012)

Entre sus principales características se pueden encontrar:

- Fácil de instalar y actualizar.
- Soporte para aplicaciones web.
- Multiplataforma.
- Genera código para Java y exportación como HTML.
- Permite la generación de BD, conversión de diagramas entidad-relación a tablas de BD, mapeos de objetos y relaciones e ingeniería inversa desde SGBD.

1.8.3 ER/Studio

ER/Studio es una herramienta de modelado de datos fácil de usar basada en UML, utilizada para diseñar y construir BD a nivel físico y lógico. ER/Studio consta con las herramientas necesarias para crear y manejar diseños confiables de BD.

Las capacidades de diseño que contiene, permiten crear un diseño lógico de la BD que posteriormente puede transformarse en diseños físicos. Como resultado, se puede mantener un diseño lógico normalizado mientras no se normalizan los diseños físicos. Esta herramienta mejora la productividad entre los desarrolladores cuando los diseños de la BD son divididos, compartidos, y reutilizados.

Dentro de las funcionalidades de ER/Studio están:

- Sincronización bidireccional de los diseños lógico y físico.
- Es un ambiente de diseño conducido por modelos.
- Construcción automática de BD.
- Reingeniería inversa de BD.
- Documentación basada en HTML.

- Genera automáticamente tablas y líneas de procedimientos almacenados y disparadores.
- Un Repositorio para el modelado.

E/R Studio permite la creación sencilla del diseño de la BD. Los diseñadores de la BD pueden crear gráficamente el modelo entidad – relación con todos los requerimientos de datos y reglas de negocio, mostrando todas las entidades, atributos, relaciones y llaves.

1.8.4 Rational Rose

Rational Rose es una herramienta para el modelado visual de software. Esta permite visualizar, entender, y clarificar los requerimientos y arquitectura de un software antes de enfrentarse al código. Esto evita desperdiciar tiempo y esfuerzos en el ciclo de desarrollo.

Esta herramienta posibilita la identificación de objetos y la construcción de un modelo de casos de usos, así como representar las interacciones con los diagramas de secuencia y colaboración. Puede además organizar los componentes de software y su despliegue, describir la estructura de las clases y generar el código fuente de las clases definidas en el diseño.

Sin embargo una vez generado el diagrama de clases persistentes a partir del cual se genera la BD del sistema, no existe la posibilidad de que el mismo exporte ese modelo hacia algún SGBD por lo que no resulta muy provechoso su empleo.

1.8.5 Herramienta de modelado empleada

Partiendo del análisis que se hizo de las principales herramientas de modelado, se decidió utilizar a E/R Studio en su versión 8.0 para modelar la BD del proyecto SIRECC. Esta cuenta con una serie de características que hacen factible su empleo por parte de los diseñadores, los que ya cuentan con cierta experiencia trabajando con esta herramienta, además de contar con un repositorio que es de gran importancia para el trabajo en equipo y generar código para distintos SGBD entre ellos PostgreSQL.

1.9 Conclusiones parciales

En este capítulo se analizaron las tecnologías más modernas que se pueden utilizar para el desarrollo de la investigación, así como conceptos y tendencias actuales que permitirán dar cumplimiento a la propuesta de solución. También se estudiaron las BD y su estado actual enfatizando en la importancia del uso del SGBD PostgreSQL, dadas las ventajas que reporta el uso del mismo, que además es de código libre. Se estudiaron otras metodologías y herramientas que permitirán la generación de un modelo de la BD eficiente como las herramientas de modelado, en especial E/R Studio que fue la utilizada por las ventajas que aporta.

Capítulo 2: Propuesta de solución

En este capítulo se describe la solución propuesta para el correcto almacenamiento de los datos del sistema SIRECC. Se describe además la arquitectura de datos y los requisitos funcionales, se determinan las características que debe cumplir la BD para su construcción obteniéndose el modelo lógico y físico de la BD, presentando el diagrama Entidad – Relación, describiendo las clases y tablas principales obtenidas en ambos modelos.

2.1 Descripción de la arquitectura de datos

La Cámara de Comercio cuenta con la siguiente Infraestructura Tecnológica aplicable a la solución para el sistema SIRECC:

- 1 servidor HP Proliant ML 1150 con distintas máquinas virtuales entre ellas:
  - 1 máquina virtual para el servidor de aplicaciones.
  - 1 máquina virtual para el servidor de BD.
- 1 servidor NFS HP Proliant ML 1150 para el almacenamiento de la información, mapeado a cada máquina virtual. Este servidor tiene configurado discos duros en RAID 1[11] para asegurar el respaldo de la información.
- Red local a 100 Mb/s.
- Estaciones de trabajo.

Cada máquina virtual tiene las siguientes características:

- Sistema Operativo Ubuntu 10.0.4.
- 512 Mb de memoria RAM.
- 8 Gb de almacenamiento en disco duro.

La máquina virtual para el servidor de BD constará con sistema operativo Ubuntu 10.0.4 y como SGBD PostgreSQL 9.1.3. La BD tendrá nombre sirecc y tendrá un solo esquema del mismo nombre.

[11]Un conjunto de discos duros en disposición RAID 1, crea una copia exacta del conjunto de datos en dos o más discos duros. Esto aumenta la fiabilidad de los datos almacenados en el conjunto pues la probabilidad de fallo total es igual al producto de las probabilidades de fallo de cada uno de los discos.

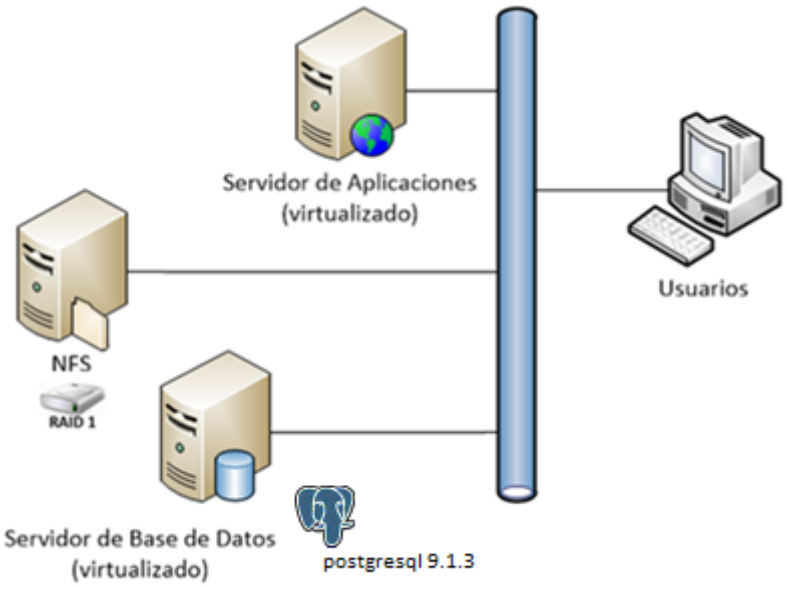


Fig. 2.1 Descripción gráfica de la arquitectura de datos.

2.1.1 Descripción del entorno de desarrollo

El entorno de desarrollo debe construirse de manera similar a la Infraestructura Tecnológica presente en el ambiente donde se desplegará el producto para evitar de esta manera incompatibilidades y ser consecuente con la arquitectura de datos y con la tecnología que se empleará.

El equipo de desarrollo contará con:

- 1 servidor para la virtualización del servidor de aplicaciones, BD y almacenamiento compartido.
- 1 servidor Subversion para el repositorio del proyecto.
- Red local 100 Mb/s.

El servidor para la virtualización tendrá estas características:

- Sistema Operativo Ubuntu 10.0.4.
- 2 Gb memoria RAM.
- 250 Gb de espacio en disco duro.

Se crearán 3 máquinas virtuales. Una para el servidor de aplicaciones, otra para el servidor de BD y una para el servidor NFS. Cada una de estas virtualizaciones tendrá:

- Sistema Operativo Ubuntu 10.0.4.
- 512 Mb de memoria RAM.

- 8 Gb de espacio en disco duro.

El software empleado para los servidores será el mismo que el que se definió en la arquitectura de datos. Los desarrolladores de la BD trabajarán con sistema operativo Windows, con la herramienta de modelado ER – Studio 8.0 y con la herramienta pgAdmin III para la administración de la BD.

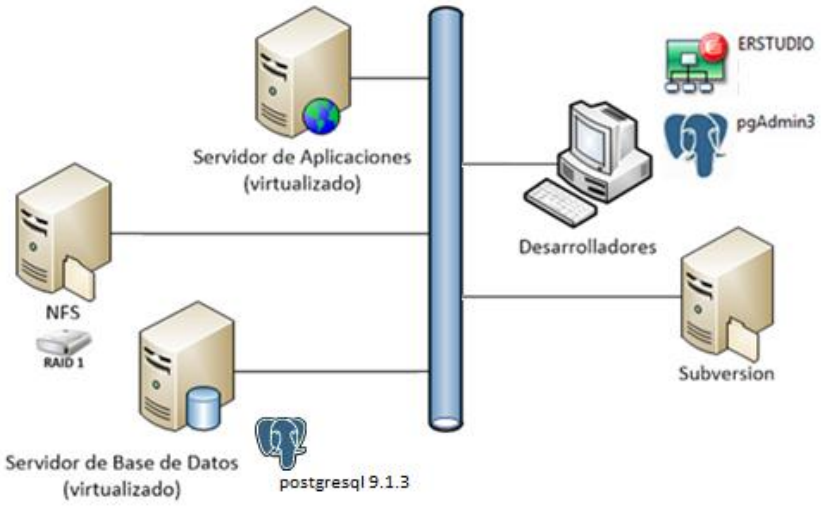


Fig. 2.2 Descripción gráfica del entorno de desarrollo.

2.2 Estándares de nomenclatura

La creación y nombramiento de las tablas se realizará de la siguiente manera:

- El nombre de las tablas nomencladoras comenzará con la letra `n`. Ejemplo: `nEstructura`. (Véase **Fig. 2.3**).
- El nombre de las tablas de datos comenzará con la letra `d`. Ejemplo: `dDocumento`. Las especializaciones se nombrarán: nombre de la tabla padre + nombre de la tabla hija. (Véase **Fig. 2.4**).

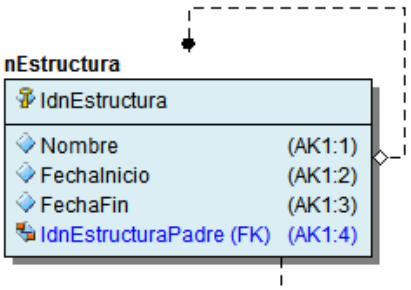


Fig. 2.3 Ejemplo de tabla nomencladora.

