Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Diseño de la Base de Datos para el Sistema de Integración Bancaria.

Autor: Alexander Pérez Gamboa

Tutores: Ing. Juan Carlos Suarez López

Ing. Abdiel Carrazana Saucedo

La Habana, Cuba
2012

Declaración de autoría

Declaro que yo, Alexander Pérez Gamboa, soy el único autor del trabajo titulado: Diseño de la Base de Datos, para el Sistema de Integración Bancaria y autorizó a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmo el presente, a los ____ días del mes de diciembre del año 2012.

_____	_____
Firma del autor	Firma del Tutor
Alexander Pérez Gamboa	Juan Carlos Suarez López
	Abdiel Carrazana Saucedo

Resumen

El presente trabajo de diploma se centra en el diseño de una Base de Datos (BD), con el objetivo de garantizar la gestión de los procesos de pago del Servicio Administrativo de Identificación, Migración y Extranjería (SAIME) a través del Sistema de Integración Bancaria (SIB). Entre los aspectos a tratar en este estudio están: pasos y principios de diseño a seguir, patrones a utilizar, restricciones del modelo, trazabilidad de la BD, nomenclatura, indexación, seguridad, validación del diseño a través de las pruebas de integridad al modelo, entre otros. También se aborda sobre las herramientas que intervienen en la confección y acceso a datos de dicha BD, además de herramientas de replicación y pruebas, como parte sugerente de buenas prácticas de optimización y validación, en aras de mejorar la disponibilidad de los datos y el rendimiento en el trabajo con la misma, garantizando así su eficiencia. Como resultado de la investigación y del trabajo se obtiene un modelo de datos normalizado, una BD optimizada y validada teórica y funcionalmente.

Palabras claves: base de datos, diseño, optimización.

Índice

Introducción

El auge de las tecnologías y en especial su impacto en la vida del hombre y de la sociedad en general, han sido y son, muy significativos. La computadora se ha convertido en una poderosa herramienta que fomenta el desarrollo que vive el mundo hoy en día, debido a la necesidad de automatizar las actividades que realiza el hombre, tratando de ganar en calidad y rapidez de manera progresiva.

Cuba, a pesar de sus condiciones económicas y su lucha por desarrollarse en un mundo globalizado y en crisis, hace un gran esfuerzo por dar pasos de avance en aras de la informatización de la sociedad y la automatización de las actividades industriales y empresariales; para lograr esto se preparan profesionales a lo largo y ancho del país. La Universidad de las Ciencias Informáticas (UCI), surgida al calor de la batalla de ideas, constituye un ejemplo de estos avances; la misma tiene un papel protagónico en la creación de proyectos informáticos que se despliegan en varios países, como por ejemplo: en el SAIME de la República Bolivariana de Venezuela.

El SAIME, es un organismo adscrito al Ministerio del Interior para las Relaciones Interiores y de Justicia en la República Bolivariana de Venezuela, cuyo objetivo principal es brindar servicios relacionados con la identificación de las personas naturales que habitan en el territorio nacional, entre otros servicios a personal extranjero, de acuerdo a lo establecido con las leyes venezolanas. Ejemplos de estos servicios que hoy en día se encuentran informatizados, lo constituyen la emisión de pasaportes electrónicos para personal nacional, el servicio de extranjería para extranjeros y para ambos tipos de personal los servicios de cedulación y de migración, funcionando este último en los puertos, aeropuertos y puntos fronterizos, gestionando así las entradas y salidas del territorio venezolano.

La gestión de estos servicios se realiza a través de trámites que generan pagos por concepto de prestación de los mismos. Dichos pagos eran manejados primeramente mediante estampillas que respaldaban el valor en unidades tributarias, las cuales eran y son el costo para cada servicio. Estas estampillas eran un sello que el cliente compraba en puntos de ventas y se pegaban posteriormente a la solicitud de servicio en las oficinas del SAIME. Una vez efectuado el trámite, la evidencia del pago quedaba registrada mediante la colocación de la estampilla al dorso de la planilla de control del servicio prestado. Pero la sustracción de estas, de los documentos de solicitud, para luego revenderlas nuevamente en la calle y la falta de

certeza de que el comprador de la estampilla era el mismo que hacia la solicitud del servicio, conllevó a tomar otra medida para la integración del SAIME con los bancos recaudadores.

Así surge el *voucher*, comprobante oficial que registra una serie de datos como ráfaga bancaria, nombres y apellidos del depositante y del beneficiario y otra serie de datos. La utilización del *voucher* constituyó un paso de avance de dicha integración, pero dicho método no está exento de malversaciones hoy en día, ya que muchos de estos comprobantes han sido falsificados y reutilizados, y lo que más se ha podido evitar, ajustando el sistema SAIME, es la reutilización de los mismos.

En general, estos procesos de pagos siguen teniendo irregularidades y aún se realizan de forma manual, lo cual provoca un trabajo engorroso, ineficiente, inseguro y costoso en tiempo y recursos. No se logra una definitiva y firme integración entre el SAIME y los bancos, continúan las falsificaciones de documentos legales, el robo, la pérdida y el deterioro de archivos, de información confidencial, entre otros. Por tal motivo el SAIME requiere de una solución inmediata a los problemas antes mencionados.

Una primera parte de la solución a la problemática antes mencionada lo representa el Sistema de Integración Bancaria (SIB), el cual será desarrollado por un equipo de proyecto del Centro de Identificación y Seguridad Digital (CISED) de la UCI, como parte del Proyecto Identidad Fase II firmado en la VII Comisión Mixta del Convenio Cuba-Venezuela. Pero debido a que la aplicación trabajará con una gran cantidad de información sensible, lo primordial será garantizar, buenas condiciones de gestión y almacenamiento de la información que arrojan los procesos antes mencionados.

Lo cual conlleva al **problema de la investigación**: ¿Cómo garantizar la persistencia y seguridad de la información manejada por el sistema de gestión de integración del SAIME y los bancos?

Con vista a la solución del problema se plantea como **objeto de estudio**: Diseño de Base de Batos para sistemas de gestión, con el **objetivo general**: Diseñar la Base de Datos del Sistema de Integración Bancaria, para gestionar los cobros abonados al SAIME por concepto de prestación de servicios, garantizando la seguridad y persistencia de la información.

Para complementar el objetivo de la investigación y solucionar la problemática planteada se proponen las siguientes **tareas investigativas**:

- Análisis del estado actual de los tipos de Bases de Datos para escoger la que más se adecua al problema en cuestión.

- Definición del Sistema de Gestión de Base de Datos (SGBD) a utilizar para el sistema.
- Identificación de las herramientas de modelado y de administración de BD a utilizar, para definir los medios de trabajo sobre la BD.
- Investigación de los procesos involucrados con los trámites de gestión de servicios y de los cobros de estos para mejor entendimiento del negocio.
- Identificación de los requisitos no funcionales para la persistencia de los datos del SIB.
- Identificación de los elementos que tributan a las entidades del modelo de datos de los módulos del SIB para la creación del modelo conceptual.
- Modelado de los conceptos del negocio de los módulos del SIB para creación del modelo lógico.
- Caracterización y aplicación de los aspectos importantes de la BD como son: la nomenclatura, los patrones de diseño, trazabilidad, seguridad y normalización, para realizar un buen diseño.
- Recomendación y aplicación de buenas prácticas de optimización para mejorar la disponibilidad de la información y la eficiencia de los procesos del SIB.
- Aplicación de pruebas para validar el diseño y funcionamiento de la BD.

Derivando como **campo de acción**: Diseño de Base de Datos para gestionar transacciones de pagos a servicios.

Los **métodos de investigación** utilizados se desglosan a continuación:

Teóricos:

Análisis Histórico - Lógico: se analizó la evolución del diseño de las Bases de Datos en la Universidad, las técnicas, sugerencias y soluciones ya desarrolladas por otros, para poder desarrollar una BD estable y segura.

Modelación: se realizaron modelos para establecer un diseño final, lógico y físico, de la BD sobre la cual trabajará la aplicación.

Empíricos:

La entrevista: se realizaron varias entrevistas con personal de experiencia, con el fin de lograr definir un buen diseño.

Esta investigación estará estructurada en 3 capítulos, donde se describe el trabajo realizado y los resultados obtenidos del mismo.

Capítulo 1: Fundamentación Teórica: en este capítulo se realiza un análisis sobre las formas de almacenamiento de la información que se gestiona en una aplicación web, además se aborda el origen, desarrollo, tipos, modelos de las BD, los gestores, entre otros aspectos importantes, todo esto con el fin de alcanzar una base de conocimiento conceptual del tema a tratar.

Capítulo 2: Descripción y análisis de la solución propuesta: en este capítulo se describe la arquitectura del sistema en el cual estará montada la BD del SIB, así como los procesos que ocurrirán en él. Se identificarán los conceptos en los requisitos funcionales que tributarán a posibles entidades para la conformación del modelo de datos. Se modelará la BD del sistema a partir de los conceptos identificados y las relaciones entre ellos, aplicándose patrones de diseño de bases de datos. Se normalizará el modelo y se definirá los índices, las restricciones del modelo de datos, la trazabilidad y la seguridad en el diseño.

Capítulo 3: Validación y optimización de la Base de Datos: en este capítulo se ofrece información acerca de la validación teórica así como funcional, realizada a la propuesta, con el objetivo de evaluar su diseño y comportamiento al ser gestionado mediante consultas. Se llevan a cabo pruebas de integridad y de rendimiento, así como también, se aborda más sobre los temas de seguridad y optimización con que contará la BD.

Capítulo I: Fundamentación Teórica

Introducción

En un inicio, el surgimiento y desarrollo del SIB sería la solución para romper con la manualidad y el trabajo engorroso de los procesos de pagos de servicios que el SAIME necesita automatizar. Pero la falta de persistencia de la información no se solucionaría sin la existencia de un medio de almacenamiento físicamente establecido; uno que le permitiera al usuario gestionar sus datos de manera rápida y segura. Por lo cual se hace urgente realizar un estudio del medio de almacenamiento más idóneo para satisfacer las necesidades del sistema de integración en cuestión.

1.1 Formas de almacenamiento de la información

El Sistema de Integración Bancaria (SIB), es una aplicación web que debe responder a los requisitos de automatización que exigen los procesos de pagos de servicios, pero también debe estar preparado para guardar de manera segura y confiable la información referente a las transacciones que se realizan en los mismos. Por lo cual se hace indispensable investigar cuáles son las formas de almacenamiento de datos que se pueden utilizar.

Entre las formas de almacenamiento que existen, están:

Base de Datos: es una colección integrada de datos almacenados en diferentes tipos de registros. Los registros se interrelacionan por medio de relaciones propias y lógicas de los datos y no mediante su ubicación física en el almacenamiento. Pueden ser creadas para soportar gran cantidad de datos de forma permanente (Pérez, 2005).

Ficheros: en informática se denomina fichero a todo el conjunto de información (programas o datos) que el ordenador almacena en un disco o cinta de manera diferenciada. Todos los ficheros se identifican, y así se diferencian los unos de los otros, por un nombre y, opcionalmente, por una extensión (Graells, 2010). Ejemplo de estos tenemos los ficheros con extensión: HTML, ASP, DOC, MP3, entre muchos otros.

Estas dos formas constituyen dos grandes familias que tienen sus clasificaciones y sus características específicas, y son las conocidas formas de almacenamientos de la información con que se trabaja en la actualidad.

Otros contenedores de almacenado de datos los constituyen los XML, por sus siglas en inglés, *eXtensible Markup Language* o en español, el Lenguaje de Etiquetado Extensible. XML es un archivo o fichero como los antes mencionados, que fue propuesto como estándar para el

intercambio de información estructurada entre diferentes plataformas. Permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información (Silberschatz, 2002).

De acuerdo con los tipos de almacenamiento antes analizados, se propone como solución para nuestro problema y se adopta también como política del proyecto, el estudio de las bases de datos.

1.2 Base de Datos

Para profundizar y entender mejor lo que es una BD es necesario conocer antes, el significado de conceptos importantes como los que se presentan a continuación:

Dato: es la unidad mínima de información que por sí sola carece de significado (Obregón, 2008).

Campo: es una pieza única de información (masadelante, 2012).

Registro: también llamado fila o dupla, representa un objeto único de datos implícitamente estructurado en una tabla. Es un conjunto de campos que contienen los datos que pertenecen a una misma repetición de entidad (buenastareas, 2012).

Tabla: colección de registros de un mismo tema (duiops, 2009).

Agrupando estos conceptos se conforma de manera sencilla la siguiente definición:

Base de datos: son un conjunto de tablas o archivos en su mayoría de diferentes temas, que se relacionan entre sí. Donde cada una de estas, esta constituidas por una serie de registros de un mismo tipo. Y cada uno de estos, están formado por uno o más campos, que son los que albergan los atributos o datos de cierta entidad.

1.3 Surgimiento de las bases de datos

Las BD tuvieron sus orígenes a principios de los años sesenta debido a la necesidad de almacenar la información que se gestionaba en la memoria de la máquina. Surgen con el objetivo de guardar grandes cantidades de datos y poder gestionarlos cada vez con mayor facilidad. Las primeras de estas se gestionaban en ficheros que eran almacenados en tarjetas o soportes magnéticos. Los discos dieron inicio a las bases de datos de red y jerárquicas, pues los programadores con su habilidad de manipulación de estructuras junto con las ventajas de los discos era posible guardar estructuras de datos como listas y árboles (Rodríguez, 2009).

En el año 1970 se convoca a una Conferencia de Lenguajes de Programación en Estados Unidos, convocada por IBM (es una empresa multinacional estadounidense que fabrica y comercializa *hardware* y *software* para computadoras), donde se establece un modelo llamado *CODASYL* (Modelo para el tratamiento de bases de datos que fue publicado por *E. Codd*). *Codd* propuso una forma de organizar las bases de datos mediante un modelo matemático lógico, a raíz de esto se estableció un modelo estándar para el desarrollo de las bases de datos (los modelos de BD relacionales) y surge años más tarde el lenguaje SQL, siendo este el más usado actualmente por los desarrolladores de *software* en sus respectivas comunidades de desarrollo (González, 2009). A partir de los aportes de *Codd* el multimillonario *Larry Ellison* desarrollo la BD *Oracle*, el cual es un SGBD, que se destaca por soportar transacciones, estabilidad, escalabilidad y multiplataforma. Inicialmente no se uso el modelo relacional debido a que tenía inconvenientes por el rendimiento, ya que no podían ser competitivas con las bases de datos jerárquicas y de red. Esta tendencia cambió por un proyecto de IBM el cual desarrolló técnicas para la construcción de un sistema de bases de datos relacionales eficientes, llamado *System R* (Rodríguez, 2009).

En la década de 1980, las bases de datos relacionales con su sistema de tablas, filas y columnas, pudieron competir con las bases de datos jerárquicas y de red, ya que su nivel de programación era bajo y su uso muy sencillo. En esta década el modelo relacional ha conseguido posicionarse del mercado de las bases de datos. Y también en este tiempo se iniciaron grandes investigaciones paralelas y distribuidas, como las bases de datos orientadas a objetos.