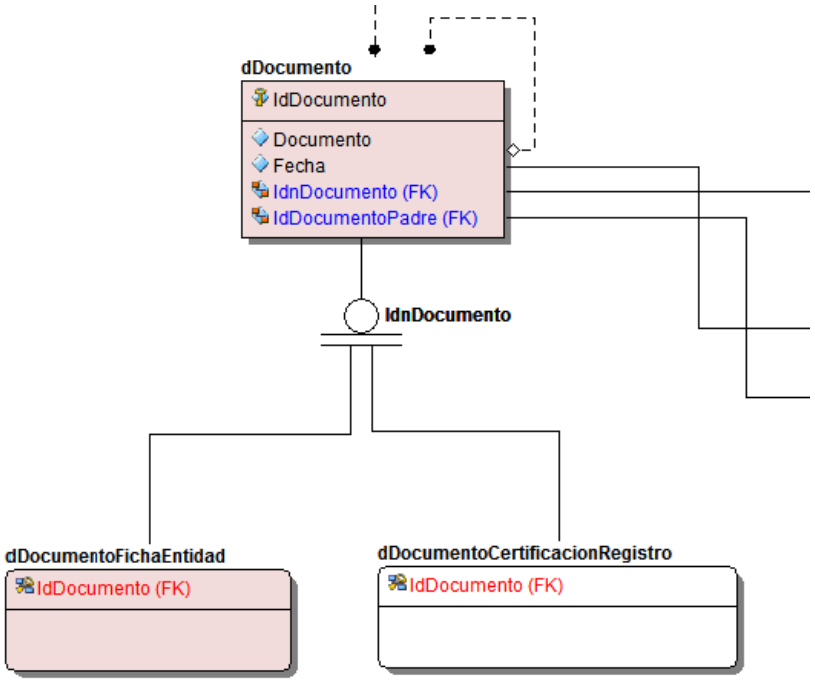


Fig. 2.4 Ejemplo de tabla de datos y especialización.

Para la notación de las llaves primarias se utilizará el siguiente patrón: `Id + nombre de la tabla`. A excepción de las llaves de las tablas de datos que comienzan con el prefijo `d` las cuales se nombraran `I + nombre de la tabla`. Ejemplo:

Tabla	Llave primaria
`nEstado`	`IdnEstado`
`dPersona`	`IdPersona`

Tabla 2.1 Notación de llaves primarias.

Las llaves foráneas deben tener el mismo nombre que aparece en la tabla a la que hace referencia donde son llave primaria, excepto cuando se especifique un rol; y la relación correspondiente entre tablas, debe nombrarse con los nombres de ambas tablas, primero con el nombre de la tabla padre y luego con el nombre de la tabla hija. (Véase **Fig. 2.5**).

Los índices se nombrarán con el prefijo `idx_+ nombre de la tabla + el nombre del campo` o los distintos campos según sea la estructura del índice. Ejemplo:

Tabla	Campo indexado	Índice
dPersona	ci	idx_dPersona_ci

Tabla 2.2 Notación de índices.

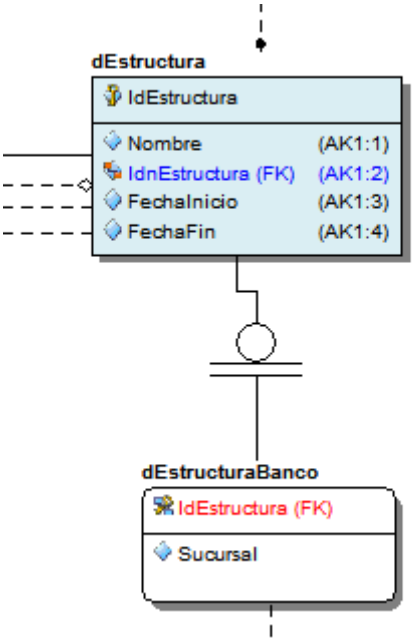


Fig. 2.5 Ejemplo de notación de llave foránea.

Otras consideraciones

- Se utilizará la notación Pascal para los nombres de los atributos y de las tablas.
- Se sustituirá la ñ por nn.
- Se utilizará el guión bajo lo menos posible.
- Se evitarán los acrónimos.
- No se utilizarán tildes, diéresis ni ningún carácter que no esté definido en el alfabeto inglés para un mejor manejo de los nombres en SGBD.
- No se utilizarán espacios en nombres incluso si el sistema lo permite. Tampoco guiones bajos en sustitución del espacio.

2.3 Requisitos del sistema propuesto

La captura de requisitos cumple un importante papel en la realización del producto, pues este describe el comportamiento del sistema a desarrollar. Los requisitos

especifican las funcionalidades que el sistema debe cumplir y las restricciones sobre las cuales este debe operar.

2.3.1 Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, más específicamente son acciones que el sistema debe ser capaz de realizar para satisfacer un contrato o documento impuesto.

RF 1 Registrar datos de la empresa

El sistema permitirá registrar todos los datos de la empresa.

RF 2 Registrar datos de expediente

El sistema permitirá registrar los datos específicos del expediente.

RF 3 Adicionar documento a expediente

El sistema permitirá adicionar al expediente el documento especificado. Estos documentos pueden ser: Carta de Solicitud de Afiliación, Ficha de la Entidad, Resolución de Constitución de la Empresa, Resolución de Nombramiento del Director, Argumentación con Criterio del Delegado y Aviso de cobro.

RF 4 Adicionar documento a expediente

El sistema permitirá adicionar automáticamente al expediente cualquier documento generado a través del sistema una vez se haya completado su edición. Estos documentos pueden ser Argumentación, Carta de aprobación, Criterio de la junta sobre admisión de asociación y Certificado de registro.

RF 5 Modificar documento de expediente

El sistema permitirá modificar la imagen de cualquier documento ya agregado al expediente.

RF 6 Eliminar documento del expediente

El sistema permitirá eliminar la imagen de cualquier documento ya agregado al expediente.

RF 7 Visualizar imagen del documento

El sistema permitirá visualizar la imagen de cualquier documento ya agregado al expediente.

RF 8 Mostrar documento pre elaborado

El sistema permitirá mostrar cualquier documento de los generados por el sistema. Estos documentos pueden ser: Argumentación, Criterio de la Junta sobre Admisión de Asociados, Carta de Aprobación o Certificado de Registro. De cada uno de ellos se mostrarán los datos que ya se tienen registrados en el sistema. Los datos son los siguientes: Argumentación (nombre de la empresa, ministerio al que pertenece, provincia y objeto social), Criterio de la Junta sobre Admisión de Asociados (nombre de la empresa y cuota), Carta de Aprobación (nombre de la empresa y cuota), Certificado de Registro (nombre de la empresa, fecha de inscripción, tomo, folio).

RF 9 Editar documento pre elaborado

El sistema permitirá editar cualquier documento generado. Estos documentos pueden ser: Argumentación, Criterio de la Junta sobre Admisión de Asociados, Carta de Aprobación o Certificado de Registro. De cada uno de ellos se mostrarán los datos que ya se tienen registrados en el sistema. Los datos son los siguientes: Argumentación (nombre de la empresa, ministerio al que pertenece, provincia y objeto social), Criterio de la Junta sobre Admisión de Asociados (nombre de la empresa y cuota), Carta de Aprobación (nombre de la empresa y cuota), Certificado de Registro (nombre de la empresa, fecha de inscripción, tomo, folio). El sistema permitirá editar los documentos pre elaborados con los datos restantes.

RF 10 Generar documento

El sistema permitirá generar en la plantilla correspondiente los documentos que se crean por el sistema. Ellos son: Argumentación, Criterio de la Junta sobre Admisión de Asociados, Carta de Aprobación, Certificado de Registro.

RF 11 Imprimir documentos

El sistema permitirá imprimir cualquiera de los documentos que se encuentren digitalizados.

RF 12 Aceptar o denegar expediente

El sistema le permitirá al Director de Formación y Membresía visualizar cada uno de los documentos que conforman el expediente, los cuales fueron presentados para solicitar la afiliación, estos deben ser: Carta de Solicitud de Afiliación, Ficha de la Entidad, Copia de la Resolución de Constitución de la Empresa y Copia de la Resolución de Nombramiento del Director. En el caso de las empresas radicadas en provincias estas deben presentar además Argumentación con el Criterio del Delegado

correspondiente. Una vez visualizados y aceptados o denegados se modificará el estado del expediente.

RF 13 Buscar empresa

El sistema permitirá buscar una empresa según los criterios de búsquedas especificados.

RF 14 Mostrar datos de la empresa

El sistema permitirá mostrar los siguientes datos de la empresa seleccionada: Datos de Contacto, Datos de Registro, Datos de la Actividad.

RF 15 Mostrar expediente

El sistema permitirá mostrar los datos del expediente de la empresa seleccionada. Estos datos son: número de expediente, fecha de creación, documentos (nombre, fecha, imagen).

RF 16 Aprobar o denegar expediente

El sistema permitirá a la Junta Ejecutiva la posibilidad de aprobar o denegar el expediente con los nuevos documentos adicionados.

RF 17 Aprobar o denegar afiliación de la empresa

El sistema modificará internamente el estado de la empresa, en correspondencia con el estado del expediente.

RF 18 Asignar tomo y folio

El sistema permitirá asignar automáticamente el tomo y folio para asentar el registro de la empresa.

RF 19 Mostrar fecha de inscripción

El sistema permitirá mostrar la fecha de inscripción de la empresa y el estado de aprobado, en los datos de la empresa.

RF 20 Modificar datos de la empresa

El sistema permitirá actualizar los datos de la empresa, teniendo como punto de partida los datos ya insertados los cuales también se pueden modificar.

RF 21 Reporte de Asamblea General

El sistema permitirá obtener un reporte de las empresas devolviendo en dicho reporte el nombre de la empresa, las siglas, la sección y la delegación.

RF 22 Reporte de cantidad de empresas por provincia

El sistema permitirá obtener un reporte de la cantidad de empresas por provincias que se encuentran afiliadas a la cámara. Mostrando el nombre de la provincia y la cantidad de empresas que le corresponde.

RF 23 Reporte de cantidad de empresas por sectores

El sistema permitirá obtener un reporte de la cantidad de empresas por sectores que se encuentran afiliadas a la cámara. Mostrando el sector y la cantidad de empresas que pertenecen al mismo.

RF 24 Reporte de cantidad de empresas por secciones

El sistema permitirá obtener un reporte de la cantidad de empresas por secciones que se encuentran afiliadas a la cámara. Mostrando la sección y la cantidad de empresas que pertenecen a la misma.

RF 25 Reporte de cuotas de asociados activos

El sistema permitirá obtener un reporte de las cuotas de todos los asociados activos. Mostrando el nombre de la empresa, la cuota, en que moneda está establecida la misma y la provincia.

RF 26 Reporte de datos de sitio web

El sistema permitirá obtener un reporte de los datos de las empresas asociadas que poseen sitio web. Mostrando de ellas los siguientes datos: entidad, siglas, dirección, teléfono, Correo, sitio web y objeto social.

RF 27 Reporte de entrega de invitaciones FIHAV

El sistema permitirá obtener un reporte de la entrega de invitaciones a la feria FIHAV de todas las empresas asociadas, como constancia de la entrega de la invitación. De todas las empresas asociadas se mostrarán los siguientes datos: número de economía, entidad, siglas, cuota.

RF 28 Reporte de fichas de cartas

El sistema permitirá obtener un reporte con las fichas de cartas para cada empresa de la provincia seleccionada.

RF 29 Reporte de reuniones sectoriales

El sistema permitirá obtener un reporte de las reuniones sectoriales por secciones. Mostrando los siguientes datos: entidad y siglas.

2.3.2 Requisitos no funcionales

Son propiedades o cualidades que el producto debe tener. Se debe considerar estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

RNF 1 Fiabilidad

- El SGBD debe presentar facilidades de administración de roles y usuarios restringiendo el acceso a los datos.
- La información manejada por el sistema deberá estar protegida de acceso y divulgación no autorizada.
- Debe garantizarse el resguardo de la información, así como el respaldo periódico de la información de la BD, de manera que sea posible la recuperación del sistema ante un eventual fallo.

RNF 2 Eficiencia

- El sistema debe funcionar con un máximo rendimiento pero ajustado a las prestaciones de hardware descritas en la arquitectura de datos basándose en el consumo mínimo de recursos.
- El intercambio de datos con el sistema mediante acciones de eliminación, inserción, búsqueda o modificación, el tiempo de respuesta debe ser menor de 1 segundo.
- La eficiencia del producto debe estar determinada por el aprovechamiento de los recursos que se disponen y la velocidad de las consultas en la BD.

RNF 3 Software

- El sistema operativo del servidor de BD debe ser Ubuntu 10.0.4 y el SGBD a emplear debe ser PostgreSQL 9.1.3.

RNF 4 Hardware

- El hardware a emplear se definió en el epígrafe 2.2 “Descripción de la arquitectura de datos”.

RNF 5 Portabilidad

- El SGBD contará con la posibilidad de ser portado a otro sistema operativo.

2.4 Diseño de la BD. Modelo de datos

La puesta en práctica de la BD es un paso decisivo en el desarrollo de aplicaciones de soporte del negocio. Para el diseño de la BD se realizó el modelo lógico partiendo de los requisitos identificados previamente por los analistas del proyecto e identificando las posibles entidades que persistirán datos y los patrones de diseño que permitirán darle solución a distintos problemas de diseño. Luego de realizado el diseño lógico de la BD se obtuvo el diseño físico que se muestra a continuación. El mismo consta con un total de 49 tablas de las cuales 21 son nomencladoras y el resto son de datos.

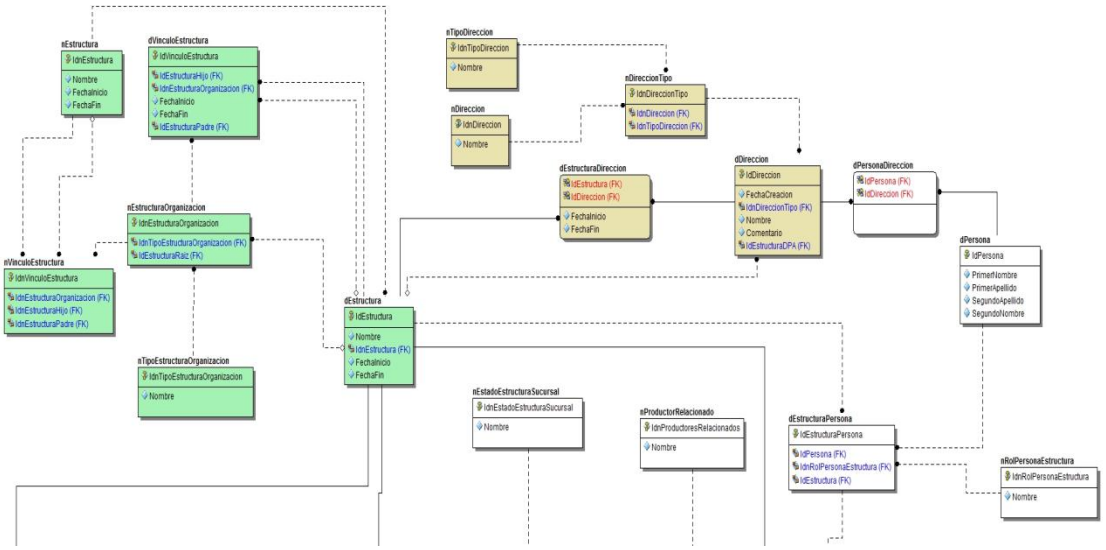


Fig. 2.6 Modelo físico (fragmento 1).

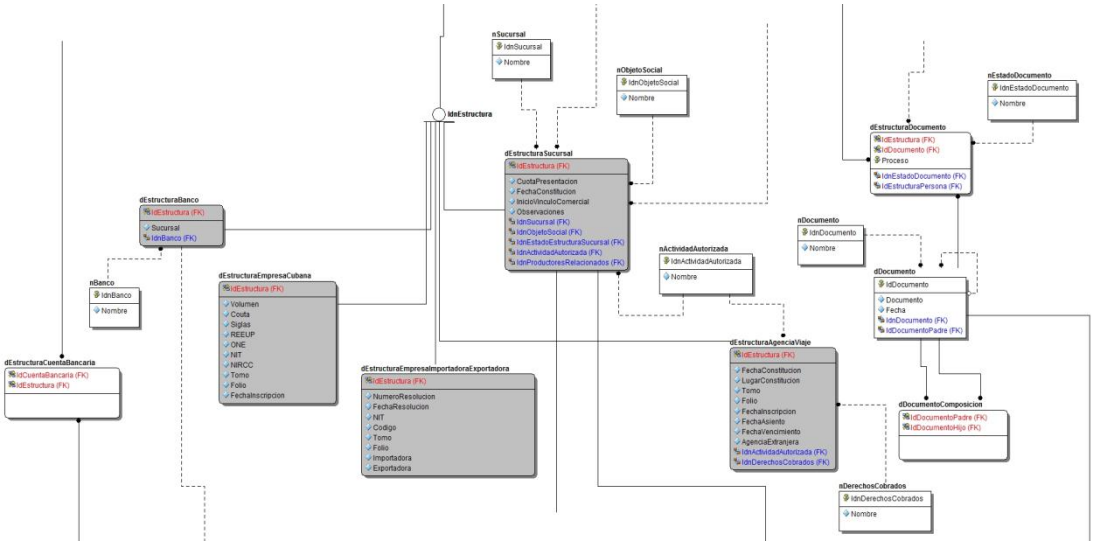


Fig. 2.7 Modelo físico (fragmento 2).

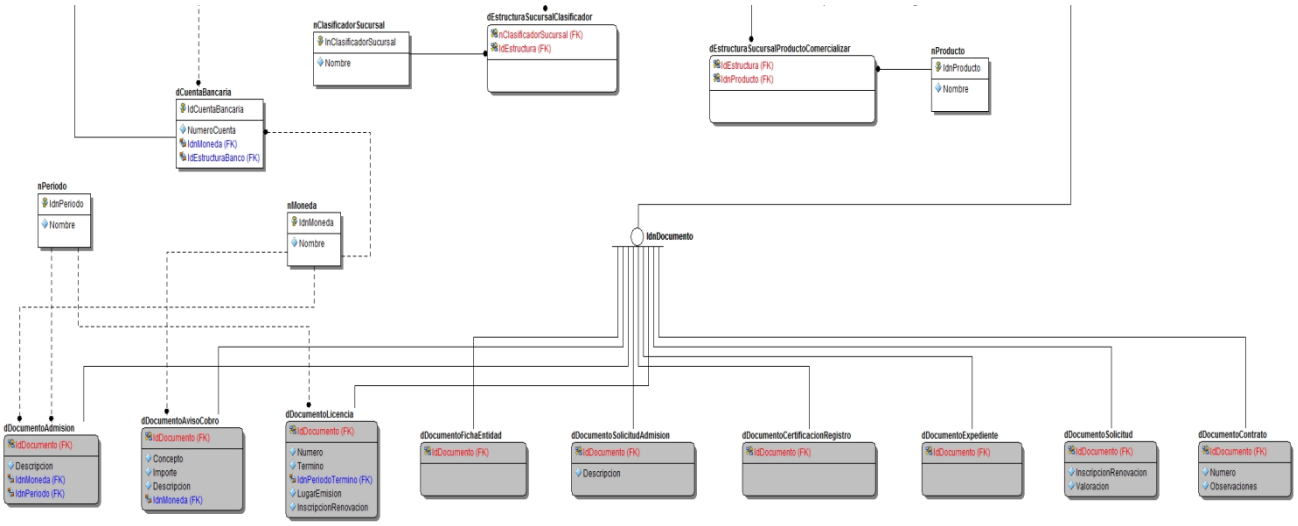


Fig. 2.8 Modelo físico (fragmento 3).

2.4.1 Patrones empleados

En el diseño de la BD del sistema SIRECC se emplearon varios patrones de diseños los cuales serán descritos a continuación.

Patrón Llave Subrogada

Este es uno de los patrones más usado. Este patrón estipula que se le cree una llave primaria única arbitraria para cada entidad. En otras palabras, esto significa que no es necesario encontrar dentro de los atributos que son representativos para el negocio uno que lo identifique inequívocamente pues se le agregará uno arbitrario para identificarlo. Normalmente se utilizan enteros para las columnas identificadoras aunque en el diseño de la BD del sistema SIRECC se emplearon además identificadores de tipo `char`.

	Attribute/Role Name	Domain	Datatype	Nulls
1	IdDireccion		SERIAL/INTEGER	IDENTITY
2	Nombre	Nombre	TEXT	NOT NULL
3	FechaCreacion	FechaHoraActual	TIMESTAMP/DATE	NOT NULL
4	Comentario	Texto	TEXT	NULL

Fig. 2.9 Empleo del patrón Llave subrogada para tipos INTEGER.

	Attribute/Role Name	Domain	Datatype	Nulls
1	IdnDireccionTipo		CHAR(10)	NOT NULL
2	IdnDireccion	EnteroLlave	INTEGER	NOT NULL
3	IdnTipoDireccion	EnteroLlave	INTEGER	NOT NULL

Fig. 2.10 Empleo del patrón Llave subrogada para tipos CHAR.

Patrón Árbol Cambiante

Este patrón es utilizado para representar la estructura jerárquica de entidades y sus relaciones. La aplicabilidad de este patrón se ve dado en que es muy flexible y permite representar estructuras que pueden dejar de existir en el tiempo o cambiar su estructura y jerarquía.

En la BD del sistema SIRECC se ve justificada su aplicación puesto que es necesario representar ciertas estructuras como empresas y sucursales donde la organización de estas no es la misma para todas las empresas afiliadas, e incluso puede dejar de existir una entidad sin que sea conveniente perder la información relacionada con ella.

La implementación de esta solución se observa en la **Fig. 2.11** y será detallada a continuación.

La tabla `nEstructura` guarda los nomencladores que denotan los tipos de estructuras. Estos pueden ser por ejemplo: Ministerio, Universidad, Provincia, Sucursal, etc. Esta tabla tiene una relación doble con `nVinculoEstructura`.

La tabla `nVinculoEstructura` almacena el vínculo que existe entre los diferentes tipos de estructuras. Es por eso la razón de la doble relación con la tabla `nEstructura`. Una almacena el tipo de estructura jerárquicamente superior y la otra la menor. Esta tabla se relaciona además con `nEstructuraOrganizacion` para indicar la organización a la que pertenece ese tipo de estructura.

La tabla `nTipoEstructuraOrganizacion` es utilizada para definir los tipos de organizaciones de estructuras. Aquí se podrá almacenar por ejemplo: División Política Administrativa, Organización Central del Estado, etc.

La tabla `nEstructuraOrganizacion` es utilizada para definir los tipos de organizaciones de estructuras con su estructura raíz que le da origen.