A principios de los noventa, se crea para la toma de decisiones ante las bases de datos, el lenguaje SQL, que es un lenguaje programado para consultas. Con él se puede analizar grandes cantidades de información el cual permite especificar diversos tipos de operaciones frente a la misma información, a diferencia de las bases de datos de los ochenta que eran diseñadas para las aplicaciones de procesamiento de transacciones. Los grandes distribuidores de bases de datos incursionaron con la venta de bases de datos orientada a objetos. A finales de esta década aparece *Word Wide Web*, medio por el cual se facilitaba la consulta de las bases de datos. Actualmente tienen una amplia capacidad de almacenamiento de información, también una de las ventajas es el servicio de siete días a la semana y las veinticuatro horas del día, sin interrupciones a menos que haya planificaciones de mantenimiento de las plataformas o el *software*.

En la actualidad, existe gran cantidad de alternativas en línea que permiten hacer búsquedas orientadas a necesidades específicas de los usuarios, una de las tendencias más amplias son las bases de datos que cumplan con el protocolo *Open Archives Initiative – Protocol for Metadata Harvesting* (OAI-PMH) los cuales admiten el almacenamiento de gran cantidad de artículos que permiten una mayor visibilidad y acceso en el ámbito científico y general (Rodríguez, 2009).

1.3.1 Clasificaciones de las Bases de Datos

- ✓ Según la movilidad de sus datos las bases de datos se clasifican como:
  - ***Base de Datos Estáticas:*** los datos no varían, solamente son consultados. Son utilizadas principalmente para almacenar datos históricos que en un momento dado servirán para la toma de decisiones.
  - ***Base de Datos Dinámicas:*** los datos cambian con el tiempo según las operaciones que se realizan (Insertar, Modificar y Eliminar).
- ✓ Según su contenido:
  - ***Base de Datos Bibliográficas:*** sólo contienen un representante de la fuente primaria que permite localizarla. Un registro típico de una BD bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición de una determinada publicación, entre otros. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo.
  - ***Bases de Datos de texto completo:*** están constituidas por los propios documentos en formato electrónico, volcado su texto de forma completa. Pueden incorporar, además, campos en los que se contiene la información fundamental para facilitar su descripción y recuperación. En estos sistemas la operación de búsqueda (que puede abarcar la totalidad del texto) y la consulta del documento se producen sin salir del propio sistema de información (Cidran, 2010).
  - ***Directorios:*** proporcionan información sobre individuos, organizaciones y servicios, nombres, direcciones entre otras cosas. Un ejemplo son las guías telefónicas en formato electrónico (González, 2009).

1.3.2 Diseño de las Bases de Datos

Un diseño correcto de una BD permite tener acceso a información exacta y actualizada, y a la vez es esencial para maximizar las ventajas de la misma como es el aumento de la seguridad y la consistencia de los datos. Es indispensable emplear todo el tiempo que se pueda para

aprender los principios y fases de un buen diseño, ya que la puesta en práctica de estos, guía a los diseñadores de BD hacia el objetivo de crear una BD optimizada y adaptada a las condiciones y exigencias de los proyectos para los cuales trabajan.

A la hora de realizar el diseño de una BD se plantean varios **principios**:

- ✓ Fidelidad: se debe crear siempre un modelo que satisfaga las necesidades del problema. No se tiene un modelo correcto si no cumple con la realidad que se pretende representar.
- ✓ Los datos no deben ser redundantes: malgastan el espacio en la BD y aumentan la probabilidad de que se produzcan errores e incoherencias.
- ✓ Simplicidad: siempre hay que procurar hacer el modelo tan simple como sea posible de manera que sea fácil de entender, fácil de extender y fácil de implementar.
- ✓ La información debe de estar correcta y completa.
- ✓ Relaciones n-arias: aun cuando se pueden presentar casos en los que una relación terciaria o n-aria parezca más conveniente, es mejor siempre pensar en términos de relaciones binarias únicamente. En el peor de los casos de que exista una relación n-aria forzosa, lo que se debe hacer es convertir esa relación R en entidad E y corregir todas las relaciones que tenía R, de manera que ahora esa nueva entidad se relacione con todas las entidades que se involucraban con R (Aguilar, 2011).

Objetivos del diseño de base de datos

- ✓ Un rápido y eficiente acceso a la información con la menor redundancia.
- ✓ Diseño de esquemas en la forma normal (FN).
- ✓ Información adicional[1].
- ✓ Especificación de limitantes (Vargas, 2005).

Un buen diseño de BD es por tanto, aquel que:

- ✓ Divide la información en tablas basadas en temas para reducir los datos redundantes.
- ✓ Proporciona los medios necesarios para reunir la información de las tablas cuando así se precise.
- ✓ Ayuda a garantizar la exactitud e integridad de la información.
- ✓ Satisface las necesidades de procesamiento de los datos y de generación de informes.

[1] Para determinar si un esquema de relaciones tiene una de las formas normales se requiere mayor información sobre la empresa del "mundo real" (negocio) que se intenta modelar con la base de datos. La información adicional la proporciona una serie de limitantes que se denominan *dependencias de los datos* (MultiMania).

El proceso de diseño consta de los pasos siguientes:

- ✓ Determinar la finalidad de la base de datos: permite estar preparado para los demás pasos.
- ✓ Buscar y organizar la información necesaria: reúne todos los tipos de información que desee registrar en la BD, como los nombres de productos o los números de pedidos.
- ✓ Dividir la información en tablas: divide los elementos de información en entidades o temas principales. Cada tema pasará a ser una tabla.
- ✓ Convertir los elementos de información en columnas: decide qué información desea almacenar en cada tabla. Cada elemento se debe de convertir en un campo y se mostrará como una columna en la tabla. Por ejemplo, una tabla Empleados podría incluir campos como Apellido y Fecha de contratación.
- ✓ Especificar claves principales: se debe de especificar la clave principal de cada tabla. La llave principal es una columna que se utiliza para identificar inequívocamente cada fila, como id. del producto o id. del pedido.
- ✓ Definir relaciones entre las tablas: permite examinar cada tabla y posibilita decidir cómo se relacionarán los datos de una tabla con las demás tablas. Con esto se añaden nuevos campos a las tablas o crea nuevas tablas para refinar las relaciones según sea necesario.
- ✓ Ajustar el diseño: se debe de analizar el diseño para detectar errores. Crear las tablas y agregar algunos registros con datos de ejemplo. Comprobar si se puede obtener los resultados previstos de las tablas. Y realizar los ajustes necesarios en el diseño.
- ✓ Aplicar las reglas de normalización: aplicar reglas de normalización de los datos para comprobar si las tablas están estructuradas correctamente. Realizar los ajustes necesarios en las tablas" (Microsoft , 2007).

El diseño de una BD está compuesto por 6 **fases**:

Recopilación y Análisis de Requisitos*:* En esta fase los diseñadores entrevistan a los futuros usuarios de la aplicación para conocer sus expectativas y, recoger y documentar sus necesidades de información. Paralelamente, conviene definir los requisitos funcionales que consisten en operaciones (transacciones) que se aplicarán a la BD e incluyen la obtención de datos y la actualización. También se identifican los grupos de usuarios reales y posibles, las áreas de aplicación y se revisa la documentación existente (González, 2011).

***Diseño Conceptual**:* Permite descubrir el significado de los datos con los que se irán a trabajar, tiene como objetivos comprender:

- La perspectiva que tiene cada usuario acerca de los datos.
- La naturaleza de los datos, independientemente de su representación física.
- El uso de los datos a través de las áreas de aplicación.

Le permite al diseñador de bases de datos transmitir a la entidad los conocimientos adquiridos sobre la información que ella maneja. Este esquema se construye utilizando la información que se encuentra en la especificación de requisitos de usuarios. Es independiente, completamente de los aspectos de la implementación. El esquema conceptual es una fuente de información para el diseño lógico de la BD.

***Elección de un SGBD**:* Para el cumplimiento de esta fase se consideran diferentes factores técnicos, económicos y de beneficio, de servicio técnico y formación de usuarios, organizativos de rendimiento, entre otros. Sin embargo, resulta difícil la medida y la forma de examinar y considerar los diferentes factores (González, 2011).

***Diseño Lógico**:* permite al diseñador construir un esquema a partir de la información de la empresa, basándose en un modelo de BD específicos. Es capaz de llevar del modelo conceptual al modelo lógico. Es fuente de información del modelo físico y juega un papel fundamental en el diseño y desarrollo de la BD. Permite que los posibles cambios que se realicen sobre los programas de aplicación o sobre los datos se representen correctamente en la base de datos. En esta parte del diseño se aplica la técnica de la normalización de BD para comprobar la validez de los esquemas lógicos del modelo relacional, ya que asegura que los datos en las tablas no contengan datos redundantes. Tanto el esquema conceptual como el lógico son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Son dos etapas claves para conseguir que un sistema funcione correctamente.

***Diseño Físico**:* es el proceso de producir la descripción de la implementación de la BD en memoria secundaria: estructuras de almacenamiento y métodos que garanticen un acceso eficiente a los datos. Entre el diseño lógico y el físico hay una retroalimentación, puesto que, algunas de las decisiones que se tomen durante el diseño lógico para mejorar las prestaciones, pueden afectar a la estructura del esquema físico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones entre tablas y las restricciones que se deben cumplir sobre ellas.
- Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.
- Diseñar el modelo de seguridad del sistema (Microsoft , 2007).

***Implementación del sistema de BD**:* En esta fase final se hace realidad la base de datos, mediante la creación y la compilación de uno o varios esquemas y ficheros de bases de datos, así como de las transacciones a través de las aplicaciones.

La metodología expuesta, que puede servir como marco de referencia general, puede modificarse y adaptarse según las características del contexto en el que se diseña y despliega el sistema de bases de datos (González, 2011).

Tipos de modelos conceptuales

Las BD deben estar respaldadas por un diseño que garantice una buena representación y fácil manipulación de los datos, es por ello que han surgido diferentes modelos de diseño. Un modelo de datos es una abstracción de un conjunto de herramientas conceptuales para describir datos, sus relaciones, su significado y sus restricciones de consistencia. Existen diferentes modelos de datos y su utilización tiene lugar durante el proceso de diseño de una BD. Los modelos conceptuales permiten crear una representación de la realidad con un elevado nivel de comprensión, sin tener en cuenta la estructura de almacenamiento de los datos; mientras que en los modelos lógicos la estructura de los datos está muy ligada a la estructura de almacenamiento y al SGBD que se utilice. Inicialmente se utiliza el modelo conceptual para modelar una representación de la realidad y el que luego transformado en un modelo lógico. Existen distintos tipos de modelos conceptuales, como por ejemplo:

Basados en registros:

- Jerárquico: datos en registros, relacionados con apuntadores y organizados como colecciones de árboles.
- Redes: datos en registros relacionados por apuntadores y organizados en gráficas arbitrarias.
- Relacional: datos en tablas relacionados por el contenido de ciertas columnas.

Basados en objetos:

- Entidad-relación: datos organizados en conjuntos interrelacionados de objetos (entidades) con atributos asociados.
- Orientado a objetos: datos como instancias de objetos (incluyendo sus métodos) (Aguilar, 2011).

A continuación, una descripción más detallada de 3 modelos tomados como propuestas:

Bases de Datos relacionales: es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de "relaciones". En este modelo tanto las entidades como las relaciones que se establecen, se representan a través de tablas, las cuales pueden o no variar en el transcurso del tiempo en cuanto volumen de información y no en estructura si son bien diseñadas. Estas tablas poseen un nombre único que las identifican y están compuestas por filas o tuplas que representan la ocurrencia de objetos almacenados en una relación. También las componen columnas o atributos que no son más que los valores pertenecientes a un dominio que puede tomar un objeto. Las tuplas se identifican mediante una llave o clave primaria, lo que las hace únicas dentro de la relación. En una relación no puede existir más de una columna con el mismo nombre, no existen dos tuplas iguales y el orden tanto de las columnas como de las tuplas no es significativo. Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información. Un ejemplo se muestra en el Anexo 1 (Hoyo, 2011).

Modelo entidad-relación (Modelo ER): este modelo está basado en una percepción del mundo real, que consta de un conjunto de objetos básicos llamados entidades y de las interrelaciones que existen entre estos objetos. Usa diagramas para representar la estructura natural de los datos, donde los rectángulos representan las entidades, las cuales están identificadas por atributos o propiedades, que son la unidad menor de información de los objetos que figuran. Y los rombos representan las interrelaciones, que son enlazadas con sus entidades, formando arcos, donde el grado de la interrelación es el indicado en el arco. Estas relaciones pueden ser de uno a uno (1:1) cuando una entidad A solo tiene una referencia de un objeto perteneciente a una entidad B, de uno a muchos (1: n) y de muchos a muchos (n: m); también se les conoce como cardinalidad (robealex1, 2011).

Modelo Orientado a Objetos: trata de almacenar en la BD los objetos completos (estado y comportamiento). Una BD orientada a objetos es la que incorpora todos los conceptos importantes del paradigma de objetos:

- Encapsulación: cada objeto es consciente de sus propias características. Es decir que permite ocultar la información de las tablas al resto de los objetos.
- Herencia: para facilitar la programación, se puede establecer toda una jerarquía de tipos o clases. Se evita la declaración de datos de forma redundante.
- Polimorfismo: propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Esto podría denominarse independencia entre programas y operaciones. Es un modelo muy complejo y aprenderlo a desarrollar puede traer consigo problemas tales como atraso en la modelación y del diseño de la BD en general (Torre, 2010).

Haciendo una comparación de todas las características de estos modelos, se determina que el modelo relacional es el que más se acopla al problema a resolver, al entorno a trabajar y a la aplicación web en cuestión, a pesar de que el modelo orientado a objeto es más novedoso y su modelación representa de manera más completa el mundo real. Lo anterior queda justificado por los aspectos siguientes:

- ✓ El modelo relacional es el más utilizado en la actualidad para modelar problemas reales, así como para diseñar y desarrollar las soluciones propuestas a los mismos.
- ✓ Permite administrar datos dinámicamente.
- ✓ Aplica en su diseño un proceso de normalización de los datos que evita la redundancia y mantiene la integridad de la información.
- ✓ De fácil entendimiento hacia los usuarios, cualquier persona observando su diseño puede comprenderlo, aprenderlo y aplicarlo.

1.3.3 Uso del Modelo entidad-relación

Independientemente de la selección del modelo relacional, todo diseño de BD comienza con el uso del modelo entidad-relación o modelo E/R. Esto se hace imperioso debido a que el diseñador de BD, debe interpretar los procesos del negocio y realizar una colección previa de objetos básicos, llamados entidades, así como las relaciones entre ellos. A esta colección se le denomina diagrama, que junto con una lista de atributos y una descripción de otras

restricciones que no se pueden reflejar en los mismos, completan el modelo. El modelado de datos no acaba con el uso de esta técnica. Son necesarias otras técnicas para lograr que un modelo se pueda transformar directamente en una base de datos. Brevemente:

- Transformación de relaciones múltiples en binarias.
- Normalización de las BD relacionales (algunas relaciones pueden transformarse en atributos y viceversa).
- Conversión en tablas en caso de utilizar una BD relacional (Figueroa, 2010).