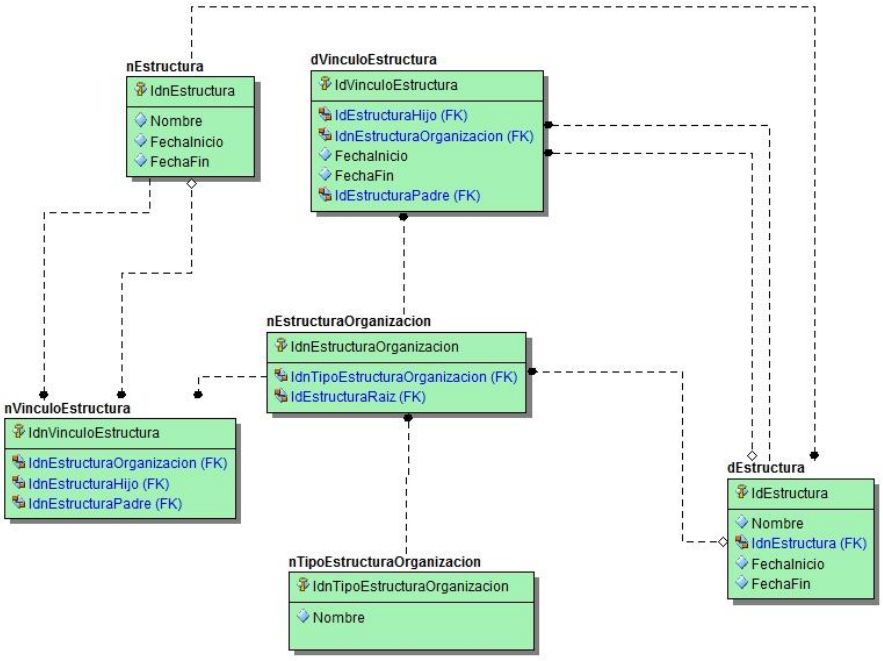


Fig. 2.11 Empleo del patrón Árbol cambiante.

La tabla `dEstructura` almacena la estructura en sí. Se relaciona con la tabla `nEstructuraOrganizacion` para almacenar a qué tipo de organización pertenece dicha estructura. Aquí se almacena por ejemplo: Cuba, CIMEX, Universidad de las Ciencias Informáticas, etc.

La tabla `dVinculoEstructura` es utilizada para definir las relaciones que existen entre las diferentes estructuras que conforman una jerarquía. Tiene una doble relación con la tabla `dEstructura` para indicar las estructuras que son jerárquicamente superiores o inferiores. Tiene otra relación con la tabla nomencladora `nEstructuraOrganizacion` para almacenar a qué tipo de organización pertenece.

Los atributos `FechaInicio` y `FechaFin` en las tablas en las que están presentes se usan para marcar el comienzo o el final de la activación de alguna estructura. Si ambos atributos son `null` entonces es que la estructura no ha dejado de existir ni ha sido creada nueva.

Patrón Árbol Simple

El patrón Árbol Simple es utilizado para representar estructuras jerárquicas simples donde los tipos de datos son del mismo tipo, por lo que pueden ser almacenados en la misma entidad. Se representa mediante la auto relación de una tabla.

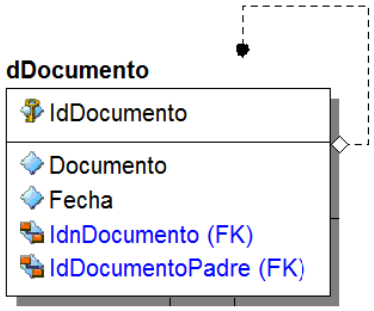


Fig. 2.12 Empleo del patrón Árbol Simple.

2.4.2 Descripción de algunas tablas de la BD

Se presentan a continuación la descripción de algunas tablas obtenidas como resultado del diseño de la BD realizado.

Nombre: dEstructura				
Descripción:	Tabla que almacena las estructuras organizativas que participan en los procesos.			
Llave	**Atributo**	**Tipo**	**¿Nulo?**	**Descripción**
PK	idEstructura	integer	No	Atributo identificador
	Nombre	text	No	Almacena el nombre de la estructura
FK	IdnEstructura	integer	No	Almacena el tipo de estructura
	FechaInicio	timestamp	Sí	Almacena la fecha a partir de la cual gana vigencia la estructura
	FechaFin	timestamp	Sí	Almacena la fecha a partir de la cual pierde vigencia la estructura

Tabla 2.3 Descripción de la tabla dEstructura.

Nombre: nEstructura	
Descripción:	Tabla que almacena los tipos de estructuras organizativas que participan en

	los procesos. Ejemplo: continente, país, provincia, universidad, empresa.			
Llave	**Atributo**	**Tipo**	**¿Nulo?**	**Descripción**
PK	IdnEstructura	int4	No	Atributo identificador
	Nombre	text	No	Almacena el nombre del tipo de estructura
	FechaInicio	timestamp	Sí	Almacena la fecha a partir de la cual gana vigencia el tipo de estructura
	FechaFin	timestamp	Sí	Almacena la fecha a partir de la cual pierde vigencia el tipo de estructura
FK	IdnEstructuraPadre	int4	Sí	Almacena el tipo de estructura jerárquicamente superior

Tabla 2.4 Descripción de la tabla nEstructura.

Nombre: dDocumento				
Descripción:	Tabla que almacena los documentos que intervienen en los procesos.			
Llave	**Atributo**	**Tipo**	**¿Nulo?**	**Descripción**
PK	IdDocumento	integer	No	Atributo identificador
	Documento	bytea	No	Almacena una imagen del documento físico
	Fecha	timestamp	No	Almacena la fecha en que se registró el documento
FK	IdnDocumento	int4	No	Almacena el tipo de documento
FK	IdDocumentoPa	int4	Sí	Almacena el documento del cual se deriva o no el documento

	dre			actual

Tabla 2.5 Descripción de la tabla dDocumento.

Nombre: nDocumento				
Descripción:	Tabla que almacena los tipos de documentos que intervienen en los procesos.			
Llave	**Atributo**	**Tipo**	**¿Nulo?**	**Descripción**
PK	IdnDocumento	int4	No	Atributo identificador
	Nombre	text	No	Almacena el nombre del tipo de documento

Tabla 2.6 Descripción de la tabla nDocumento.

Nombre: dPersona				
Descripción:	Tabla que almacena las personas naturales que participan en los procesos			
Llave	**Atributo**	**Tipo**	**¿Nulo?**	**Descripción**
PK	IdPersona	char(10)	No	Atributo identificador
	PrimerNombre	char(10)	Sí	Almacena el primer nombre de la persona
	SegundoNombre	char(10)	Sí	Almacena el segundo nombre de la persona
	PrimerApellido	char(10)	Sí	Almacena el primer apellido de la persona
	Segundo Apellido	char(10)	Sí	Almacena el segundo apellido de la persona
	Título	char(10)	Sí	Almacena el título de la persona

Tabla 2.7 Descripción de la tabla dPersona.

Nombre: dDireccion				
Descripción:	Tabla para almacenar los medios de comunicación			
Llave	**Atributo**	**Tipo**	**¿Nulo?**	**Descripción**
PK	IdDireccion	integer	No	Atributo identificador
	Nombre	text	No	Almacena el nombre del medio de comunicación
	FechaCreacion	timestamp	No	Almacena la fecha en que se creó el medio de comunicación
	Comentario	text	Sí	Almacena algún comentario sobre el medio de comunicación
FK	IdnDireccionTipo	char(10)	No	Almacena el tipo de medio de comunicación

Tabla 2.8 Descripción de la tabla dDireccion.

Nombre: nDireccion				
Descripción:	Tabla para almacenar los medios de comunicación. Ejemplo: teléfono, correo electrónico, URL, correo postal			
Llave	**Atributo**	**Tipo**	**¿Nulo?**	**Descripción**
PK	IdnDireccion	int4	No	Atributo identificador
	Nombre	text	No	Almacena el nombre del medio de comunicación

Tabla 2.9 Descripción de la tabla nDireccion.

Nombre: dEstructuraEmpresa	
Descripción:	Tabla para almacenar la información relacionada a las empresas.

Llave	Atributo	Tipo	¿Nulo?	Descripción
PK, FK	IdEstructura	int4	No	Atributo identificador
	Volumen	char(10)	Sí	Almacena la cantidad de operaciones que realiza la empresa
	Siglas	char(10)	Sí	Almacena las siglas de una empresa
	NIT	char(10)	Sí	Almacena el número de inscripción en el tomo
	NIRCC	char(10)	Sí	Almacena el número de inscripción en el registro de la Cámara de Comercio
	FechaInscripcion	char(10)	Sí	Almacena la fecha en que se inscribió en la Cámara de Comercio
	Tomo	char(10)	Sí	Almacena el tomo en que fue inscrita la empresa
	Folio	char(10)	Sí	Almacena el folio en que fue inscrita la empresa

Tabla 2.10 Descripción de la tabla dEstructuraEmpresa.

Nombre: dPersonaDireccion				
Descripción:	Tabla para almacenar las direcciones de contacto de las personas			
Llave	**Atributo**	**Tipo**	**¿Nulo?**	**Descripción**
PK, FK	IdPersona	char(10)	No	Atributo identificador
PK, FK	IdDireccion	Int4	No	Atributo identificador

Tabla 2.11 Descripción de la tabla dPersonaDireccion.

2.5 Optimización de la BD

Es posible lograr optimizaciones en la BD para agilizar las demoras del sistema ante operaciones que involucren un gran volumen de datos. Los mecanismos para optimizar BD se centran tanto en el nivel físico, ya sea distribuyendo la información en distintos ficheros, discos, o incluso servidores (BD distribuidas), realizando un mayor número de operaciones en paralelo; como en el nivel conceptual, realizando un modelo con el que se realicen menos operaciones costosas.

Para lograr la optimización de la BD se definieron ciertas pautas en su diseño para lograr un resultado óptimo en la realización de las consultas:

- El tamaño de los campos de las tablas está ajustado al máximo para evitar el desperdicio de espacio.
- Las tablas están normalizadas al menos hasta la 3ra forma normal lo que evita la duplicidad de los datos y consiguientemente el tiempo de búsqueda.
- Se emplearon índices.

Empleo de índices

Los índices en una BD se usan para agilizar las consultas a determinados datos, generalmente aquellos que son empleados con más frecuencia en las operaciones de búsqueda. La utilización de índices agilizará el acceso a los datos en las consultas que tengan condiciones asociadas a estos. Sin embargo la utilización de índices puede resultar contraproducente si se crean a campos sobre los que no se realizan ninguna petición ya que además de acrecentar el tamaño de la BD se ralentizan otras tareas de la BD como la inserción, actualización y eliminación.

Los criterios analizados para la creación de índices fueron:

- Las operaciones de unión entre tablas son grandes consumidoras de tiempo, por lo que la creación de índices para las llaves foráneas, que son por donde se realizan la mayoría de las uniones, puede ser ventajoso.
- Las operaciones de búsqueda que se realizan con mayor frecuencia sobre determinados datos de una tabla puede ser mejoradas con la utilización de índices.
- El número de filas que contiene una tabla así como los diferentes posibles valores que se pueden almacenar para una columna (cardinalidad de tabla y de

columna respectivamente). Mientras mayores sean estos valores, mayor será la ventaja del empleo de índices para este caso.

En la BD propuesta se crearon varios índices con el objetivo de disminuir el tiempo de repuesta e incrementar el rendimiento de la BD. Fueron creados índices para los campos que son llaves primarias y foráneas ya que generalmente son los usados para las operaciones de unión entre tablas. No fue necesaria la creación de otros índices puesto que con los criterios anteriormente expuestos no se justificó la creación de ningún otro índice.

2.6 Políticas de respaldo

Se creó una tarea programada que permitirá hacer una copia diaria de la BD y una copia total cada 7 días, y así tener un nivel de respaldo para enfrentar cualquier posible incidente en el que se pierdan los datos de la misma. Se almacenará una copia en el servidor de almacenamiento de datos y otra en el repositorio del proyecto. Ambos constarán con discos duros en disposición RAID 1 como media auxiliar de respaldo.

2.7 Conclusiones parciales

El diseño de la BD relacional obtenido responde a las necesidades identificadas por los requisitos funcionales y cumple con las especificaciones de los requisitos no funcionales para demás características. Se logró reducir la existencia de datos repetidos innecesariamente, disminuyendo la redundancia de datos y evitando consiguientemente posibles errores en la integridad de estos.