El modelo entidad-relación tiene en cuenta una serie de aspectos importantes en su desarrollo como por ejemplo: la herencia, la agregación, atributos en relaciones, llaves primarias, entre otras características de dicho modelo. Pero sin duda el aspecto de la cardinalidad en las relaciones es uno de los más notorios, ya que influye en la transformación del modelo entidad-relación al modelo relacional. Dado que una de las técnicas de este modelo es la transformación de relaciones múltiples en binarias, se presentan los siguientes ejemplos, con un breve estudio de lo que es cardinalidad para luego comprender los pasos que siguen en la transformación al modelo relacional. Dado un conjunto de relaciones binarias y los conjuntos de entidades A y B, la correspondencia de cardinalidades puede ser:

- Uno a Uno: Una entidad de A se relaciona únicamente con una entidad en B y viceversa (ejemplo relación vehículo - matrícula: cada vehículo tiene una única matrícula, y cada matrícula está asociada a un único vehículo).
- Uno a varios: Una entidad en A se relaciona con cero o muchas entidades en B. Pero una entidad en B se relaciona con una única entidad en A (ejemplo vendedor - ventas).
- Varios a Uno: Una entidad en A se relaciona exclusivamente con una entidad en B. Pero una entidad en B se puede relacionar con 0 o muchas entidades en A (ejemplo empleado-centro de trabajo).
- Varios a Varios: Una entidad en A se puede relacionar con 0 o muchas entidades en B y viceversa (ejemplo asociaciones- ciudadanos, donde muchos ciudadanos pueden pertenecer a una misma asociación, y cada ciudadano puede pertenecer a muchas asociaciones distintas).

La optimización de los diagramas teniendo en cuenta los aspectos antes mencionados, da paso a la transformación al modelo relacional rigiéndose por los siguientes pasos:

- Toda entidad se transforma en una tabla, todo atributo se transforma en una columna dentro de la tabla a la que pertenece y el identificador de la entidad se convierte en la clave primaria de la tabla.
- Toda relación de muchos a muchos (N: M) se convierte en una tabla que tendrá como clave primaria las dos claves primarias de las entidades que se asocian.
- En las relaciones de uno a mucho (1:N) la clave primaria de la entidad con cardinalidad 1 pasa a la tabla de la entidad cuya cardinalidad es N
- En las relaciones N: M existen tres posibilidades: Si la cardinalidad es (0,1) en ambas entidades, se crea tabla. Mientras que si la cardinalidad de una es (0,1) y de la otra es (1,1) se suele pasar la clave primaria de (1,1) a la de (0,1). Si la cardinalidad de ambas es (1,1) se pasa la clave de cualquiera de ellas a la otra (Alvarez, 2007).

Ejemplo de la trasformación que sufre un modelo de entidad-relación, se presenta a continuación en la Figura 1, dando paso a las siguientes tablas:

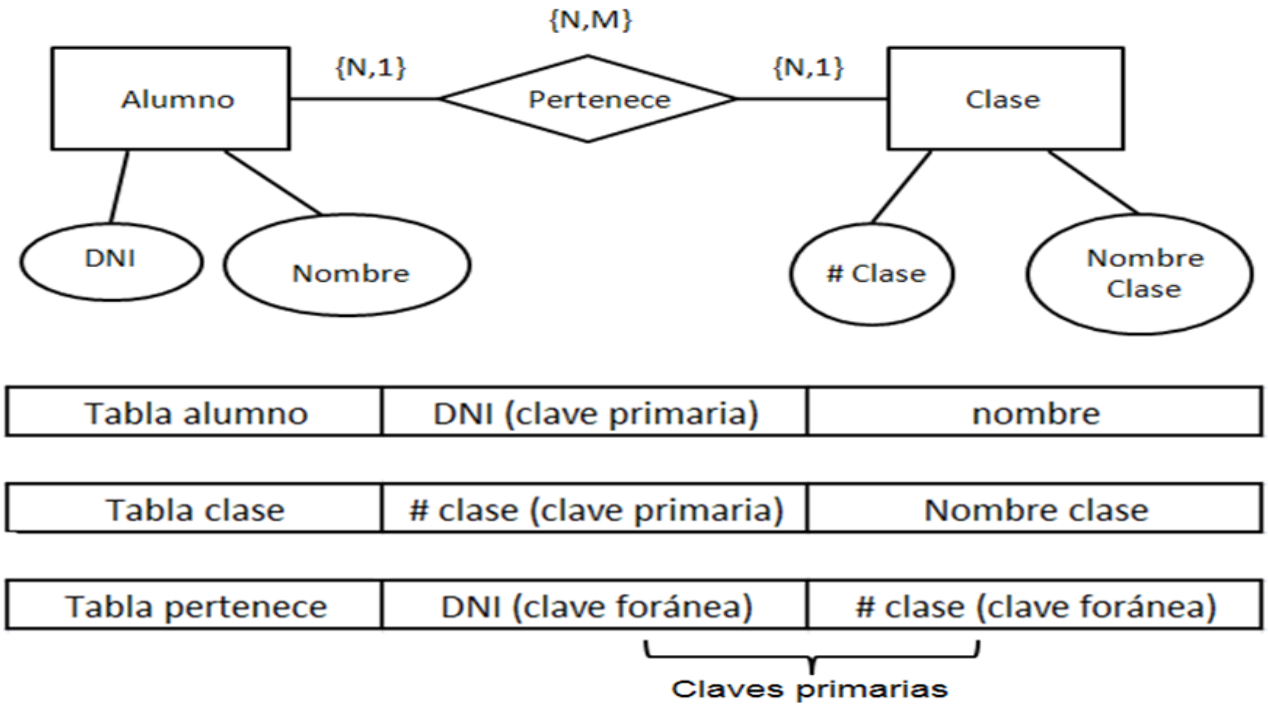

Figura 1: Paso del modelo E/R al modelo relacional.

1.3.4 Restricciones de un modelo de datos

El modelo relacional debe cumplir un grupo de restricciones o reglas con el objetivo de apoyar la integridad de la base de datos. Dos pasos importantes en el diseño de las tablas son la identificación de valores válidos para una columna y la determinación de cómo forzar la integridad de los datos en la columna. La integridad de datos puede verse en entidades, dominios, referencias o algunas definidas por el usuario:

- ✓ Integridad de entidad:

Define una fila como entidad única para una tabla determinada. La integridad de entidad exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones.

- ✓ Integridad de dominio:

Viene dada por la validez de las entradas para una columna determinada. La integridad de dominio se crea para restringir los tipos de datos, el intervalo de valores posibles o el formato mediante reglas y restricciones.

- ✓ Integridad referencial:

Protege las relaciones definidas entre las tablas cuando se crean o se eliminan filas. La integridad referencial garantiza que los valores de clave sean coherentes en las distintas tablas. Cuando se exige la integridad referencial, se impide a los usuarios:

- Agregar o cambiar filas en una tabla relacionada si no hay ninguna fila asociada en la tabla principal.
- Cambiar valores en una tabla principal que crea filas huérfanas en una tabla relacionada.
- Eliminar filas de una tabla principal cuando hay filas relacionadas coincidentes.

- ✓ Integridad definida por el usuario:

Permite definir reglas de empresas específicas que no pertenecen a ninguna otra categoría de integridad. Todas las categorías de integridad admiten la integridad definida por el usuario, esto incluye las restricciones a nivel de columna y nivel de tabla, procedimientos almacenados y desencadenadores (liz555, 2011).

1.4 Sistemas Gestores de Bases de Datos (SGBD)

Con la necesidad de lograr la gestión de las Bases de Datos surgen los Sistemas Gestores de Bases de Datos (SGBD), que según publicaciones, no es más que una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se

compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.
Los SGBD son un tipo de *software* muy específico y según el Dr. Rogelio S Silverio Castro: sus funciones principales son la creación, mantenimiento y eliminación de BD, 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 (González, 2009). Se compone de un lenguaje de definición de datos (DDL), de un lenguaje de manipulación de datos (DML) y de un lenguaje de consulta.

Hoy en día existe gran cantidad de SGBD, aunque con diferencias de funciones, expresan un mismo objetivo. Se analizaron tres sistemas propuestos: ***MySQL, Oracle*** y ***PostgreSQL***, con los cuales se estableció una comparación como se refleja en la Tabla 1, con el objetivo de seleccionar el gestor que más se adecua al entorno de trabajo que se quiere desarrollar. En la misma se registran una serie de características específicas, que ayudarán con la selección. Atendiendo a la comparación, la cual refleja ventajas y desventajas de dichos sistemas según las características propuestas, el orden para la selección quedaría regido por *Oracle* como primera opción, debido a los excelentes resultados representados, *PostgreSQL* como segunda y *MySQL* como última. Pero debido a que este trabajo hace referencia al diseño de una BD, para un proyecto que no tiene grandes proporciones, como las que pudiera tener cualquier otro que trabaje con *Oracle* y a que se trabaja para un cliente ubicado en un país que aboga a través de la ley, el uso obligatorio de herramientas libres (Ley de la Soberanía Tecnológica, rige el uso obligatorio del *software* libre de la Rep. Bolivariana de Venezuela para todas las dependencias públicas de carácter oficial (Cooperativa Centro de Estudios para la Educación Popular, 2009)) se propone como herramienta más idónea, a *PostgreSQL*, ya que *MySQL*, que también es libre, está destinado para proyectos más pequeños que los que cubre el seleccionado y no está preparado para trabajar por defecto con la integridad referencial y las transacciones, dos pilares fundamentales en la gestión de los procesos de pagos en cuestión.

Es válido resaltar como punto importante de la decisión realizada, la preparación de los diseñadores respecto al trabajo con el gestor seleccionado. Para la gestión de la BD se utilizará la versión 8.4, la cual ha sido recientemente liberada. Teniendo a favor de la misma la disponibilidad de 293 funcionalidades nuevas o mejoradas para la administración, consulta y programación, que hacen que *PostgreSQL* sea más fácil que nunca y una mejor opción.

Tabla 1: Tabla comparativa de SGBD propuestos.

SGBD	Soporte S.O	Rendimiento	Libre	Robusto	Integridad Referencial
MySQL	bueno	bueno	si	regular	Depende
Oracle	muy bueno	muy bueno	no	si	si
PostgreSQL	bueno	bueno(casi como MySQL)	si	regular	si

SGBD	Proyectos	ACID	Consultas Paralelas	Transacciones
MySQL	Relativamente pequeños	Depende	no	Depende
Oracle	grandes	si	si	si
PostgreSQL	medianos	si	no	si

1.5 Optimización de base de datos

La optimización de BD es un aspecto de suma importancia en el desarrollo de un sistema de este tipo. Esta permite mejorar el rendimiento, la eficiencia y las funcionalidades de dichos sistemas en las condiciones más adecuadas de acuerdo a sus características y propósitos. La optimización de una BD depende esencialmente de un correcto diseño inicial, tanto lógico como físico. El objetivo de la optimización consiste en minimizar el tiempo de respuesta para cada petición y maximizar el rendimiento de todo el sistema disminuyendo el tráfico de red, el acceso a disco y el tiempo de CPU[2].

Los planteamientos sobre la optimización deben ser propuestos durante todo el ciclo de desarrollo, y no sólo una vez que el sistema ya está implementado. Esta es la mejor forma de lograr el afinamiento de las BD. Existen dos clasificaciones en las que se puede trabajar el afinamiento de una BD:

- Afinamiento Proactivo: destacado por ser el más efectivo y menos costoso, ya que es abordado como mayor profundidad en las etapas tempranas del diseño, los cual repercute considerablemente en el trabajo de los arquitectos y diseñadores del sistema, responsables, entre otras cosas, de aplicar técnicas para mejorar el rendimiento, así como de establecer qué configuración de hardware será la apropiada para responder y cumplir con los requisitos y expectativas.
- Afinamiento reactivo: se pone en aplicación para mejorar sistemas en explotación. Aunque no es tan efectivo como el anterior, no se puede prescindir de él ya que para que un sistema bien diseñado cuente con un buen rendimiento, debe contar con un

[2] CPU: Unidad Central de Procesamiento (Central Process Unit)

mecanismo que identifique y combate los riesgos a que se expone un sistema en explotación.

Por otra parte es importante conocer que un problema que pueda provocar un mal rendimiento puede ocurrir después de iniciar el sistema, por ello, para comenzar, el rendimiento debe ser monitoreado y evaluado regularmente después de comenzar la operación.

El proceso de optimización no debe comenzar una vez que los usuarios de un sistema plantean quejas e inconformidades sobre el rendimiento y los tiempos de respuesta del mismo, cuando esto sucede usualmente es demasiado tarde para implementar algunas de las estrategias de optimización más efectivas. En este caso, si no es posible rediseñar completamente la BD, entonces lo único que se puede hacer para mejorar el rendimiento del sistema es reasignar más memoria y mejorar los parámetros de entrada y salida de disco (Perdomo, 2009).

1.5.1 Técnicas de optimización

Diseño de bases de datos

Es relevante destacar que de la elaboración de un adecuado diseño depende la eficacia del cumplimiento de los objetivos definidos para la BD, razón por la cual es lógico e imperioso el empleo del tiempo que sea requerido para aprender y aplicar los principios y aspectos fundamentales de un correcto diseño.

Normalización de bases de datos

La normalización se ha desarrollado con el propósito de crear estructuras de datos eficientes que eviten las anomalías que ocurren durante la actualización de los datos. Este concepto fue introducido por Edgar F.Codd y no solo es aplicable a sistemas relacionales. Este proceso permite eliminar la información redundante en una BD e involucra varias fases, que al completarse, se dice que la relación está en una de las formas normales que se presenta a continuación:

- ✓ Primera Forma Normal (1FN): una relación se encuentra en 1FN cuando todos los atributos que componen las tuplas son atómicos.
- ✓ Segunda Forma Normal (2FN): una relación se encuentra en 2FN si, y solo si, se encuentra en 1FN y los atributos no llaves, son funcional y completamente dependientes de la clave primaria, por lo general compuesta. Una relación que posea una única clave primaria se encuentra en 2FN.

- ✓ Tercera Forma Normal (3FN): una relación se encuentra en 3FN si, y solo si, se encuentra en 2FN y no existe dependencia transitiva entre los atributos no llaves respecto a la llave primaria. Este procedimiento elimina la transitividad entre los atributos.
- ✓ Forma normal de *Boyce-Codd* (FNBC): "Una relación R está en FNBC si, y solo si, cada determinante[3] es una superllave[4] (candidata[5] o primaria)" (GARCIA, 2005).

Hoy en día la mayoría de los diseñadores normalizan los modelos hasta la 3FN debido a su fácil modo de aplicación. Sin embargo, las BD aún pueden ser vulnerables a algunas anomalías menos comunes, lo que hace que tenga que aplicarse una normalización más compleja y técnica, que puede terminar en complicados modelos de datos, difíciles de aplicar, de mantener y de usar.

Índices

Hacer que una consulta funcione correctamente no es lo mismo a que lo haga de la forma más rápida. Se puede acelerar las consultas afinando el servidor para que responda lo mejor posible, así como haciendo uso de índices de una manera inteligente.

En una BD, los índices son estructuras de datos que mejoran la velocidad de las operaciones en las tablas asociadas. Permiten búsquedas de información más rápidas al proporcionar un orden e indicador de búsquedas para las consultas y evitan que el gestor revise todos los datos disponibles para devolver el resultado. Son un grupo de datos que el SGBD asocia con una o varias columnas de la tabla en los que aparece la relación entre el contenido y el número de fila donde está ubicado.

La indexación, tanto de claves primarias como extranjeras, se puede obtener del modelo relacional:

- Las claves primarias identifican unívocamente a cada elemento de una tabla.
- Las claves extranjeras marcan las relaciones entre tablas.

Disponer de índices en los campos adecuados optimizará sus resultados. A continuación se ofrecen algunas de las mejores prácticas encontradas durante la investigación:

- Crear un índice sobre los campos que son utilizados en las búsquedas (los que aparecen en las cláusulas WHERE o JOIN) para mejorar una consulta (SELECT).