Se mostró el modelo físico de la BD en el que se identificaron 49 entidades y fueron empleados varios patrones de diseño para asegurar la calidad del mismo. Se describieron además algunas de las tablas que pueden crear más dudas. La utilización de índices permitió mejorar considerablemente las operaciones sobre los datos.

Capítulo 3: Validación de la solución

En el presente capítulo se realiza la validación del diseño de la BD teniendo en cuenta ciertos aspectos como: normalización, redundancia, integridad y seguridad. Se realizan además pruebas de rendimiento con el objetivo de comprobar el correcto funcionamiento de la BD.

3.1 Integridad de los datos

La integridad de los datos significa la exactitud, validez y corrección de los datos almacenados. Cuando se modela una BD se deben identificar ciertas reglas de integridad con el fin de asegurar los datos físicos, protegiéndolos de operaciones que atenten contra su integridad y validez. La integridad se expresa a través de reglas que la BD no puede quebrantar y de las que el SGBD es el encargado de hacer cumplir.

Cuando se modifica el contenido de la BD ya sea con operaciones *UPDATE, INSERT o DELETE,* puede verse comprometida la integridad de los datos si se adicionan de manera inválida, o se modifican otros ya existentes de modo que tomen valores incorrectos, o eliminando datos a los otros hagan referencia, por ejemplo eliminando datos de una tabla nomencladora.

La integridad de los datos se puede clasificarse en diferentes categorías: Integridad de Dominio, Integridad de Entidad e Integridad Referencial.

Integridad de Dominio

Una restricción de integridad de dominio es una regla que define el dominio de todos los valores válidos para los atributos de las diferentes tablas de una BD. Una de las tareas que permiten velar por la integridad del dominio es definir el tipo de datos, el formato y el rango de valores que puede aceptar un atributo. Para el caso del atributo `CI` que hace referencia al carnet de identidad de una persona se definió un dominio varchar (11) puesto que este número es de longitud fija de 11 caracteres.

En el caso de la tabla `destructuraempresa` se validaron los campos folio, tomo y volumen de manera tal que fuesen mayores que ceros mediante el empleo de restricciones *CHECK.*

```
CONSTRAINT destructuraempresa_folio_check CHECK
(folio>0::numeric),
```

```
CONSTRAINT destructuraempresa_tomo_check CHECK
(tomo>0::numeric),

CONSTRAINT destructuraempresa_volumen_check CHECK
(volumen>0::numeric)
```

Para lograr la integridad de dominio se hizo uso de los tipos de datos predefinidos por PostgreSQL, específicamente integer, varchar, serial, timestamp. Además se revisan los valores que no pueden clasificarse de nulos haciendo uso de la restricción NOT NULL.

Integridad de Entidad

La integridad de entidad identifica a cada instancia de una entidad como única. Esta se refiere a las llaves primarias de las tablas, estableciendo que ningún atributo que forme parte de la llave primaria de la tabla puede aceptar valores nulos y que cada instancia debe poseer un valor único. En el caso de la BD del sistema SIRECC, cada entidad tiene definida su llave primaria, que no puede ser nula ni se puede repetir, a través de la restricción *PRIMARY KEY,* que automáticamente asegura que sea única (*UNIQUE*) y no nula (*NOT NULL*).

```
CONSTRAINT pk_dpersona PRIMARY KEY (idpersona)
```

Integridad Referencial

La integridad referencial se especifica entre 2 o más relaciones y se utiliza para asegurar la consistencia entre las tuplas de las relaciones. Esta integridad cuida que un dato al que se haga referencia en una relación o tabla, exista realmente en la tabla referenciada. Este dato o conjunto de datos es denominado llave foránea y se establece a través de la cláusula *FOREING KEY* que se añade al atributo. Cuando se define una columna como clave foránea, las filas de la tabla pueden contener en esa columna el valor nulo, o el valor al que se hace referencia en la otra tabla.

Por ejemplo, la tabla `dpersonadireccion` se creó como resultado de la relación *mucho a mucho* entre la tabla `dpersona` y la tabla `ddireccion`. Esta tabla almacena los identificadores de ambas tablas que intervienen en la relación; que juntas constituyen su llave primaria y cada una por separada son llaves foráneas que hacen referencia al identificador de cada una de las tablas de la relación. Para poder crear un registro en la tabla `dpersonadireccion` es necesario validar previamente la existencia de los identificadores de las tablas relacionadas.

```
CONSTRAINT fk_ddirecciondpersonadireccion FOREIGN KEY
(iddireccion)

    REFERENCES sirecc.ddireccion (iddireccion) MATCH
    SIMPLE

    ON UPDATE NO ACTION ON DELETE NO ACTION,

CONSTRAINT fk_dpersonadpersonadireccion FOREIGN KEY
(idpersona)

    REFERENCES sirecc.dpersona (idpersona) MATCH
    SIMPLE

    ON UPDATE NO ACTION ON DELETE NO ACTION
```

En el caso de la BD del sistema SIRECC, el SGBD PostgreSQL se encargar de validar implícitamente la validez de la integridad referencial. Siendo así, el sistema no permite añadir datos en una tabla con llaves foráneas a otra y que no esté previamente creado estos datos.

3.2 Normalización de la BD

El proceso de normalización se realiza mientras se crea el diseño lógico de la BD. Este proceso consiste en la aplicación de una serie de reglas o condiciones a las relaciones (tablas) identificadas en el modelo con el objetivo de proteger la integridad de los datos y evitar:

- La redundancia y dependencia de los datos.
- Inconsistencia de actualización de datos.
- Anomalías de borrado e inserción de datos.

Existen varias formas normales (FN), sin embargo con las 3 primeras es suficiente para resolver los problemas de redundancia e inconsistencia de los datos. Las distintas FN se relacionan dependientemente entre ellas. Esto significa que para que una BD se encuentre en una determinada forma normal, debe cumplir con las FN de los niveles inferiores.

Primera Forma Normal (1FN)

Para que una relación se encuentre en 1FN se debe asegurar que todos sus atributos sean atómicos, o sea, indivisibles. Además esta debe poseer una llave primaria única que no contenga atributos que puedan ser nulos y así poder identificar cada tupla

inequívocamente. Además los atributos no pueden ser multivaluados y además deben ser del mismo tipo de dato para todos los registros de la tabla.

La BD del sistema SIRECC cumple con esta restricción para todas sus tablas ya que los datos de las relaciones involucradas son atómicos, no son ni multivaluados, ni compuestos y cada una de las tablas obtenidas cuenta con una llave primaria única.

Segunda Forma Normal (2FN)

El primer requisito para que una tabla o conjunto de tablas estén en 2FN es que se encuentre previamente en 1FN. Una vez esté ya la relación en 1FN, esta debe cumplir con las nuevas restricciones que añade este nivel de normalización a tablas que tienen llave compuesta.

La 2FN compara cada uno de los atributos que no forman parte de la llave primaria compuesta y analiza si estos dependen total y funcionalmente de la llave primaria. En caso de que exista solamente dependencia de uno de los atributos que componen la llave primaria entonces la relación no está en 2FN. De aquí que una tabla estará solamente en 2FN si y solo si cualquier atributo que no sea parte de la llave primaria depende de toda la llave primaria en vez de solo una parte de ella. Es por ello que cuando una tabla no tiene una llave primaria compuesta, está de por sí en 2FN.

Para pasar una relación en 1FN a 2FN hay que eliminar las dependencias parciales de la llave primaria. Para lograrlo, se eliminan los atributos que son funcionalmente dependientes y se ponen en una nueva relación con una copia de su llave primaria.

La BD en cuestión se encuentra también en 2FN puesto que además de estar en 1FN, las llaves primarias de las distintas tablas no son compuestas y todos los atributos dependen totalmente de ellas; y las que tienen llaves compuestas cumplen con la dependencia funcional completa de la llave primaria.

Tercera Forma Normal (3FN)

Una precondición para que una tabla o conjunto de tablas estén en 3FN es que se encuentren previamente en 2FN.Luego se analiza si cada atributo de la relación que no está contenido dentro de la llave primaria depende solo de ella y no de ningún otro atributo. El objetivo es eliminar las dependencias transitivas, o sea, aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave.

La BD propuesta para el sistema SIRECC, cumple con la 3FN para todas sus tablas, pues todas se cumplen con los requisitos planteados hasta la segunda forma normal y

además todos los atributos dependen de manera directa de la llave primaria de la tabla a la que pertenecen.

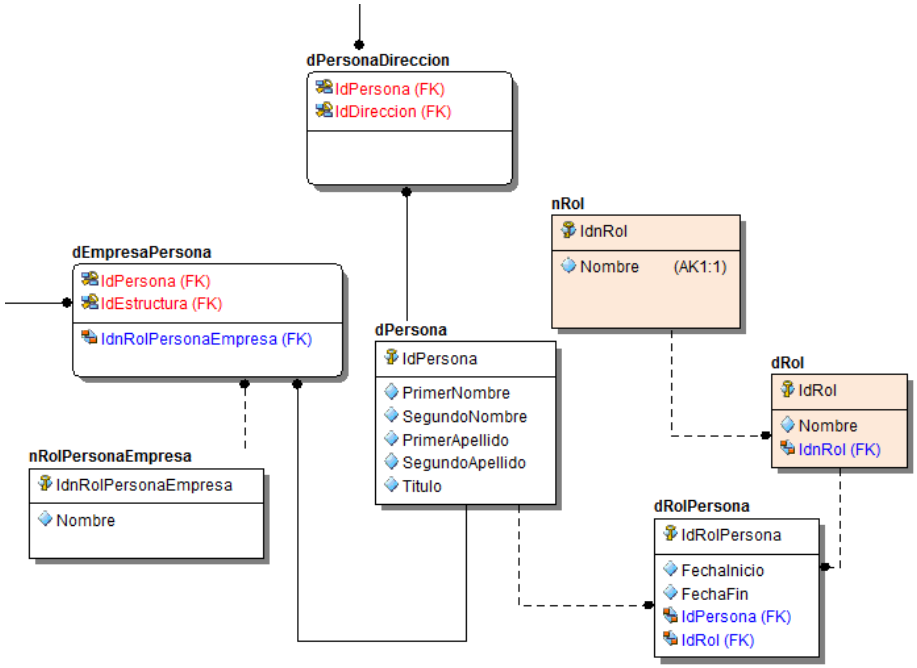


Fig. 3.1 Ejemplo de tablas en 3 FN.

Desnormalización

La desnormalización es una de las tareas que se realiza durante el diseño lógico de la BD, después de que se le hayan practicado reglas de normalización. Este proceso introduce redundancias controladas en ciertas tablas con el objetivo de mejorar las prestaciones del sistema puesto que al hacer esto se evita tener que hacer operaciones de unión (join) entre diferentes tablas para obtener ciertos datos.

La desnormalización puede hacer que el acceso a datos sea más rápido, sin embargo ralentiza las actualizaciones ya que hay que actualizar en cascada todos los datos redundantes.

3.3 Análisis de redundancia

Redundancia de información en una BD significa la repetición innecesaria de datos. Además del aumento innecesario del tamaño de la BD y de aumentar los costes de mantenimiento y el tiempo de acceso a los datos, la redundancia es una de las fuentes más frecuentes de inconsistencia de datos.

Un punto importante para lograr la eliminación de la redundancia es la normalización del diseño. Un buen diseño normalizado del modelo de datos al menos hasta la 3 FN, logrará evitar la aparición de datos redundantes.