[3] "Un determinante es cualquier atributo o conjunto de atributos del cual depende funcional y completamente cualquier otro atributo". (GARCIA, 2005)

[4] Una superclave es un conjunto de atributos que permite identificar a cada tupla de forma única.

[5] Una llave candidata es aquel subconjunto, formado a partir de una superclave, que no puede ser superclave.

- Utilizar índices sobre campos con valores únicos. Los índices funcionan peor si el campo tiene valores duplicados.
- Tratar de que los índices sean cortos. Si indexa un campo de texto, evite hacerlo sobre campos de longitud variable, y acorte siempre el tamaño del índice a lo que considere más adecuado.
- Evitar la creación de índices innecesarios. Estos se actualizan con cada cambio en la tabla asociada y pueden ralentizar las modificaciones de la misma.

Cuando en una tabla no existen índices en los cuales auxiliarse, el gestor tendrá que leer todos los registros de la tabla de manera secuencial para resolver una consulta. Esto es comúnmente llamado un "escaneo completo de una tabla", y es muchas veces algo que se debe evitar pues trae consigo:

- **Sobrecarga de CPU**. El proceso de chequear cada uno de los registros en una tabla es insignificante cuando se tienen pocos datos, pero puede convertirse en un problema a medida que va aumentando la cantidad de registros en la tabla. Existe una relación proporcional entre el número de registros que tiene una tabla y la cantidad de tiempo que le toma a los gestores revisarla completamente.
- **Concurrencia**. Mientras un gestor de BD está leyendo los datos de una tabla, éste la bloquea, de tal manera que nadie más puede escribir en ella, aunque si pueden leerla. Ocurre de forma similar cuando se está actualizando o eliminando filas de una tabla, con la diferencia que aquí no se puede leer.
- **Sobrecarga de disco**. En una tabla muy grande, un escaneo completo consume una gran cantidad de entrada/salida en el disco. Esto puede calentar significativamente el servidor de bases de datos, especialmente si se tiene un disco IDE antiguo.

La operación de indexación, creada por el SGDB, ordena los registros de un archivo de datos de acuerdo con los campos utilizados como llave primaria e incrementa sensiblemente la velocidad de ejecución de algunas operaciones sobre el archivo de datos. Normalmente para cada archivo de datos debe existir un índice cuya llave de indexación sea idéntica a su llave primaria, el mismo es denominado índice primario. También existen los índices secundarios asociados a uno o varios atributos, diferentes a la clave primaria. Estos son utilizados para reducir el tiempo de localización de una determinada información dentro de un archivo o para clasificar los registros del archivo de acuerdo con el orden necesario para la obtención de la información deseada (González, 2011).

Optimización de consultas

El uso de procedimientos almacenados ayuda mucho a mejorar el rendimiento en el ciclo de vida de una consulta, ya que permite acortar la transferencia de datos a un mínimo cuando se tienen consultas muy grandes o largas que no se construyen de forma avanzada; así como cuando las consultas pasan directamente a ser ejecutadas sin ser analizadas por el servidor (Greg Smith, 2012).

Existen algunos tipos de optimización de consultas como son (González, 2011):

Optimización basada en reglas:

- Heurísticas generales basadas en la experiencia del administrador o de los diseñadores, en la que el programador puede especificar reglas que guíen al optimizador en su trabajo.

Optimización basada en costos:

- Estimación de los costos de realizar una operación física. Depende:
  - ✓ Tamaño de la estructura: estadísticas.
  - ✓ Tamaño de la memoria: depende de los procesos que se ejecuten simultáneamente.

Los optimizadores basados en costos asignan un costo (que intenta estimar el costo de la consulta en términos de operaciones de entrada-salida requeridas, requisitos de CPU y otros factores) a cada uno de esos planes y selecciona el de menor valor. El conjunto de planes de ejecución se forma examinando los posibles caminos de acceso, mediante índices o secuenciales o algoritmos de *"join"* (*sort-merge join*, *hash join*, bucles anidados). Este optimizador no puede ser accedido directamente por los usuarios, sino que, una vez enviadas las consultas al servidor, pasan primero por el analizador y entonces llegan al optimizador. La herramienta más usada para realizar la optimización a las consultas de una BD es el comando *SQL EXPLAIN ANALYZE*[6]. Este permite ver el registro de cada consulta SQL que es llevada a cabo y ver exactamente como el planificador de *PostgreSQL* procesa cada consulta.

Generalmente la mejor forma de optimizar las consultas es usar índices en columnas específicas y combinaciones de columnas que corresponden a las consultas más frecuentes. También debe tenerse en cuenta que incrementar el número de índices en una tabla incrementa el número de operaciones de escritura que se deben realizar para cada *INSERT* y *UPDATE*.

[6] EXPLAIN ANALYZE es un comando que se utiliza para chequear la exactitud del plan de ejecución, lo muestra de una forma más detallada, lo que el planificador Postgres genera para la consulta dada. Muestra los costos promedios para cada iteración y el costo total.

1.6 Optimización para PostgreSQL

En la actualidad, una de las tareas más importantes de la optimización se realiza en *PostgreSQL* cambiando algunos de los parámetros de configuración a conveniencia de los administradores y condicionado por los requisitos del sistema. Una meta común es hacer que este gestor corra más rápido en la plataforma en la que se encuentra instalado, ya sea afinando los parámetros de rendimiento o añadiendo más o mejor hardware. Entre los aspectos importantes para garantizar la optimización para una BD en *PostgreSQL*, está la buena configuración de archivos como *postgresql.conf*, el afinamiento de consultas, el ajuste del optimizador genético de consultas, los parámetros de *Autovacuum*, entre otros.

***Postgresql.conf*:** este es el archivo donde se guarda la configuración del *PostgreSQL* y se optimizan parámetros como la caché, la memoria compartida y la paginación. En *PostgreSQL* estos aspectos son vistos de la siguiente forma: *shared_buffers, cache_miss, wal_buffers*, *full_page_writes, effective_cache_size*.

Afinamiento de consultas: en este punto se optimizan las consultas SQL realizadas en la BD. Este optimizador debe garantizar un plan de ejecución que logre la efectividad de la consulta y en su comprobación toma un valor significativo las pruebas mediante el *EXPLAIN ANALYZE* para conocer el camino de la ejecución de las consultas, los tiempos asociados a la misma y mejorarlas en caso de que los resultados no respondan a los requisitos (González, 2011).

Optimizador Genético de Consulta: es un algoritmo genético que puede ser habilitado o deshabilitado, que fue incorporado dentro del analizador de consultas de *PostgreSQL*, para optimización de consultas que incorporan *JOIN*.

***Autovacuum*:** es un subproceso separado llamado demonio *autovacuum*. Es un proceso de limpieza que se encargada de revisar periódicamente la tablas con modificación considerables a sus tuplas, recuperando espacio de disco ocupado por filas modificadas o borradas. Actualiza las estadísticas usadas por el planificador/optimizador y evitar la pérdida de datos muy antiguos debido al rehúso del identificador de transacción (Greg Smith, 2012).

1.7 Herramientas de Modelado de Bases de datos

Luego de completar un modelo conceptual, se hace necesaria una herramienta para digitalizar el modelado del negocio, reflejando las entidades y relaciones desprendidas de la interpretación del mismo. Una herramienta que además, permita realizar la transformación de un modelo entidad-relación a un modelo relacional, así como, preste funcionalidades y

propiedades a los diseñadores de BD, en aras de establecer un buen diseño de BD. A continuación se realizó un estudio de 3 herramientas de modelado propuestas, con el objetivo de seleccionar la más idónea a las condiciones del diseño que se quiere establecer. Estas herramientas fueron: el *Embarcadero ER/Studio*, *Visual Paradigm* y el *CASE Studio.* Las cuales luego de ser caracterizadas, comparadas y analizadas con las características en las cuales se desarrolla el proyecto, arrojaron al *Visual Paradigm* como herramienta seleccionada para desarrollar el modelado la BD del SIB. Lo cual queda justificado por las siguientes razones:

- ✓ La universidad cuenta con la licencia de uso.
- ✓ Permite modelar en UML 2.0.
- ✓ Genera código SQL.
- ✓ Permite una buena integración con el gestor seleccionado y genera todos los artefactos necesarios para el trabajo con la BD.
- ✓ Permite el uso de control de versiones mediante la herramienta utilizada, Subversión.
- ✓ El dominio de los diseñadores sobre ella, permite disminuir la curva de aprendizaje y asimilación en el desarrollo del diseño.

1.8 Herramientas de Gestión de Bases de Datos

Las herramientas de gestión de BD son programas que poseen un conjunto de funcionalidades que posibilitan administrar, manejar las BD desde un gestor cualesquiera. Se propusieron las 2 herramientas de gestión más conocidas para realizar un análisis de sus características, en aras de determinar cuál puede ajustarse más al SGBD escogido y a las necesidades del proyecto. Estas fueron: *PgAdmin lll* y el *PhpPgAdmin*. Al analizar dichas herramientas de gestión, el conocimiento de los diseñadores en cada una de ellas y las especificaciones del proyecto, se aprobó la utilización como herramienta de administración de bases de datos, el *PgAdmin III*. Ya que la misma es la herramienta *Open Source* de administración por excelencia para bases de datos *PostgreSQL*, brinda soporte para todos los tipos de objetos del gestor en cuestión y se puede usar en el sistema operativo (SO) con el que se va a trabajar, Linux; aunque brinda además la posibilidad de migrar la BD a otro SO de existir la necesidad. Posibilita la asignación de roles, brinda buena seguridad de los datos, está diseñado para satisfacer cualquier demanda de los usuarios. Su sencilla interfaz gráfica facilita enormemente su administración, posee un editor de texto para sintaxis SQL con completamiento de código, sintaxis resaltada y depuración de errores y su documentación está disponible en más de una docena de idiomas, entre ellos el español.

1.9 Spring, Hibernate, Symfony y Doctrine

La posibilidad de que un programador no necesite plantearse la estructura global de una aplicación antes de hacerla, es una razón más que suficiente para justificar la utilización de un *framework*. Un *framework*, en términos generales, es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular. Es una estructura conceptual y tecnológica de soporte definido, mediante la cual otro proyecto de *software* puede ser organizado y desarrollado (Chávez, 2011). ***Spring*****,** ***Hibernate*****,** ***Symfony*** y ***Doctrine*** son ejemplos de *framework* y son los que por definición de arquitectura fueron escogidos para gestionar el sistema SIB para el SAIME.

Spring es un *framework* de código abierto muy popular y ampliamente desarrollado que ayuda a los desarrolladores a construir aplicaciones de alta calidad y con mayor rapidez. Ofrece una programación coherente, un modelo de configuración bien entendido, una gama de capacidades para la creación de proyectos de *Java*, ricos para web y aplicaciones empresariales de integración que pueden ser consumidas con un costo ligero (SpringSource, 2012). Está preparado para hacer frente a todos los niveles de arquitectura de una aplicación *J2EE* típico (*Java* 2 *Platform*, *Enterprise Edition*), es el único que ofrece una amplia gama de servicios, así como un contenedor ligero. A continuación se cita uno de sus módulos más importantes:

Abstracción de acceso a datos: *Spring* anima a un enfoque coherente de arquitectura para acceso a datos, y proporciona una abstracción única y poderosa para su ejecución. *Spring* proporciona una jerarquía de excepciones rica de acceso a datos, independiente de cualquier producto de la persistencia en particular. A diferencia de los *frameworks* de un solo nivel, tales como *Hibernate, Spring* tiene como objetivo ayudar a las aplicaciones en toda la estructura de una manera coherente y productiva, reuniendo más *frameworks* de un solo nivel para crear una arquitectura lógica. Un ejemplo de esta arquitectura se puede observar en la Figura 2 (Infociberland, 2012).

Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma *Java* que facilita el mapeo de atributos entre una BD relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (xml) o anotaciones en los *beans* (se usan para encapsular varios objetos en un único objeto) de las entidades que permiten establecer estas relaciones. No sólo se encarga de mapear las clases de *Java* a tablas de BD (y de tipos de datos de *Java* a tipos de datos SQL), sino que también provee facilidades de consulta y

recuperación de datos y puede reducir significativamente el tiempo de desarrollo que de otra forma se gasta en el manejo de los datos en SQL y JDBC (Figueroa, 2011).

En la Figura 3 se muestra una representación de una visión a alto nivel de la arquitectura de *Hibernate*. En este diagrama se muestra a esta herramienta usando la BD y los datos de configuración para proveer servicios de persistencia (y objetos persistentes) a la aplicación. Por lo que se puede abordar más diciendo que *Hibernate* constituye un *framework* de persistencia que entra en la categoría de *software* libre y que establece con Spring, una muy buena integración, haciendo a sus respectivas características óptimas potencialidades a la hora de gestionar una aplicación (Infociberland, 2012).

Por otra parte ***Symfony*** es un conjunto de relacionados, pero independientes *sub-frameworks*, que forman un completo *framework* MVC (Modelo, Vista, Controlador). Es un completo *framework* diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web.

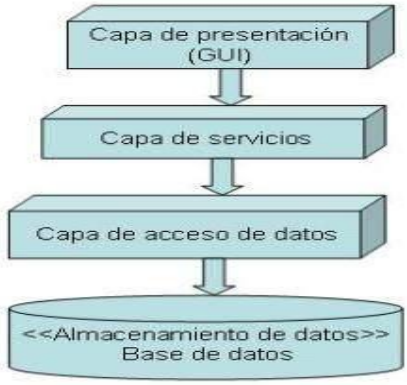

Figura 2. Arquitectura típica de una solicitud *Spring*[7].

Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web (Librosweb, 2009).

[7] Tomando del sitio web: http://infociberland.comxa.com/spring/config.php

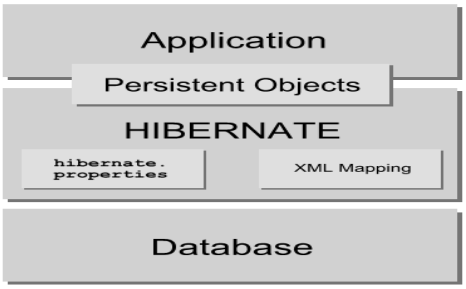

Figura 3. Arquitectura de *Hibernate*[8].

Doctrine es un mapeador de objetos-relacional (ORM) escrito en PHP y proporciona una capa de persistencia para objetos de este tipo de programación. Es una capa de abstracción que se sitúa justo encima de un SGBD. Es una librería para PHP que permite trabajar con un esquema de BD como si fuese un conjunto de objetos, y no de tablas y registros. En general *Doctrine* está inspirado en *Hibernate*, que es uno de los ORM más populares y grandes que existen y nos brinda una capa de abstracción de la BD muy completa. La característica más importante de este ORM es que da la posibilidad de escribir consultas de BD en un lenguaje propio llamado *Doctrine Query Language* (DQL) (Guardado, 2010). Utilizar un ORM como *Doctrine*, posibilita una serie de ventajas que facilita enormemente tareas comunes y de mantenimiento, como reutilización de métodos, encapsulación de la lógica de datos, portabilidad de sintaxis capaz de traducir a diferentes tipos de bases de datos, seguridad ante ataques como SQL *Injections*, mantenimiento del código, muy buen rendimiento en ejecución y una forma muy concisa al escribir consultas complejas (Mata, 2009).

La integración de este *framework* (*Symfony*) con este mapeador (*Doctrine*) posibilita como característica más importante contar con unas series de tareas programables como la que permite la creación de un esquema que representará la BD de determinado sistema, en el entorno de trabajo del mismo (IDE de programación), lo cual conformará la creación del modelo de la BD de *Symfony*.

Generalizando el tema, una de las tareas más comunes y desafiantes para cualquier aplicación consiste en la persistencia y lectura de información hacia y desde una BD como se muestra en el Anexo 2. Afortunadamente, *Symfony* viene integrado con *Doctrine*, una biblioteca, cuyo único objetivo es brindar poderosas herramientas para facilitar la tarea antes mencionada (Pacheco, 2012). Algo parecido a lo que se muestra en la Figura 4**.**

[8] Tomado de Documentación de Referencia de *Hibernate* versión 3.0.5

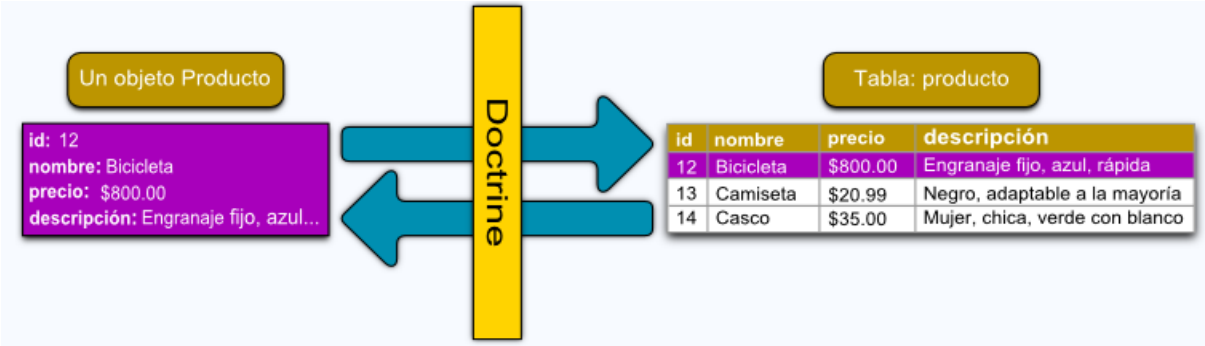

Figura 4. Ejemplo del trabajo de Doctrine[9].

1.10 Conclusiones parciales

Luego del estudio realizado se determinó como forma de garantizar la persistencia y la seguridad de los datos obtenidos en los procesos de pago que gestionará el SIB, el diseño de una BD relacional. Para esto se tuvieron en cuenta aspectos como los principios y fases del diseño, los que darán paso a un buen diseño de la BD y permitirá que la información pueda ser recuperada y almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder administrativo. Además la selección se justifica también, por las características antes expuestas que sitúan a dicho modelo de datos, como uno de los modelos más usados a nivel mundial. Se utilizará el modelo entidad relación para el diseño del modelo conceptual y el modelo relacional para el diseño lógico de la BD. La normalización adecuada de las entidades del modelo disminuirá la redundancia de información y por consiguiente evitará las anomalías en la actualización. En cuanto a las herramientas a utilizar en el diseño, se determinó utilizar como SGBD *PostgreSQL* 8.4 para el almacenamiento de los datos del SIB, por todas las ventajas que ofrece y en conveniente con las particulares del proyecto. Para la administración de la BD, *PgAdmin* III, el que brindará una buena interrelación por la sencillez de su interfaz gráfica, a los administradores de BD, así como aspectos de seguridad a la información, y para el modelado de las tablas de la BD, se escogió el *Visual Paradigm* en su versión 6.4, justificando la familiarización de los diseñadores con el mismo, así como su ayuda en la rápida construcción de aplicaciones de calidad y menor coste de recursos. Y en cuanto al acceso a datos se presentaron los *frameworks* involucrados y se explicó cómo es su interacción con las BD.

[9] Tomado de http://gitnacho.github.com/symfony-docs-es/book/doctrine.html

Capítulo II: Descripción y análisis de la solución propuesta

Introducción

Las características funcionales y técnicas de la BD del SIB a partir de los procesos referentes a la aplicación de este sistema, son la clave para la identificación de los elementos arquitectónicamente significativos, cada uno de ellos son puntos vitales a tener en cuenta para el diseño y para la decisión de elegir o no una herramienta que ayude a solventar situaciones que pudieran afectar y retardar la disponibilidad de la información de la BD en un momento determinado. En este apartado se abordan cuestiones sobre el negocio, se genera el modelo de datos del sistema luego de valorar los procesos que incluyen a la BD como un elemento fundamental. Se lleva a cabo el proceso de normalización con el objetivo de validar la estructura, y se tratan una serie de aspectos utilizados, como la nomenclatura, los patrones de diseños, la trazabilidad, la seguridad, entre otros.

2.1 Descripción de los procesos del sistema

Los procesos relacionados con la gestión de solicitudes, comunicación bancaria y pagos de servicios del SIB están encaminados a mejorar los mecanismos que actualmente utiliza el SAIME con los bancos recaudadores, para el cobro de los servicios que ofrece. Propiciando así, seguridad y eficiencia de la prestación de sus servicios, así como un mejor trabajo que elimine los problemas que por años ha sufrido esta entidad.

El proceso de gestión de solicitudes hace referencia a una aplicación web que está de cara a Internet, facilitándole al cliente del SAIME acceder con la finalidad de gestionar los servicios que sean de su interés. Además de esto también podrán: crear solicitud, administrar categorías de mensajes, ver sus preferencias, editar perfil, ver sus solicitudes activas, recuperar contraseña, solicitar devolución de pago, validar solicitud de devolución, aprobar devolución de pago. La cantidad de clientes que podrán interactuar con la aplicación es estimada por la cantidad de personas que accederán por internet a realizar solicitudes, más los que acceden para consultar los servicios u otras opciones, más los clientes que hacen su gestión por medio de las oficinas del SAIME en busca de beneficiarse de los servicios que este brinda. Por tanto se prevé la gestión de aproximadamente 7 mil solicitudes diarias. Esta cifra nos da la medida de que al interactuar el servidor de aplicaciones donde va a estar esta aplicación del SIB, con el servidor de BD a través de un protocolo TCP/IP, significaría más de 7 mil consultas solamente en lo que implica el entorno de este proceso, lo que exigiría una gran demanda de la BD en condiciones de tiempo real, tanto en términos de inserción como de muestreo de información.

La comunicación bancaria entre el sistema de los bancos y el SIB es sumamente importante porque es la que establece los canales de comunicación para que el proceso de pago sea posible. La gestión de comunicación bancaria se encarga de la comunicación con los bancos, configurar las direcciones de los servidores SFTP de los respectivos bancos, para generar y notificar los archivos de seguridad. En este proceso se crean una serie de archivos de seguridad que son almacenados en la BD, luego son enviados a cada banco correspondiente y a las entidades que lo secundan (Sucursales). Es necesario aclarar que a cada banco le corresponde un archivo de seguridad por el cual se van a regir las validaciones de sus transacciones. Por cada envío que se hace de estos códigos, se crea una notificación que cambia su estado en dependencia de que los bancos a los que se realizaron estos envíos notifiquen haber recibido los mismos.

El pago de servicios es el proceso más importante y complejo entre los procesos que maneja el sistema SIB, por tal razón es que requiere de un gran nivel de seguridad. Este ofrece un conjunto de servicios web que serán consumidos por la BD, para intercambiar información con el sistema de oficinas, referente a los pagos de los ciudadanos. El proceso consiste en que el cliente realiza un depósito bancario en una sucursal o un banco que atiende al SAIME, con intensiones de solicitar algún servicio del mencionado proveedor de servicios. Se genera un archivo *xml* acto seguido del pago efectuado, que lleva el código de la transacción realizada y el código de seguridad que responde al banco y sucursal donde se realizó esta (ráfaga bancaria). El banco le da un comprobante o *voucher*, para que vaya a las oficinas de este organismo a hacer la solicitud del servicio. Este archivo viaja desde los servidores de los bancos, a la BD del SIB donde es almacenado, para su posterior consulta. Cuando el cliente llega a las oficinas del SAIME el sistema de oficinas consulta la BD con ayuda de los servicios web y chequea que el código que trae el *voucher* que también registra el código del banco y de la sucursal donde se emitió, concuerda con el del archivo *xml* recibido. Aunque el archivo referido no haya llegado, se hace una búsqueda en la BD de los códigos de seguridad emitidos para cada banco y sucursal, para ver si coincide con el código de seguridad que trae el comprobante. Una vez comprobada su veracidad es validado como parcialmente correcto, y se le comunica al cliente que espere una notificación para poder comenzar los trámites por los cuales pagó o a solicitar el servicio por el cual pagó. Posterior a esto, del archivo *xml*, que lo que lleva en su contenido es una descripción de la transacción y una ráfaga (código completo que genera la transacción) de la misma, se extrae dicha ráfaga, la cual es sometida a un algoritmo aritmético, que genera un resultado. Esto mismo se hace en el sistema de los bancos

y luego se comparan los resultados para comprobar que la transacción ha sido totalmente correcta y relacionar de ser así, la solicitud de servicio realizada por el cliente y el pago efectuado por este. De no coincidir los dos resultados, se le comunica al usuario que no puede consumir el servicio o los servicios solicitados, y se avisa a las autoridades el intento de estafa del usuario. El mismo aviso ocurre cuando se está en presencia de una falsificación, al no poder validar el *voucher* con el chequeo de los códigos de seguridad. Lo anterior explicado se puede observar más abreviadamente en el Anexo 3, donde se muestra el diagrama de secuencia de dicho proceso.

En el transcurso de este proceso de pagos, son consumidos los siguientes servicios:

- ✓ Validación de ráfaga: valida una ráfaga que se envía desde las oficinas. Se utiliza para decirle al sistema SAIME si la ráfaga suministrada está correcta. De estar correcta, asocia el pago contenido en el *voucher* suministrado con la solicitud seleccionada y devuelve el identificador del pago al cual se le asoció la solicitud.
- ✓ Pagar solicitud: se utiliza para pagar una solicitud, el SAIME envía el identificador del pago del SIB que fue dado en la validación de la ráfaga, de donde el sistema responde si puede ser efectuado el pago en dependencia del costo del trámite y de ser positivo marca el servicio como pagado, de ser negativo envía una respuesta negativa y mensaje con la causa para que el oficinista esté informado, además envía el ID de la solicitud.
- ✓ Cambiar estado de solicitud: se encarga de cambiar el estado de las solicitudes enviadas según lo que necesita el SAIME, recibiendo de este el número de la solicitud y el estado al cual necesita cambiar. Es utilizado para liquidar las solicitudes, o modificarlas porque se haya cancelado mientras estaba pagado.
- ✓ Validar ráfagas *SemiOK*: (Consumido durante la noche), se encarga de decirle al SAIME cuáles de las ráfagas que en el momento de la validación no estaba la confirmación del banco está completamente bien.

Además de estos procesos que se van a implementar en el diseño de la BD, es fundamental mencionar otros dos servicios web que mantienen una interacción activa con la BD del SIB, ellos son: el servicio de actualización de pagos y el monitor web. Estos permiten tener un control notable a tiempo real de la información que se almacena en la BD. Tanto es así que el servicio de actualización de pagos está destinado para gestionar todas las lecturas que se efectúan en los servidores SFTP de los bancos, y esto conlleva a que vaya registrando toda

esta información como lecturas en la BD. Por otra parte el monitor web, es un servicio web ejecutado sobre un contenedor de servicios como lo es Axis2, que permite la ejecución de tareas programables que provocan búsquedas diarias en la BD de correos o mensajes listos para ser enviados, los cuales si existen los carga para que puedan ser enviados por el servidor de correos del SAIME. También ayuda al servicio anterior a monitorear las ejecuciones de las lecturas de forma automática, así como evaluar los servicios no consumidos y determinar el pago de estos como excedidos de tiempo de consumo.

2.2 Diagrama del sistema

La BD del SIB estará ubicada en un servidor de *Postgres SQL*, en el Centro de Datos del SAIME en el cual se encontrará la solución, se controla el acceso y se gestiona la información. En este lugar la BD interactuará mediante el protocolo TCP/IP con los servidores de aplicaciones y a través del consumo de servicios, con el servidor de correo y servidor SFTP de los bancos. Toda la comunicación con el Centro de Datos estará protegida por una red privada que brindara acceso a las oficinas del SAIME, a la Sede Central donde están los usuarios administrativos y los servidores ubicados en los bancos. La BD del SIB solo podrá ser accedida con permisos administrativos, a través de la aplicación por funcionarios con dichos permisos o directamente a través de la interfaz gráfica del gestor, por un administrador de BD. En conjunto con la BD del Sistema de los Bancos (SeB), esta gestionará archivos *xml* que se conforman de las notificaciones de los pagos y de los resúmenes de pagos diarios. Estos *xml* viajarán indistintamente a través de servidores SFTP y serán gestionados en la BD a través de servicios brindados por los contenedores de servicios del SAIME. Ejemplo de la explicación antes expuesta, se presenta en el Anexo 4.

2.3 Necesidades técnicas del Sistema

Requisitos no funcionales

- ✓ *Disponibilidad*: la BD del SIB debe estar disponible las veinticuatro horas del día, siempre que sea un día laboral, a no ser que desee realizar mantenimiento fuera de este período. Esto se debe a que las validaciones de las ráfagas se realizarán en el horario donde la red no esté tan afectada por el gran flujo de información que provocan estos procesos en ejecución. Este horario deberá ser en la noche luego del cierre de todos los bancos que son atendidos por el SAIME.

- ✓ *Desempeño*: en este sentido, la información almacenada de la BD del SIB, podrá ser consultada y actualizada de forma permanente y simultánea, sin que se afecte el tiempo de respuesta.
- ✓ *Escalabilidad*: la BD debe ser construida sobre la base de un desarrollo evolutivo e incremental, de manera tal que nuevas funcionalidades y requisitos relacionados puedan ser incorporados afectando el diseño y código existente de la menor manera posible; para ello deben incorporarse aspectos de reutilización de componentes.
- ✓ *Seguridad*: el acceso a la BD debe estar restringido por el uso de claves asignadas a cada uno de los usuarios. Sólo podrán ingresar al sistema las personas que estén registradas, estos usuarios serán clasificados en varios tipos de usuarios (o roles) con acceso a las opciones de trabajo y a la información definida para cada rol, posibilitando garantizar la confidencialidad e integridad de la misma. Además el acceso a los datos debe realizarse a través de comunicaciones seguras utilizando el protocolo *Secure Socket Layer* (SSL), el cual es un protocolo que establece un canal de comunicación seguro, cifrando el intercambio de datos entre dos equipos.
- ✓ *Validación de Información*: la BD debe validar automáticamente mediante su gestor la información contenida en los formularios de ingreso. En el proceso de validación de la información, se deben tener en cuenta aspectos tales como campos obligatorios, longitud de caracteres permitida por campo, manejo de tipos de datos, entre otros.
- ✓ *Software*: los programas utilizados para la confección del diseño de la BD serán preferentemente programas gratis, libres del pago de licencias, para poder ser utilizados.
- ✓ *Hardware*: la BD deberá estar instalada en un servidor destinado solamente para ella, ubicado en un lugar especifico, y deberá contar con una capacidad de almacenamiento de 1 *Terabyte*.