Teóricamente lo ideal sería lograr la redundancia cero en el modelo, sin embargo existen ocasiones, que por lo repetido que se necesitan ciertos datos, es más factible tenerlos repetidos en una tabla, para de esta manera evitar la pérdida de tiempo en las consultas durante la unión de varias tablas. La presencia de redundancia implicaría la necesidad de utilizar triggers para asegurar que se mantenga la consistencia cuando los datos son insertados, modificados o eliminados.

En el caso del diseño de la BD para el sistema SIRECC no fue necesario mantener datos duplicados por motivos de la índole anteriormente expuesta por lo que al estar en 3FN se elimina totalmente la redundancia en todo el diseño.

3.4 Análisis de seguridad

El tema de la seguridad de los sistemas informáticos siempre es un punto clave en el diseño de los mismos, especialmente en aquellos a los que se accede desde la Internet. Siendo los datos almacenados en BD uno de los principales objetivos de los ataques a los sistemas, cobra siempre vital importancia el análisis de este punto para evitar así acceso no autorizado a los datos o la manipulación malintencionada de estos.

Una de las medidas de seguridad tomada fue que la conexión con la BD no se establecerá con el usuario propietario de la misma o super-usuario, ya que este tiene control total de la BD y puede ejecutar cualquier consulta, pudiendo modificar el esquema de esta; eliminando, modificando tablas o eliminando los datos almacenados en ellas. Solamente el arquitecto y el diseñador de la BD se podrán conectar mediante este usuario.

Para la conexión del resto del equipo de desarrollo se creó un usuario *desarrollador*, a través del cual se conectarán a la BD con permisos limitados. Esto posibilita que solamente se tenga acceso a los datos a los que fueron definidos para ese usuario y se evita que si algún intruso gana acceso a los datos, este pueda afectar más que lo que está definido para el usuario por el que se conectó. Se reduce de esta manera las posibilidades de ataques o errores por acceso en la BD.

Las contraseñas de los usuarios, tanto para los desarrolladores como para los administradores de la BD estarán encriptadas y solo podrán ser cambiadas por el administrador de la BD. Estas deberán además tener caracteres no alfabéticos como medida extra para fortalecer la contraseña.

En caso de una posible corrupción de los datos por fallas en la seguridad de la BD, estos podrán ser restaurados a partir de las salvas diarias que se realizan a la misma.

3.5 Pruebas de volumen

El objetivo de la realización de las pruebas de volumen es analizar el comportamiento de la BD, con volúmenes de datos similares a los esperados que serán almacenados en el ambiente en el que será explotado realmente el sistema. Se verifica si la BD alcanza su límite de almacenamiento pudiendo causar fallas en el sistema.

Para poblar la BD con datos de prueba se utilizó la herramienta *EMS Data Generator for PostgreSQL* en su versión 3.0. Esta herramienta permite seleccionar tablas y columnas para generar datos, definir rangos de valores dependiendo del tipo de dato, generar columnas de caracteres por máscara, etc. Una de las ventajas que tiene es que genera los datos que provienen de otras tablas para evitar errores de integridad referencial.

En las pruebas de volumen se agregaron un total de 51046 tuplas en aproximadamente 17:06 min aproximadamente, quedando distribuidas de la siguiente manera:

Tabla	Tuplas
dCuentaBancaria	1000
dDireccion	1000
dDocumento	5000
dDocumentoAdmision	1000
dDocumentoAvisoCobro	1000
dDocumentoCertificacionRegistro	1000
dDocumentoComposicion	1000
dDocumentoContrato	1000

dDocumentoExpediente	1000
dDocumentoFichaEntidad	1000
dDocumentoLicencia	1000
dDocumentoSolicitud	1000
dDocumentoSolicitudAdmision	1000
dEstructura	2500
dEstructuraAgenciaViaje	500
dEstructuraBanco	100
dEstructuraCuentaBancaria	1000
dEstructuraDireccion	5000
dEstructuraDocumento	3000
dEstructuraEmpresaCubana	1000
dEstructuraEmpresaImportadoraExportadora	1000
dEstructuraPersona	5000
dEstructuraSucursal	1000
dEstructuraSucursalClasificador	500
dEstructuraSucursalProductoComercializar	500
dPersona	2000
dPersonaDireccion	5000
dVinculoEstructura	5000
nActividadAutorizada	200
nBanco	20
nClasificadorSucursal	2
nDerechosCobrados	100
nDireccion	20

nDireccionTipo	20
nDocumento	20
nEstadoDocumento	10
nEstadoEstructuraSucursal	10
nEstructura	50
nEstructuraOrganizacion	15
nMoneda	2
nObjetoSocial	20
nPeriodo	15
nProducto	200
nProductoRelacionado	20
nRolPersonaEstructura	100
nSucursal	2
nTipoDireccion	20
nTipoEstructuraOrganizacion	50
nVinculoEstructura	50
Cantidad total	**51046**

Tabla 3.1 Cantidad de datos introducidos.

En la realización de las pruebas fueron creadas 5000 tuplas en la tabla `dDocumento` en el que el atributo `documento` almacena datos binarios de alrededor de 1 MB. Es por esto que el llenado de la BD se tomó un tiempo considerable.

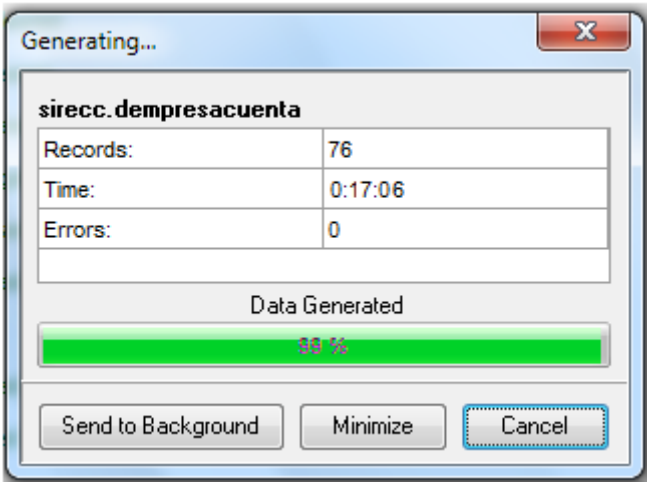


Fig. 3.2 Realización del llenado de la BD.

3.6 Pruebas de carga

Una prueba de carga se ejecuta para conocer el comportamiento de la BD ante situaciones de estrés determinada. Estas pruebas son realizadas normalmente para someter la aplicación ante un nivel de carga de ejecución muy superior al esperado y así conocer las potencialidades del sistema ante situaciones exigentes. Otro objetivo de estas pruebas es el de determinar los limites reales en cuanto al número de usuarios concurrentes que puede soportar la aplicación y la velocidad de las transacciones que esta permite.

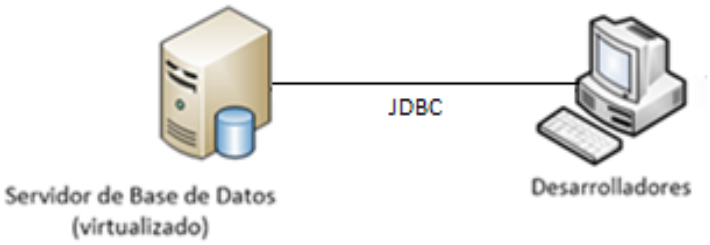


Fig. 3.3 Configuración para las pruebas de carga.

La configuración del Apache Jmeter quedo de la siguiente forma:

- Se creó un grupo de hilos donde se definió la cantidad peticiones al servidor. (véase **Fig. 3.4**)
- Se configuró el acceso a la BD en el archivo *Configuración de la conexión JDBC* en el cual se detalla la dirección url a través de la cual se conecta a la BD, usuario y contraseña (véase **Fig. 3.5**)

- Se definió la sentencia SQL que serviría de petición se le realizaría a la BD en el archivo *Petición JDBC*. (véase **Fig. 3.6**)

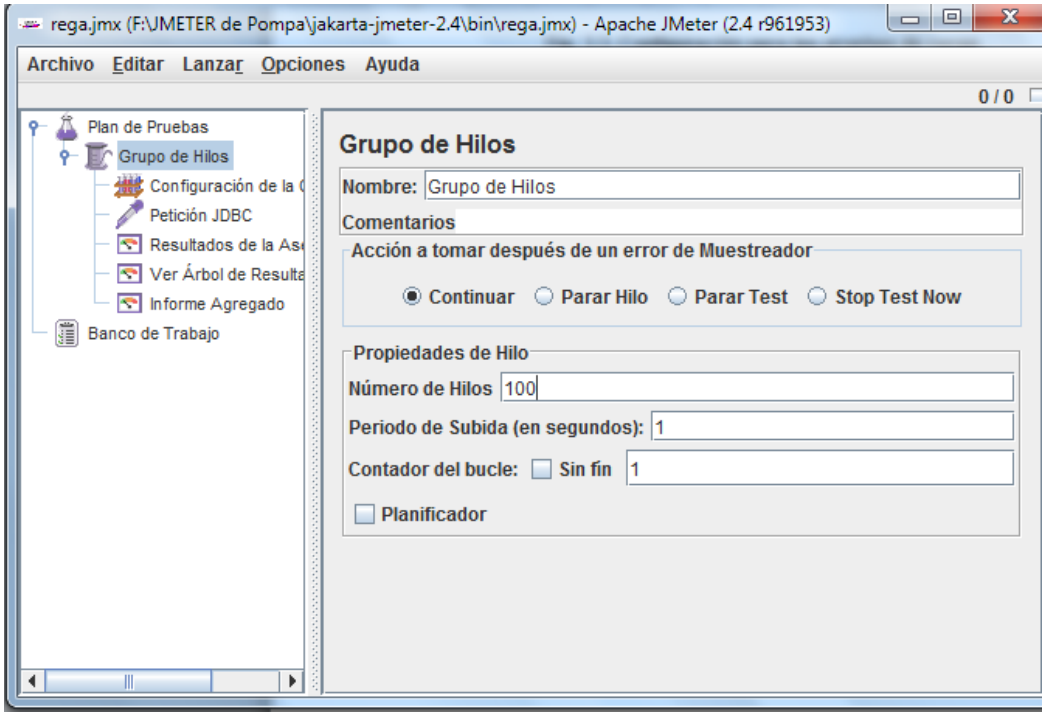


Fig. 3.4 Configuración de Grupos de Hilos.