El caso de la disponibilidad de la información, es un aspecto que se debe analizar dentro de la arquitectura de BD, debido a la estimación de la gestión de solicitudes más 3 mil pagos por bancos que se esperan a diario. Estos dos procesos son capaces de generar la misma cantidad más uno, de los ficheros *xml* que se gestionan en la BD, ya que se elabora un fichero resumen al concluir la jornada. Además de esto se debe contar las consultas y entradas de los usuarios con permisos administrativos del SAIME que interactúan directamente con la BD en sus labores de control. Por tal razón se hace necesario proponer, la implantación de otro

servidor de BD que actué conjuntamente con el otro, es decir, dos servidores trabajarán como uno solo, a esto se le denomina: clúster, con el objetivo de garantizar la estabilidad de la BD, la disponibilidad y respaldo de la información, así como el aumento del rendimiento del sistema.

2.4 Réplica de datos y herramienta de replicación

Actualmente en el mundo la responsabilidad y la exigencia, sobre las soluciones informáticas que tengan una mayor rapidez de respuesta sobre solicitudes que se le hacen a los datos en tiempo real, recaen sobre los servidores de BD, los cuales pueden ser congestionados de tantos pedidos y acciones de respuesta que se ejecutan al mismo tiempo. La replicación de la información entre servidores constituye actualmente, una solución muy de moda para resolver dicha problemática.

La replicación juega un papel fundamental en los sistemas de BD que se utilizan en aplicaciones de tiempo real. Permite obtener mejoras en cuanto a disponibilidad de los datos, ya que la información tendrá al menos una copia como mínimo, que pueda ser usada para la recuperación del sistema ante un fallo ocurrido (Hoyo, 2011).

Los servidores sobre los cuales se aplica la técnica de replicación, son ordenadores que se integran en conjunto con la idea de que operen como uno solo, en aras de garantizar disponibilidad, realizar la réplica de datos, balancear las cargas entre los servidores y por consiguiente, elevar el rendimiento del sistema. Pero los mismos pueden ser ayudados o no, en este proceso de replicación por herramientas destinadas a este fin, todo está en dependencia de que si el sistema gestor que tienen instalados pueden o no realizar esta técnica automáticamente. En el caso de *PostgresSQL*, gestor seleccionado para trabajar en esta investigación, hasta la versión 8.4 se necesita una herramienta de replicación que responda a las expectativas que se buscan.

Por tal motivo se realiza una investigación que analiza varias herramientas de replicación, desprendiendo una comparación entre ellas y la selección final de la herramienta, como se puede observar en la Tabla 2. Dicho estudio permite a la arquitectura de la BD del SIB, la posibilidad de contar con una buena propuesta de herramienta, bien documentada, aprobada y apoyada por otros proyectos y su comunidad. La selección da nombre al Pgpool-II, el cual es un *software* intermediario que opera entre las aplicaciones clientes (*frontends*) y los procesos de *PostgreSQL* (*backends*) manejando las comunicaciones entre ellos y posibilitando además de la réplica, el balanceo de cargas y la ejecución concurrente de consultas.

Tabla 2: Comparación de herramientas[10].

Programas	Licencia	Madurez	Método de Replicación	Sincronización	Agrupación de conexiones	Balanceo de cargas	Particionamiento de consultas
PgCluster	BSB	estancada	Maestro-Maestro	sincrónico	No	Si	No
pgpool-I	BSB	estancada	Basado en Middleware	sincrónico	Si	Si	No
Pgpool-II	BSB	Reciente liberación	Basado en Middleware	sincrónico	Si	Si	Si
slony	BSB	estancada	Maestro-Esclavo	asincrónico	No	No	No
Bucardo	BSB	estancada	Maestro-Maestro, Maestro-Esclavo	asincrónico	No	No	No
Londiste	BSB	estancada	Maestro-Esclavo	asincrónico	No	No	No
Mammoth	BSB	estancada	Maestro-Esclavo	asincrónico	No	No	No
rubyrep	MIT	Reciente liberación	Maestro-Maestro, Maestro-Esclavo	asincrónico	No	No	No

2.5 Diseño de la BD del SIB

El diseño de la BD inicia a partir de la identificación de los conceptos de negocio más importantes en el dominio del problema, determinándose así las entidades, atributos y restricciones que son necesarios para el modelado de la misma. Posteriormente se normaliza el modelo llevándolo a una de las formas normales, según las necesidades del sistema y finalmente se realiza el diseño físico de la BD a partir del modelo construido. Para el inicio de este proceso se realizó un análisis de los requisitos más significativos extraídos del documento de levantamiento de requisitos del proyecto Integración Bancos con el SAIME.

Requisito funcional (RF) 1. Crear solicitud de pago de servicios

La función del requisito es la misma que se anuncia en él, y para esto es necesario que exista al menos un proveedor con una serie de servicios que deben ser mostrados al cliente para que este pueda solicitarlos y pagar por ellos, para su posterior cobro. En esta solicitud se pueden comprar varias unidades de un mismo servicio, y todo esto influye en el monto final que es calculado y guardado luego junto con la solicitud completada. En esta funcionalidad se hace referencia a las tablas: Dcliente, Dproveedor, Dservicio, Dunidadservicio y dSolicitud.

RF 6. Administrar tipos de pago

Aquí pasa lo mismo que en la pasada funcionalidad descrita, pero ahora con los tipos de pagos. Por lo cual se propone una tabla Dtipopago con sus atributos ya definidos en esta funcionalidad, con el fin de lograr lo que pide el requisito.

[10] Sacado de la wiki de *PostgreSQL.*
http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling

RF 8. Registrar pago ordinario de solicitud activa

Este requisito hace énfasis en el registro de un pago ordinario de una solicitud activa que realiza un operador del sistema cuando el cliente paga por los servicios plasmados en una solicitud. El operador antes mencionado resulta ser un trabajador que se encuentra registrando el depósito en una sucursal perteneciente a un banco, por lo cual es indispensable tener una tabla bancos y otra de sucursales relacionadas entre ellas y cada una con un código que la identifique. Por tal razón se propone la tabla DPago, que representa el pago ordinario y las tablas Dbancorecaudador, Dsucursal y Dcodigosucursal por ser un código con cierta complejidad que no se puede registrar de forma simple en la tabla de sucursales. Además de lo planteado, en el presente análisis se observa que hasta el momento cada vez que se habla de una notificación, no se habla de una sola, por lo cual se propone a TipoNotificacion como nuevo nomenclador, para albergar a todas las notificaciones posibles.

RF 17. Administrar cuentas recaudadoras en bancos

Para la administración de las cuentas recaudadoras en los bancos se requiere de la existencia de una tabla de la que se pueda gestionar todo lo referente a estas cuentas, por tal razón se crea la tabla Dcuentarecaudadora. Dicha tabla tiene entre sus atributos el nomenclador denominado: EstadoCuenta.

RF 18. Gestionar preferencias de los clientes

Esta funcionalidad está relacionada como ya lo dice, con las preferencias que tiene el cliente, pero solo a la hora de recibir los avisos o las notificaciones de cualquier eventualidad. Para esto se propone la tabla Dpreferenciacliente, donde se registrará todo lo referente a las notificaciones que el cliente desea recibir y por cual medio prefiere hacerlo. Esto se trabajará a nivel de usuario, por lo cual se tendrá que guardar el usuario de cada cliente.

RF 22. Generar reporte de pagos diarios por oficina para un proveedor de servicio

En el presente requisito se crea la tabla Doficina, como representación de los lugares donde el proveedor ofrece sus servicios y desde donde se generan los reportes de pagos. Cada oficina tiene un código complejo con una serie de datos que la identifican, por tal razón se crea otra tabla llamada dCodigoOficina.

RF 24. Configurar dirección sftp

El presente requisito fomenta la creación de una tabla que responden a la configuración sftp o del servidor correspondiente a un banco determinado. La tabla a crear es Dservidor.

RF 30. Administrar lecturas de depósito de pagos

Para llevar a cabo la gestión y administración de las lecturas, se define la tabla con el nombre de Dlectura y con los atributos que se interpretan del negocio. Debido a que las lecturas las realiza el proveedor sobre el servidor de cada banco, se establece una relación entre servidor y lectura, así como proveedor y lectura.

En el Anexo 5 se muestra una representación de las primeras entidades o clases, así como algunas de sus relaciones que se desprenden de los previos análisis del negocio. La misma hace una representación conceptual del negocio, a través de los conceptos desprendidos de los requisitos más relevantes y las relaciones entre ellos.

La relación entre Dcliente y Dsolicitud, significa que un cliente puede crear una solicitud de un servicio o de varias unidades de estos, por la cual va a generarse un pago, el cual será en dependencia de los tipos de pagos de los servicios solicitados. Estos servicios serán ofertados por un proveedor que tendrá además de cuentas recaudadoras en bancos que responderán ante él como un cliente más, un conjunto de oficinas que se encargarán luego de que las transacciones bancarias referentes a los pagos de servicios sean efectuadas, darle al cliente la posibilidad de disfrutar de sus peticiones. Cada banco contará con un conjunto de sucursales desplegadas por todo el país, para que los clientes que no puedan acceder por INTERNET en busca de solicitar dichos servicios, lo hagan en dichos locales. Contarán además con servidores que gestionarán todas las transacciones, los cuales podrán ser controlados por el proveedor de servicios.

Además del análisis realizado de los requisitos más involucrados con el negocio, existen otros que van a infringir con el crecimiento de la BD. Tal es el caso del requisito número cuatro del documento de requisitos del proyecto en cuestión, el cual sugiere que para la comunicación entre el proveedor y el cliente, debe existir un contenedor de información donde se registren listas de usuarios que en un momento determinado deben ser avisados ante cualquier eventualidad por sus proveedores. Cada aviso tendrá un mensaje diferente que hará referencia a un servicio, una notificación o felicitación y podrá ser almacenado como un correo una vez enviado o también, aun cuando no se tenga intenciones en ese momento de ser enviado, se guardará y será enviado posteriormente mediante una planificación automática. Por tal motivo se añadieron 5 nuevas clases más al diseño, las cuales se presentan en el Anexo 6 del documento.

Otro caso que requiere de atención es el tema de que los clientes pueden realizar pagos por servicios que solicitan y luego consumirlos posteriormente a la validación de los pagos. Pero

estos servicios pagados tendrán un tiempo para consumirlos, que al ser cumplido el cliente no podrá reclamar por el servicio ni por el dinero. También existe un tiempo de reclamación, el cual al igual que el de expiración, es definido por el cliente. Es por eso que un tiempo antes de que estos pagos pasen a ser tomados como inválidos, son calificados como excedentes o habilitados para reclamación, y relacionados en la entidad DExcedentePago, la cual registrará tanto a estos pagos no consumidos, como a los consumidos y que hayan generado un sobrante de dinero que el cliente tiene el derecho de reclamar. Una vez que pase este tiempo de reclamación, el dinero no podrá ser reclamado y se le destina un empleo, el cual generará un registro en la BD que albergará el destino de estos pagos. El contenedor de datos que permitirá el control de esta información será la entidad: DdestinoExcedente. Las entidades mencionadas son mostradas en el Anexo 7 donde se hace referencia a este proceso.

Las lecturas realizadas por el proveedor sobre los servidores de los bancos responden al proceso de actualización de pagos y son de vital importancia para el desarrollo de un proceso monitorizado y seguro. Es por tal razón que se hace indispensable crear un representativo de un contenedor de información, que registre las transacciones que se hacen a diario en estos bancos. Haciendo posible la posterior consulta de los proveedores de servicios sobre estas, con el fin de mantener el control de sus intereses. El contenedor antes referenciado lo constituye la entidad Dlectura, la cual dará almacenamiento a dichos registros. Pero las lecturas pueden terminar, en un momento determinado siendo no atendidas por problemas con los servidores de los bancos o de cualquier otra índole, por lo que se hace necesario tener un registro de cuáles son, para en otro momento ser ejecutadas. Por tal razón se crea la entidad Dpendiente, con la posibilidad de almacenar datos de las mismas, los cuales serán consultados a la hora de ejecutar la lectura nuevamente y eliminarlos de esta entidad una vez que esta se ejecute. La ejecución de estas será una vez iniciado el sistema o para cuando se hayan configurado. Las entidades mencionadas son mostradas en el Anexo 8 que hace referencia a este proceso.

Luego de estas primeras representaciones se agregaron dos nuevas clases mostradas en el Anexo 9, que culminarían con el diseño de clases persistentes de la BD del sistema SIB. La primera de ellas hace referencia a los pagos realizados en puntos de ventas atendidos por el SAIME, los cuales, aunque son pagos de servicios normales como cualquier otro, son registrados de forma diferenciada para establecer más control sobre estos. Y la segunda es una representación de las notificaciones que serían emitidas por los bancos en el proceso del negocio descrito como: comunicación bancaria.

Posterior al análisis de las clases persistentes se dio paso a la materialización digital de la BD del SIB, con la creación de las tablas que la conforman. Algunas de las clases que se definieron en un principio fueron eliminadas junto con sus relaciones, como es el caso de: Dnotificacion, así como otras que no eran necesarias. Todo esto se hizo con el objetivo de congeniar, un modelo de tablas que se ajustara a las exigencias de la BD. Luego de varias modelaciones y ajustes de las tablas involucradas en la BD del SIB, se crea un modelo físico previo y posteriormente uno final. En el cual, debido a la gran cantidad de atributos contenidos en las representaciones gráficas de cada tabla, se muestran una representación de estas con algunos de sus atributos y sus respectivas relaciones, en el Anexo 10. La descripción de los atributos que cada una de las tablas, aparece relacionada más adelante en el Anexo 11.

2.6 Nomenclatura

En el diseño de la BD del SIB se empleó una nomenclatura sencilla, que fue mejorada luego del proceso de resaltar las clases persistentes. De esta se esperaba que ofreciera una buena compresión y organización de las tablas creadas, así como de sus respectivos atributos y relaciones. Los nombres utilizados para identificar las tablas de la BD están formados por un prefijo seguido de un criterio de agrupación. Las palabras que las componen en un primer momento estaban compuestas por el prefijo “D”, pero luego se asumió que siempre comenzarán con una “d” minúscula y seguido la palabra que identifica la tabla, la que siempre comienza con mayúscula. Ejemplo: dPendiente, en caso de que el nombre de la tabla sea compuesto, se establece la misma regla, ejemplo: dPagoPuntoVenta. También se puede encontrar la utilización del guión bajo “_”en tablas que responden a relaciones de muchos a muchos, como en atributos, ejemplo: dProveedor_dServidor y user_id. En el caso de las tablas del plugin empleado (vea en epígrafe 2.12 Seguridad en el diseño) el prefijo utilizado es “sf_guard” que acompaña al nombre de la tabla en minúscula, ejemplo: sf_guard_user y sf_guard_user_permission en caso de que el nombre de la tabla sea compuesto.

En la nomenclatura de los atributos identificadores la mayoría de las veces se establece el profijo “id” solamente o este, acompañado de la descripción de la tabla como es el caso de idLectura. Para los atributos que constituyen llaves foráneas se establece que su nombre estará compuesto por el nombre de la tabla que hace referencia a dicho atributo más el nombre del atributo en dicha tabla (salvo excepciones), un ejemplo de esto es: dSucursalNoSucursal en la tabla dCodigoSucursal, pero cuando es el caso de que un atributo que es referenciado en varias entidades consecutivamente se pone el nombre del atributo primero y luego el nombre

de la entidad que lo contiene (salvo excepciones), por ejemplo: idBancoRecaudador que es referenciado en dSucursal y en dCodigoSucursal. Fuera de toda regla y exceptuando un predominio de las inicializaciones con mayúsculas, los atributos que en su mayoría no son ni llaves primarias ni foráneas, se describen normalmente con su nombre, ejemplo: IP, Puerto, tipoMensaje, NombreServicio, NumeroCuenta, Descripcion, Monto, fecha, username, destino, entre otros.

Para la nomenclatura de los dominios se definió como prefijo a (dom_) y posterior el tipo de datos o el nombre del atributo como por ejemplo: dom_username, y en caso de que se refiera a cadenas de caracteres tanto numéricas como alfanuméricas, se procede a anteponer (dom-_cadena) y posterior la dimensión que tendrán. En el caso de que sea una cadena que solo permite letras se pone (dom_cadena).

La nomenclatura de los índices estará conformada por el nombre de la tabla a la cual responden, luego un guión bajo, el nombre del atributo, otro guión bajo y terminará con “pkey”, siempre que se haga referencia a una llave primaria o a un atributo único (*UNIQUE*). Existirá otro caso que representará a los índices creados por el diseñador de la BD, los cuales tendrán la misma nomenclatura al principio, lo que luego del guión bajo, terminara con “idx” o el nombre del atributo.

La creación de los disparadores (*Triggers*), estará normada para todos los casos, por la nomenclatura que comienza con el prefijo: “t”, luego un guión bajo y el nombre del *triggers*. Este nombre responderá a una función que hace vigente el objetivo del disparador y cuyo prefijo será “ft” y luego la misma nomenclatura descrita anteriormente.

Es importante aclarar en este documento, que en las descripciones que hacen referencias a figuras o anexos, donde se habla de las entidades o tablas de la BD, se procede a describir a estas, por el nombre en que aparecen en la interfaz gráficas del gestor de base de datos. El cual traduce siempre sus nombres en letras minúsculas.

2.7 Normalización de la BD

El proceso de normalización de las BD es de importancia primordial a la hora de realizar su diseño, el mismo evita la redundancia de información o por lo menos permite la existencia de esta en menor grado, protege la integridad de los datos ante cualquier operación realizada en las tablas, ya sea de inserción, eliminación o actualización de los datos, y así permite optimizar los procesos de gestión de información almacenada.

Al aplicar la normalización a la BD del SIB, se fue tratando de que cada entidad cumpliera con las reglas y restricciones establecidas para cada una de las formas normales, lográndose en todas ellas. Un ejemplo de esto son las entidades involucradas en el proceso de envió de notificaciones del proveedor por medio de correos o SMS a los clientes, mostradas en el Anexo 12. Estas entidades cumplen con las restricciones establecidas en la 1FN, pues poseen una llave primaria y sus dominios no tienen elementos que, a su vez, sean conjuntos. Cumplen además con las restricciones propuestas por la 2FN, pues toda relación que se encuentre en 1FN y que además su clave primaria no sea compuesta, se encuentra en 2FN y también se encuentran en 3FN pues no existen dependencias transitivas entre atributos ni llaves.

En el anexo antes mencionado se puede observar que existen dos tablas que poseen dos llaves primarias y foráneas cada una. Esto ocurre como resultado a la normalización automática que realiza el modelador, referente a la ocurrencia de relaciones de mucho a mucho entre entidades, siguiendo reglas de diseño que contrarrestan los modelos ineficientes y mal estructurados. Estas reglas de diseño aplicadas en este caso, son evidenciadas en el patrón de asociación antes expuesto, y en el diseño de la BD del SIB este fenómeno ocurre cuatro veces más en distintas relaciones.

Sin embargo luego de la normalización de la BD hasta la 3FN se analizó si existía la posibilidad de mejorar el rendimiento da la misma y se decidió por este mismo motivo, desnormalizar algunas entidades hasta la 2FN. Un ejemplo de esto es la entidad dBancoRecaudador, representado en el Anexo 13, la cual no cumple con las restricciones propuestas por la 3FN, que plantea que la relación debe estar en 2FN y además se deben eliminar las dependencias transitivas entre los atributos respecto a la llave primaria. La figura muestra la dependencia transitiva que existe entre los atributos id y NombreContacto y los demás que están señalados, debido a que el contacto de un banco, que es el encargado de recibir las notificaciones de cualquier índole enviada por un proveedor determinado, está identificado con ese banco, es decir este atributo depende del identificador de ese banco. Pero a su vez, de NombreContacto dependen tres atributos más que son los que están señalados en la figura con la flecha en amarillo. Esta inclusión realizada en la tabla dBancoRecaudador, afectó la normalización de la misma en aras de mantener la tercera forma normal, pero fue un ajuste necesario para con el cual, ganar en rendimiento. Esto evitaría no tener que recurrir a la tabla de los contactos para obtener información de estos, lo cual significaría realizar otra búsqueda sobre otra tabla del modelo.

De esta manera queda evidenciado como la BD del SIB queda normalizada de manera general hasta la tercera forma normal. Y estructurada correctamente, según lo más conveniente para el sistema ante el cual tendrá que responder, permitiendo un mínimo de redundancia y una buena optimización de las consultas, según el diseño práctico realizado.

2.8 Restricciones del modelo de datos

Este modelo cumple con un grupo de restricciones mostradas en el Anexo 14 del documento, con el objetivo de apoyar la integridad de la BD:

- ✓ Todos los atributos se encuentran sujetos a un dominio con el objetivo de garantizar la consistencia y validez de los datos, exceptuando aquellos que son de tipo TIMESTAMP y BIGSERIAL.
    - ✓ En dicho anexo aparecen los dominios definidos para la BD.
- ✓ La integridad de la entidad se garantiza para todas aquellas entidades que poseen más de un atributo como llave primaria, pues ninguno de ellos puede tomar valores nulos.
- ✓ No existe más de una tupla perteneciente a una relación con la misma clave.
- ✓ Todas las relaciones entre entidades se encuentran sujetas a referencias entre entidades del modelo. Con ello se garantiza que una llave foránea apunte a una entidad con una tupla, fila o registro existente dentro de la BD, lo que permite minimizar la redundancia de los datos y con ello aumentar la integridad de los mismos.

2.9 Indexación

Un índice es una estructura auxiliar que permite acelerar el rendimiento de las búsquedas siempre que la consulta tenga condiciones asociadas a ellos y pueden ser creados utilizando una o más columnas de una tabla. Una incorrecta estrategia de indexado puede repercutir negativamente en el rendimiento, pues para aquellas tablas que posean un número pequeño de tuplas, resulta más costoso realizar una búsqueda indexada que secuencial, además las operaciones de inserción y actualización se tornan más lentas, ya que requieren de una modificación del índice.

PostgreSQL ofrece cuatro tipos de índices: B-Tree, Hash, GIST y GIN. Cada uno de estos tipos utiliza un algoritmo diferente para determinados tipos de consulta. Si no se especifica el tipo de índice a utilizar, *PostgreSQL* crea por defecto los índices de tipo B-Tree, debido a que es el más utilizado (Hoyo, 2011).

En el modelo de datos del SIB, el tipo de índice utilizado fue el B-tree pues además de ser muy fácil su uso y poseer probados beneficios en sus métodos de inserción y eliminación al

mantener balanceado el árbol, todos los creados están acordes con los requisitos funcionales del sistema. Fueron indexadas primeramente aquellas columnas que en el momento de su creación se clasificaron como atributos únicos (*UNIQUE*), además de la indexación que realiza *PostgreSQL* automáticamente cuando se define una llave primaria. Luego para evitar la creación de índices innecesarios que pudieran influir negativamente en el rendimiento de la BD, se realiza un análisis y se decide dejar los índices básicos creados por el gestor y colocar algunos en las tablas de mayores registros. Ejemplo de esto se tienen a la tabla dPago, que además de su identificador se indexó los atributos: numerovoucher y fechapago. Esta sencilla indexación, permitirá que la BD del SIB esté preparada para que en eventos de mucha gestión de información, esta pueda proporcionar una mayor maniobrabilidad y disponibilidad de la información.

2.10 Patrones de diseño utilizados

En los proyectos, tesis y trabajos donde el diseño y modelación de una BD es el tema a resolver, se pueden observar similitudes en sus modelos, factores y representaciones que se repiten y que si son correctamente identificados llegan a convertirse en patrones de diseño. Un patrón de diseño es una solución a un problema determinado que surge durante el proceso de desarrollo de *software* que está probado y documentado; está definido como una: “solución común a un problema común de un determinado contexto” (Jacobson, 2000).

Durante el diseño de la BD para el SIB se utilizaron diferentes patrones de diseño que conllevaron a la creación de nuevas tablas. Los patrones fueron el patrón de asociación, patrón de llaves subrogadas y el patrón RBAC, los cuales ayudaron a ajustar el diseño de la BD a buenos niveles de aceptación por sus diseñadores.

El patrón de asociación representa las relaciones entre entidades del modelo, este fue utilizado para representar asociaciones de tipo muchos a muchos. Las bases de datos relacionales no soportan relaciones directas de este tipo. Esta situación puede ser resuelta creando una tabla en el modelo de la BD que contenga las claves primarias de dos tablas involucradas en este tipo de relación. Mostrando así su asociación, sin afectar la funcionalidad de lo que se quiere representar. La tabla resultante como se puede observar en el Anexo 15, estará compuesta por dos llaves primarias y foráneas al mismo tiempo, las cuales serán las llaves primarias de las entidades a relacionar. También en este caso se contará con dos atributos que determinarán la accesibilidad de una de ellas sobre la otra entidad que se está relacionando.

El patrón de llaves subrogadas fue empleado en la BD, para generar llaves primarias únicas con el objetivo de conducir el diseño a implementaciones más eficientes. Garantizando una absoluta certeza de que su utilización, no será susceptible de cambiar con el tiempo, facilitando así el manejo de las relaciones en lugar de dificultarlo. Normalmente en los modelos de datos se utilizan números enteros en columnas auto-incrementales o identificadoras del tipo UUID. Este patrón es muy utilizado por las entidades que integran el modelo del SIB, ya que presentan en su mayoría identificadores de tipo int8, los cuales son generados a través de secuencias auto-incrementales. Un ejemplo de este patrón se puede observar en la Figura 5, la que representa la tabla dServicio.

El modelo RBAC (Role Based Access Control o control de acceso basado en roles) es una tecnología creada para satisfacer las principales necesidades del control de acceso, en el que las tareas no están ligadas directamente a los usuarios. El concepto de rol forma una capa intermedia entre los usuarios y las tareas. A través de este modelo patrón, los permisos para ejecutar estas tareas son asignados a roles específicos, y estos roles son asignados a los usuarios del sistema. El manejo de los permisos de cada usuario está determinado por la asignación del rol apropiado, ya que a los usuarios no se les asignan permisos directamente, sino que los adquiere a través de los roles que le son asignados. Este patrón se pone de manifiesto en las tablas que conforman el *plugin sfGuard* (Anexo 16), donde se hace uso del modelo de usuarios, grupos y permisos para hacer posible que un usuario en dependencia de la asignación de roles que se le haga, pertenezca a un grupo que determina un rol específico y disfrute de los permisos que obtiene como miembros de dicho grupo.

idservicio	idtipot	idproveedor	estadoservicio	tipopago	nombreservicio
1	104	1	3	1	Original Cle
2	106	1	3	1	Renovacion Actualizacion Cle
3	105	1	3	1	Renovacion Cle
4	34	1	3	1	OriginalSC
5	6	1	3	1	Original
6	7	1	3	1	Duplicado
7	8	1	3	1	Renovacion
8	9	1	3	1	Renovacion Onidex
9	26	1	3	1	Actualizacion
10	63	1	3	1	Entrada al pais

Figura 5. Ejemplo del patrón de llaves subrogadas.

Michael Blaha, en su libro “Patterns Of Data Modeling”, describe a los patrones de BD como plantillas que ya han sido evaluadas como la responsable de resolver un problema, son la guía para apoyarse en realizar el trabajo (Ecured, 2010). En ocasiones se utilizan soluciones que no están evaluadas o estandarizadas, para resolver problemas que posibilitan un buen diseño.

El uso de una representación arbolada en una de las tablas de la BD del SIB, posibilitó almacenar los nomencladores que fueron apareciendo en el análisis del negocio y la modelación. Estos se convirtieron en atributos de la tabla dNomenclador, como por ejemplo: estadosolicitud, estadoservicio, tiposervicio, entre tantos otros que presentan una serie de valores multievaluados que ante una normalización, harían que la BD del SIB, creciera considerablemente el número de tablas solamente para consultar.

Esta estructura mostrada en la Figura 6, permitió encapsular de manera organizada a todos los nomencladores y sus variables en una sola tabla. Facilitando además el uso de los mismos por parte del gestor y acto seguido, del administrador de BD. A favor de ella se puede decir que para proyectos muy grandes con gran cantidad de nomencladores se hace muy ineficiente debido a que la búsqueda en la tabla se haría muy lenta, pero para el caso del sistema SIB se adapta perfectamente a las necesidades de abstracción y encapsulamiento que se requieren.

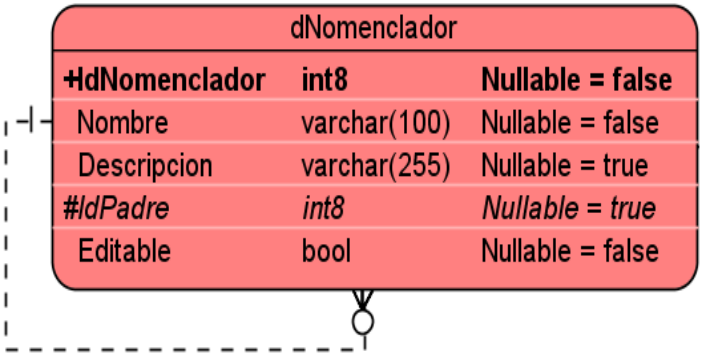

Figura 6. Tabla nomenclador.

Como se puede observar en la Figura 7, los atributos nomencladores (hijos) circulados son los atributos multievaluados que el idpadre es 0 (número que hace referencia a la raíz del árbol), y los valores en los cuales se desglosan, heredan de ellos mismos (es decir su idpadre es igual al idnomenclador del nomenclador como tal).

	idnomenclador [PK] bigserial	nombre character varying(100	descripcion character v	idpadre bigint	editable boolean
1	0	Nomenclador	Padre de todc		FALSE
2	1	estadoservicio	Diferentes es	0	FALSE
3	2	Habilitado	Trámites que	1	FALSE
4	3	Deshabilitado	Trámites que	1	FALSE
5	4	estadocuenta	Estados o tipc	0	FALSE
6	5	Corriente	Cuentas de a	4	FALSE
7	6	Ahorro	Cuentas de a	4	FALSE

Figura 7. Registros de la tabla de nomencladores[11].