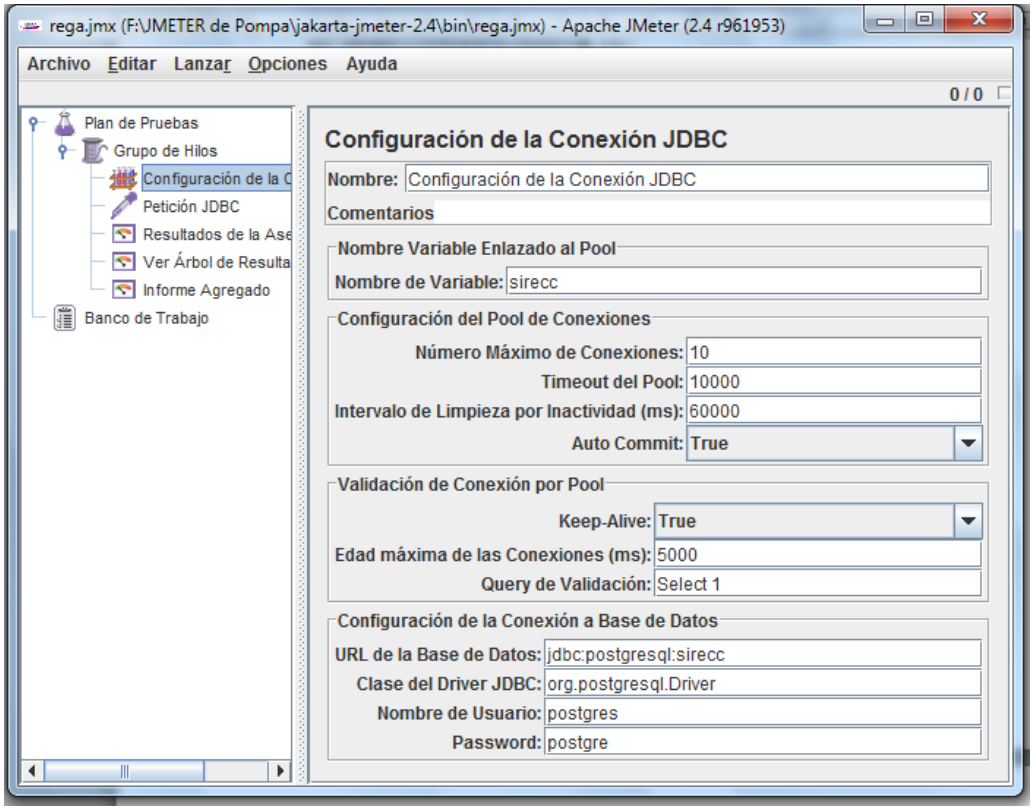


Fig. 3.5 Configuración de la conexión JDBC.

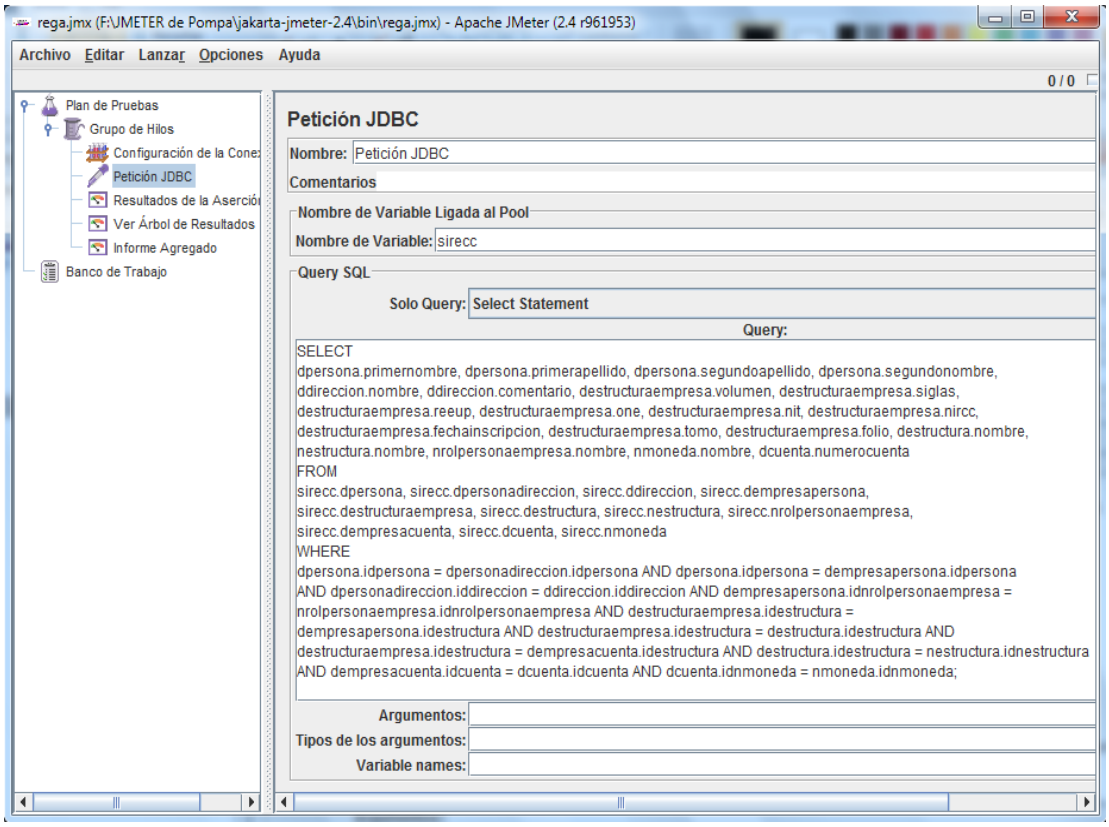


Fig. 3.6 Configuración de la petición JDBC.

Las pruebas fueron realizadas utilizándose varias consultas (Véase **Anexo 3**) con diferente cantidad de usuarios concurrentes. Los resultados arrojados se muestran en las siguientes variables:

- **Media**: tiempo promedio de respuesta de todas las peticiones.
- **Mediana**: tiempo promedio de la mitad de los resultados, la otra mitad tomará un tiempo entre la mediana y el valor máximo.
- **Mínimo**: mínimo tiempo de respuesta de una petición.
- **Máximo**: máximo tiempo de respuesta de una petición.

Usuarios concurrentes	Media (ms)	Mínimo (ms)	Máximo (ms)	Mediana (ms)
5	6 858	6 180	9 438	8 721
100	24 351	8 232	50 113	35 178

Tabla 3.2 Resultados de la prueba 1.

Usuarios concurrentes	Media (ms)	Mínimo (ms)	Máximo (ms)	Mediana (ms)

5	8 435	7 713	10 503	9 047
100	28 115	9 458	57 134	38 899

Tabla 3.3 Resultados de la prueba 2.

Usuarios concurrentes	Media (ms)	Mínimo (ms)	Máximo (ms)	Mediana (ms)
5	7 180	6 640	9 704	9 125
100	27 696	10 668	53 272	36 348

Tabla 3.4 Resultados de la prueba 3.

Usuarios concurrentes	Media (ms)	Mínimo (ms)	Máximo (ms)	Mediana (ms)
5	10 947	8 790	12 078	9 812
100	26 515	11 856	54 113	38 781

Tabla 3.5 Resultados de la prueba 4.

Usuarios concurrentes	Media (ms)	Mínimo (ms)	Máximo (ms)	Mediana (ms)
5	9 846	8 198	10 874	9 124
100	25 435	10 804	53 303	39 047

Tabla 3.6 Resultados de la prueba 5.

3.7 Valoración de los resultados

Después poblar la BD con la ayuda de la herramienta de software utilizada, no se encontraron errores con respecto al límite de capacidad del volumen de datos de la misma, desbordamientos de columnas, atributos o tipos de datos. Esto asegura un buen diseño de las entidades identificadas en el diseño y que el SGBD utilizado soporta los niveles de almacenamiento requeridos por la aplicación.

Se obtuvieron además los siguientes resultados:

- La BD cumple con los requisitos funcionales.
- La BD fue correctamente normalizada hasta la 3 FN.

- Las restricciones de integridad aseguran que los datos introducidos sean los correctos.
- Se evitó la redundancia de la información.

Luego de realizadas las pruebas de carga a la BD se concluyó que estas arrojaron resultados satisfactorios para la cantidad de información generada para las pruebas, la que es superior que será generada por el proyecto en su puesta en explotación.

Sin embargo estos resultados no pueden ser tomados como reales ya que existen varios factores que influyen en la rapidez como la cantidad de usuarios conectados a la aplicación simultáneamente y el nivel de concurrencia de las solicitudes.

3.8 Conclusiones parciales

Durante el desarrollo del presente capítulo se analizaron aspectos de gran importancia para el correcto funcionamiento de la BD. Se concluyó que las restricciones de integridad garantizan la validez de los datos introducidos. No se presenta redundancia de datos en el diseño realizado puesto que el mismo fue correctamente normalizado hasta la 3 FN. Fue tomada en cuenta la seguridad de la BD para lo que se crearon varios usuarios con distintos privilegios así como contramedidas para posibles eventualidades.

La utilización de herramientas para la realización de las pruebas de carga y volumen de la BD permitieron asegurar el correcto funcionamiento de esta bajo condiciones críticas de trabajo y alta concurrencia de peticiones, así como que las restricciones de integridad y relaciones entre las tablas fueron correctamente diseñadas.

Conclusiones

Las BD representan uno de los elementos fundamentales para el correcto funcionamiento de cualquier empresa, pues almacenan toda la información valiosa que se maneja en estas.

En el presente trabajo de diploma se reflejó el estudio realizado que permitió la generación de una propuesta de solución que permitiera realizar los procesos de gestión del registro de la Cámara de Comercio de una manera rápida, eficiente y centralizada, obteniéndose las siguientes conclusiones:

- Se realizó un estudio acerca de los elementos teóricos relacionados con el diseño de BD que favoreció el correcto diseño de la misma.
- Se realizó el diseño de la BD, el cual cumple con los requerimientos especificados, aplicándose diversos patrones para mejorar su calidad.
- Se validó la BD en la que se realizaron pruebas de carga y volumen obteniéndose resultados satisfactorios y se analizó la integridad de los datos, la normalización, la redundancia y la seguridad de la BD.

Recomendaciones

Las metas planteadas con este trabajo se han cumplido y en base a los resultados obtenidos se recomienda:

- Mejorar las prestaciones del hardware utilizado en el servidor de la BD para lograr un mayor rendimiento.
- Dar continuidad al trabajo realizado para robustecer el diseño.
- Que se ponga en práctica la utilización de este diseño.
- Proporcionar un mantenimiento y soporte regular a la base de datos enfocados a su mejor funcionamiento.

Bibliografía

Abraham Silberschatz, Henry F. Korth y S. Sudarshan. 2002. *Database System Concepts. Cuarta edición.* 2002.

Alberca Manzaneque, Alejandro y Díaz, Jesús Galvez. 2008. *Modelos Avanzados de Bases de Datos. Bases de datos Orientadas a Objetos y Bases de Datos Objeto-Relacionales.* 2008.

ANSI/SPARC. 1975. *Study Group on Data Base Management Systems. Interim Report. FDT. Volumen 7 No.2.* 1975.

Bachmann, Charles. 2009. *The Origin of the Integrated Data Store (IDS): The First Direct-Access DBMS.* 2009.

Blaha, Michael. 2010. *Patterns of Data Modelling.* s.l. : CRC Press, 2010.

Boehm, Barry W. 1996. *Anchoring the Software Process.* 1996.

Catalá Mallofré, Andreu. 2006. Sistema de agentes portables incrustados para entornos naturales seguros (SAPIENS). 2006.

Codd, Edgar F. 1970. *A Relational Model for Large Shared Data Banks.* 1970.

Date, C. J. 2001. *An introduction to datábase systems. Seventh Edition.* 2001.

Esakkirajan, S. Sumathi y S. 2007. *Fundamentals of Relational Database Management Systems.* 2007.

García, Rosa M. Mato. 2005. *Diseño de Bases de Datos.* 2005.

Gray, Jim. 1981. *The Transaction Concept: Virtues and Limitations.* 1981.

Hansen, Gary W. Hansen y James V. 2006. *Diseño y administración de Bases de Datos, 2 edición.* 2006.

Larman, C. 2004. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2004.

Matthew, Neil y Stones, Richard. 2005. *Beginning Databases with PostgreSQL. From Novice to Professional, Second Edition.* 2005.

Microsoft Corporation. 2012. http://www.microsoft.com/. [En línea] 2012. [Citado el: 16 de 1 de 2012.] http://www.microsoft.com/sqlserver/en/us/product-info/why-sql-server.aspx.

Oracle Corporation. 2012. http://www.mysql.com/. [En línea] 13 de 1 de 2012. http://www.mysql.com/why-mysql/.

Orallo, José Hernández. 2002. *La Disciplina de los Sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas.* 2002.

PostgreSQL Global Development Group. 2012. http://www.postgresql.org/. [En línea] 2012. [Citado el: 13 de 1 de 2012.] http://www.postgresql.org/about/.

Quiroz, Javier. 2003. *El modelo relacional de bases de datos.* 2003.

Rational®. 1998. Best Practices for Software Development Teams. 1998.

—. 2001. *Rational Unified Process:Best Practices for Software development Teams.* 2001.

Riordan, Rebecca M. 1999. *Designing Relational Database Systems.* 1999.

Rivera, Javier Fernández. 2011. http://aurea.es/. [En línea] 2011. [Citado el: 6 de 12 de 2011.] http://aurea.es/wp-content/uploads/modelodedatos.pdf.

Silverio Castro, Rogelio. 2005. *La problemática de las Bases de datos Distribuidas.* 2005.

Solis Álvarez, Camilo Javier y Gus Díaz, Roberth . 2007. Metodologías Tradicionales vs. Metodologías Ágiles. 2007.

Visual Paradigm. 2012. http://www.visual- paradigm.com/. [En línea] 2012. [Citado el: 26 de 1 de 2012.] http://www.visual- paradigm.com/product/vpuml/.

Zorrilla, Pantaleón. 2003. *Introducción a las Bases de Datos.* 2003.

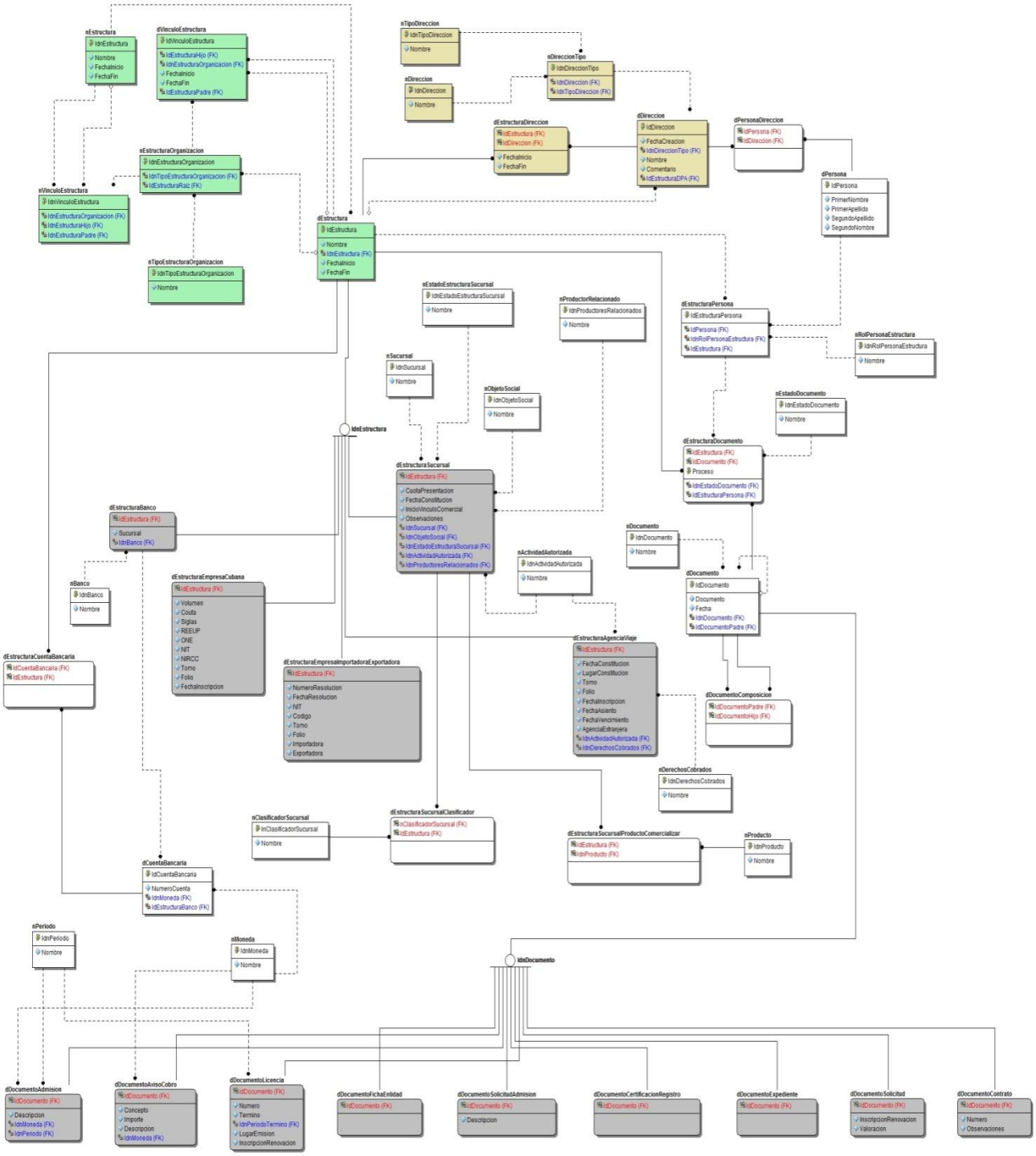
—. 2009. *Modelos de Datos.* 2009.

Anexos

Anexo 1. Diccionario de Datos

Dominio	Tipo de Dato	Valor por Defecto	Campos	Restricción
Binario	VARBINARY/BLOB		Cualquier fichero físico binario.	
Booleano	BIT	FALSE	Campos de verdadero o falso.	@var = TRUE or @var = FALSE
CorreoElectronico	TEXT		Campo de correo electrónico.	@varlike '%_@___%'
EnteroLlave	INTEGER		Números enteros positivos. Campos llave primaria de las tablas nomencladoras.	@var> 0
FechaHoraActual	TIMESTAMP/DATE	NOW()	Fecha y hora actual.	
FechaHoraNull	TIMESTAMP/DATE	NULL	Fecha y hora que admiten valores nulos.	
Nombre	TEXT		Nombres	
Numero	NUMERIC(0, 0)		Números positivos con lugares decimales.	@var> 0
NumeroCI	VARCHAR(11)		Número de carne de identidad.	@varlike '__________'
Postgres_IPAddress	INTEGER	CIDR	Números IP.	
Postgres_MACAddress	INTEGER	MACADDR	Números MAC.	
Postgres_User	TEXT	USER	Usuario conectado a la BD.	
	SERIAL/INTEGER		Campos llave primaria de las tablas de datos.	
Texto	TEXT		Texto en general.	

Anexo 2. Diagrama Entidad – Relación



Anexo 3. Consultas de Pruebas

Prueba 1

```
SELECT
  destructuraagenciaviaje.fechaconstitucion,
  destructuraagenciaviaje.lugarconstitucion,
  destructuraagenciaviaje.tomo,
  destructuraagenciaviaje.folio,
  destructuraagenciaviaje.fechainscripcion,
  destructuraagenciaviaje.fechaasiento,
  destructuraagenciaviaje.fechavencimiento,
  destructuraagenciaviaje.agenciaextranjera,
  destructuraagenciaviaje.idnactividadautorizada,
  destructuraagenciaviaje.idnderechoscobrados,
  dpersona.primernombre,
  dpersona.primerapellido,
  dpersona.segundoapellido,
  dpersona.segundonombre
FROM
  sirecc.destructura,
  sirecc.destructuraagenciaviaje,
  sirecc.destructurapersona,
  sirecc.dpersona
WHERE
  destructura.idestructura = destructuraagenciaviaje.idestructura AND
  destructurapersona.idestructura = destructura.idestructura AND
  dpersona.idpersona = destructurapersona.idpersona;
```

Prueba 2

```
SELECT
  destructura.nombre,
  destructurasucursal.cuotapresentacion,
  destructurasucursal.fechaconstitucion,
  destructurasucursal.iniciovinculocomercial,
  destructurasucursal.observaciones,
  destructurasucursal.objetosocial,
  nestadoestructurasucursal.nombre,
  nclasificadorsucursal.nombre,
  nsucursal.nombre
FROM
  sirecc.nclasificadorsucursal,
  sirecc.nestadoestructurasucursal,
  sirecc.nsucursal,
  sirecc.destructurasucursal,
  sirecc.destructurasucursalclasificador,
  sirecc.destructura
WHERE
  nestadoestructurasucursal.idnestadoestructurasucursal =
destructurasucursal.idnestadoestructurasucursal AND
  nsucursal.idnsucursal = destructurasucursal.idnsucursal AND
  destructurasucursalclasificador.inclasificadorsucursal =
nclasificadorsucursal.inclasificadorsucursal AND
  destructurasucursalclasificador.idestructura =
destructurasucursal.idestructura AND
  destructura.idestructura = destructurasucursal.idestructura;
```

Prueba 3

```
SELECT
  nmoneda.nombre,
  nbanco.nombre,
  dcuentabancaria.numerocuenta,
```

```
  destructurabanco.sucursal,
  destructura.nombre
FROM
  sirecc.destructura,
  sirecc.destructurabanco,
  sirecc.nbanco,
  sirecc.dcuentabancaria,
  sirecc.nmoneda
WHERE
  destructurabanco.idestructura = destructura.idestructura AND
  nbanco.idnbanco = destructurabanco.idnbanco AND
  dcuentabancaria.idestructurabanco = destructurabanco.idestructura AND
  nmoneda.idnmoneda = dcuentabancaria.idnmoneda;
```

Prueba 4

```
SELECT
  dpersona.primernombre,
  dpersona.primerapellido,
  dpersona.segundoapellido,
  dpersona.segundonombre,
  ddocumento.fecha,
  ddocumento.iddocumento,
  destructura.nombre,
  destructuradocumento.proceso,
  ddireccion.nombre,
  ddireccion.comentario
FROM
  sirecc.dpersona,
  sirecc.dpersonadireccion,
  sirecc.destructura,
  sirecc.ddireccion,
  sirecc.destructurapersona,
  sirecc.ddocumento,
  sirecc.destructuradocumento
WHERE
  dpersona.idpersona = dpersonadireccion.idpersona AND
  dpersonadireccion.iddireccion = ddireccion.iddireccion AND
  destructura.idestructura = destructurapersona.idestructura AND
  destructurapersona.idpersona = dpersonadireccion.idpersona AND
  ddocumento.iddocumento = destructuradocumento.iddocumento AND
  destructuradocumento.idestructura = destructura.idestructura;
```

Prueba 5

```
SELECT
  ndocumento.nombre,
  ddocumento.iddocumento,
  ddocumento.fecha,
  destructura.nombre,
  nestructura.nombre,
  ddocumentocomposicion.iddocumentopadre,
  ddocumentocomposicion.iddocumentohijo
FROM
  sirecc.ddocumento,
  sirecc.destructuradocumento,
  sirecc.ndocumento,
  sirecc.destructura,
  sirecc.nestructura,
  sirecc.ddocumentocomposicion
WHERE
  ddocumento.iddocumento = destructuradocumento.iddocumento AND
  ndocumento.idndocumento = ddocumento.idndocumento AND
  destructura.idestructura = destructuradocumento.idestructura AND
  nestructura.idnestructura = destructura.idnestructura AND
  ddocumentocomposicion.iddocumentopadre = ddocumento.iddocumento;
```