2.11 Trazabilidad

La trazabilidad es la capacidad que tiene una organización o sistema para rastrear, detectar, reconstruir o restablecer relaciones entre objetos monitoreados para analizar y especificar situaciones específicas o generales en los mismos (Cano, 2005). La trazabilidad de la BD del SIB será manejada a través de acciones registradas en la BD durante la jornada, a las que tienen acceso el administrador de BD y los funcionarios con permisos administrativos. Es decir que cada vez que se realice una acción, se generará una consulta que almacenará en la tabla dTraza representada en la Figura 8**,** el identificador de la traza, el tipo de acción que se ejecuta, el evento, el usuario de la persona que la ejecuta, una descripción, así como la hora en que se ejecutó y se actualizó. Todo esto procurando mantener un buen control, gestionado por una buena gestión documental de los eventos que ocurren sobre la BD del SIB.

dTraza		
+id	int8	Nullable = false
#TipoTraza	*int8*	*Nullable = false*
#TipoEvento	*int8*	*Nullable = false*
#sf_guard_userid	*int8*	*Nullable = false*
DescripcionEvento	text	Nullable = true
created_at	timestamp	Nullable = false
updated_at	timestamp	Nullable = false

Figura 8. Tabla de trazas[12].

2.12 Seguridad en el diseño

La seguridad en el diseño está representada por el *plugin sfGuard*, el cual además de hacer uso del modelo patrón ya mencionado RBAC, para la asignación de tareas por medio de roles, posibilita usar en cualquier aplicación el ACL (*Access Control List* por sus siglas en Inglés) o lo que es igual, la autorización y autenticación de usuarios. Este *plugin* representa una capa más

[11] Tomado de la interfaz gráfica escogida.
[12] Creación en la herramienta de modelado.

de seguridad por encima de la seguridad predefinida por *Symfony*, debido a estas limitantes de accesibilidad destinadas a los usuarios.

Otro aspecto que se maneja en este *plugin* de seguridad, es el uso del algoritmo hash y la utilización del salt. El empleo del algoritmo hash para contraseñas es una de las consideraciones de seguridad que se deben llevar a la práctica al diseñar una aplicación que trabaje con contraseñas de los usuarios. Sin él, cualquier contraseña que se almacene en la BD de una aplicación podrá ser robada si la BD se ve comprometida, con lo que inmediatamente no sólo estaría comprometida la aplicación, sino también las cuentas de otros servicios de los usuarios. La aplicación del mismo a las contraseñas, antes de almacenarlas en la BD, dificulta al atacante determinar la contraseña original, ya que se estaría guardando en BD no la contraseña como tal, sino un código como resultado del algoritmo. Pese a esto, el atacante en un futuro podría comparar el hash resultante con la contraseña original y lograr sus intenciones. Es por eso que se emplea el salt con el objetivo de hacer los hash sean significantemente más complejos, más difíciles de crackear. Este es un dato añadido a la contraseña que pasa por el proceso de hash, provocando que la búsqueda de un hash resultante sea prácticamente imposible (The PHP Group, 2012).

2.13 Acceso a datos

El acceso a datos de la BD del SIB estará definido a nivel de aplicación mediante los *framework Symfony* junto con *Doctrine* por la parte de PHP y *Hibernate* con *Sprint* por *Java*, debido a las posibilidades que brinda el uso de estos, como la mejora de tiempo, el trabajo en tres capas: modelos-vista-controlador, el desarrollo rápido de aplicaciones y la reutilización. Estos permitirán que el acceso y la modificación de los datos almacenados se realice mediante objetos; de esta forma nunca se accederá de forma explícita a la BD, lo cual trae como ventaja un alto nivel de abstracción, permiten además una fácil portabilidad de los datos, encapsulación del acceso a los objetos de dominio persistentes, posibilidad de la persistencia de objetos transitorios y actualización de los existentes. El acceso a los datos se realizará además, a través de comunicaciones seguras utilizando el protocolo *Secure Socket Layer* (SSL), el cual establece un canal de comunicación segura, cifrando el intercambio de datos entre dos equipos. El único que podrá acceder a ella con todos los privilegios será el administrador de BD mediante la herramienta gráfica seleccionada para *PostgreSQL*, mientras que los funcionarios del SAIME y los usuarios de internet lo harán mediante la aplicación ubicada en los servidores de aplicación del SAIME. Por otro lado, los operadores de oficinas (actualmente 100 oficinas)

que serán los encargados de recibir a los usuarios que realizarán los pedidos de solicitud, accederán a la misma mediante servicios web que serán consumidos por los servidores de *Oracle* del SAIME.

2.14 Arquitectura de *software*

Luego de haber realizado el diseño previo y haber propuesto la necesidad de la utilización de otro servidor e identificado una herramienta para la replicación de los datos que permita la distribución de cargas y las multi-conexiones, se propone la siguiente arquitectura de *software* representada en el Anexo 17, con el objetivo de garantizar la disponibilidad de la información. La comunicación entre las aplicaciones clientes y ambos servidores se realizará a través de Pgpool-II (nodo), quien actuará como *software* intermediario entre la aplicación web y los servidores de BD (2 nodos), interceptando las sentencias SQL y enviándolas sincrónicamente hacia estos últimos; de esta forma se podrá balancear la carga de consultas debido a que será capaz de distribuirlas y se efectuará la replicación. Esta herramienta trabajará con los servidores simultáneamente, evitando la sobre carga de peticiones, posibilitando el aumento del rendimiento y de la disponibilidad de la BD.

2.15 Conclusiones parciales

En este capítulo se analizaron las características del negocio, describiendo los procesos y requisitos más importantes que exige el sistema para la BD, con el objetivo de identificar los elementos que tributan a las entidades del modelo de datos del mismo. Se declararon las necesidades técnicas del sistema que infligen en la BD, atendiendo a las exigencias del cliente, procurando siempre la satisfacción de sus exigencias, así como la persistencia y la seguridad de los datos. Se identificó al Pgpool-II como la herramienta para la replicación de datos del sistema SIB, estableciendo una comparación de las herramientas homólogas que se abordan en la tesis referenciada en el epígrafe 2.4. La realización de modelaciones previas de la BD del SIB, a partir de la descripción de los principales conceptos del negocio, permitió establecer un diseño final ajustado a las necesidades del sistema. Se realizó el proceso de normalización, con el objetivo de validar la lógica y estructura del modelo, en aras de lograr un buen diseño. El mismo arrojó una BD bien estructurada y de fácil entendimiento. El trabajo con técnicas como la utilización de patrones, permitieron ganar en organización y normalización. Y la inclusión de temas importantes como las restricciones de dominio, la indexación, la trazabilidad, la seguridad y el acceso a datos, permitió al diseñador de BD establecer pautas para fortalecer un buen diseño de la BD del SIB.

Capítulo III: Validación y optimización de la Base de Datos

Introducción

Las pruebas permiten comprobar si un sistema se comporta según lo esperado, por lo que aplicar pruebas de integridad al modelo, permitirá verificar su validez. La seguridad de las BD, es considerada como el proceso más importante a la hora de implementar soluciones que gestionan información sensible, ya que la pérdida o alteración de la información determinaría el nivel de competencia del producto. Es por esto que es necesario tener en cuenta estos aspectos, tanto en el diseño, como a la hora de realizar una instalación. El rendimiento de una BD permite disponer de una BD estable y rápida de gestionar, es por lo cual que se hace necesario establecer mecanismos que posibiliten estas características.

3.1 Validación teórica del diseño

La validación teórica del diseño incluye fundamentalmente un análisis muy detallado de la integridad de la información, característica altamente deseada, porque asegura la calidad de almacenamiento y disponibilidad de los datos. Con su tratamiento se evitan errores de entrada introducidos por los usuarios descuidados o cualquier otra circunstancia de intento de violar la información existente en la BD.

3.1.1 Integridad

La integridad es uno de los factores más importantes a la hora de realizar el diseño de una BD. Con el objetivo de garantizar la integridad de los datos almacenados de la BD del SIB, se realizaron un conjunto de pruebas, entre las que se encuentran las pruebas de integridad de entidad, integridad de dominio e integridad referencial. Se validó el funcionamiento de la misma en cada una de estas, las cuales arrojaron como resultado un promedio de velocidad de respuesta de consultas, de treinta milisegundos. Lo cual constituye un comportamiento previo adecuado y esperado ante la ocurrencia de operaciones, para una BD que tendrá un alto funcionamiento. Con las pruebas de integridad de entidad, se verificó que las claves primarias de las entidades del modelo, no pueden tomar valores nulos, ni duplicados; para lograr esto, *PostgreSQL* asigna automáticamente restricciones *UNIQUE* y *NOT NULL* sobre las claves primarias de las entidades. Un ejemplo de ello, se evidencia en el Anexo 18, donde, al tratar de insertar un valor nulo en la clave primaria de la entidad dsucursal (nosucursal), el gestor lanza un error; igualmente sucede si se trata de insertar una clave primaria duplicada, como se

muestra en el Anexo 19. En donde se gestiona la información de la tabla dproveedor, y donde el gestor lanza un error al determinar que ya existe un proveedor con esa llave primaria.

Los campos de las BD pueden ser restringidos por uno o varios dominios que definen y garantizan la validez de entradas para una columna determinada. Mediante las pruebas de integridad de dominio, se garantiza que cada valor que se desee insertar, cumpla con un conjunto de condiciones y reglas definidas en el negocio. Un ejemplo de ello, se puede observar en el Anexo 20, en el cual al tratar de realizar una inserción sobre la entidad sf_guard_user, el gestor lanza un error, indicando que el valor viola la restricción implementada para el dominio dom_username, el cual solo admite letras minúsculas de la a-z, números del 0-9 y algunos caracteres como el guión bajo, el punto y el signo de peso. También se puede exigir la integridad de dominio para restringir el tipo mediante tipos de datos, el formato mediante reglas y restricciones *CHECK*, o el intervalo de valores posibles mediante restricciones *FOREIGN KEY*, definiciones *DEFAULT*, definiciones *NOT NULL* (Macías, 2008).

"La integridad referencial garantiza que las relaciones entre filas de tablas relacionadas son válidas y que no se eliminan o se cambian datos relacionados de forma accidental" (MSDM Library, 2012). Con ella se garantiza que los valores de clave sean coherentes entre las diferentes tablas del modelo, o sea, al modificar una llave que es referenciada desde otras tablas, esta debe actualizarse en toda la BD, de igual forma, no deben existir referencias a valores inexistentes. Cuando se exige la integridad referencial, *PostgreSQL* impide de forma automática agregar o modificar filas en una tabla relacionada, si no existe una fila asociada en la tabla principal; de igual forma impide eliminar filas de una tabla principal, cuando haya filas de otras tablas asociadas a ella, a menos que las actualizaciones y eliminaciones se realicen en cascada. Un ejemplo de ello se demuestra en el Anexo 21, en el cual al tratar de eliminar una fila de la entidad dbancorecaudador, el gestor lanza un error, debido a que esta fila, con ese identificador, esta referenciada en la entidad dsucursal.

3.1.2 Análisis de redundancia de información

La redundancia de información se refiere a aquellos datos duplicados que generan inconsistencia en la BD, requiriendo más espacio en disco; sin embargo en proyectos grandes es imposible evitarla completamente, lo que a veces hace que se desee por cuestiones de rendimiento (González, 2009).

Los procesos de normalización y las pruebas de integridad realizadas a la BD de SIB, muestran como se ha trabajado para mejorar en buena medida el comportamiento de este parámetro y lo cual es demostrado por ejemplo en una de las figuras anexadas cuando no se permite atributos duplicados. En la BD del SIB, la información redundante hasta este momento es mínima (con tendencia a incrementarse por la existencia de tablas desnormalizadas en busca de rendimiento), gracias a que en la misma se analizaron las tablas con sus respectivas relaciones y se fueron eliminando consecuentemente todos los datos que podrían provocar anomalías en ella.

3.1.3 Análisis de seguridad de la BD

Las BD son víctimas constantemente de ataques por piratas informáticos, que tratan de acceder con el objetivo de robar o modificar alguna que otra información o simplemente violar la seguridad del sistema. Por lo cual es de vital importancia velar la seguridad de las mismas, protegiéndola contra accesos no autorizados o cualquier acción que puedan violar la integridad y confidencialidad de los datos. Los SGBD existentes garantizan en cierta medida el control de acceso por parte de los usuarios a la BD de acuerdo a los privilegios que tengan asignados y los lugares donde ellos se encuentren ubicados, así como los datos manejados deben de estar encriptados, garantizando así su seguridad.

En la confección de la BD del SIB se tuvieron en cuenta los principales términos referidos al control de acceso, integridad y disponibilidad de los datos. El *framework Symfony* que es con el cual se gestiona la aplicación, le ofrece a la BD del SIB una serie de aspectos de seguridad que son los que abogan por la seguridad de la información, los cuales se encuentran en uno de sus ficheros llamado, security.yml. Este fichero se crea para cualquier módulo que se esté trabajando, y posibilita la creación de permisos restringidos que requieren de la autenticación de los usuarios o no, en dependencia como se haya configurado el fichero. La realidad de la seguridad de la BD del SIB está definida en la dirección de los niveles de la aplicación (sistema) hacia el SGBD, quiere decir que las restricciones de seguridad van a estar abarcadas más en la aplicación que en la BD, debido a las condiciones en que se encontrará rodeado el servidor de BD. Otras barreras de seguridad de este *framework* en función de la protección de la información, son una serie de clases como *sfBasicSecurityUser* que proporciona métodos para gestionar la autenticación de usuario y autorización. Para la gestión de la autenticación de usuario, utiliza los métodos *isAuthenticated()* y *setAuthenticated(),* los cuales son métodos booleanos, y en cuanto a la autorización, *Symfony* posee un sistema de credenciales que es

bastante simple y poderoso. Una credencial puede representar todo lo que sea necesario para describir la seguridad al modelo de la aplicación (como grupos o permisos). Uno de los grandes puntos fuertes del *framework Symfony* son los *plugins*. Un ejemplo de esto lo constituye el *sfDoctrineGuardPlugin*, el cual proporciona una acción de ingreso en el módulo *sfGuardAuth* para autenticar a los usuarios. Es por esto que se decidió integrar un esquema de seguridad compuesto por 8 clases del *sfGuardPlugin*, ya que proporciona a la BD del SIB funciones para la validación de usuarios o grupos de usuarios (roles) con sus respectivos permisos a tener en cuenta a la hora de autenticarse al sistema y ejecutar acciones sobre este. Abordando más en la seguridad, se puede decir que *Symfony* está preparado para contrarrestar ataques que afectan a los datos, ya que cuando es ejecutada una de sus tareas, como es *generate*:*app*, esta prepara el sistema automáticamente para evitar ataques XSS y para prevenir ataque CSRF, los cuales van dirigido a las páginas web y que puede poner en peligro o en duda la información que se representa en estas. Cuando se trabaja con *Doctrine*, se generan en algunos casos sentencias preparadas que reducen la exposición a los ataques de inyecciones SQL por los usuarios de internet. También posibilita la encriptación de contraseñas, mediante la utilización del código *salt* y el algoritmo *hash*, aspectos explicados en el epígrafe 2.12.

Por otra parte el gestor a utilizar, *PostgresSQL*, materializa la seguridad mediante diferentes aspectos, en primer lugar garantiza la seguridad en la manipulación de los ficheros, es decir, se establece al usuario del sistema de dicho gestor como el propietario de los ficheros y le define los permisos de lectura, escritura y ejecución sobre el directorio DATA que contiene la información más crítica. Otro aspecto importante a tener en cuenta es la seguridad en los accesos de los clientes, ya que es necesario definir que usuarios se conectarán a las BD, desde qué equipo y por cuál método de autenticación lo harán. La seguridad en este nivel se realiza mediante el fichero *pg_hba.conf* (de sus siglas en inglés *host based authentication*), fichero de configuración para la autenticación de los clientes y usuarios, en él se define el acceso a las BD del clúster (Ver Anexo 22). El uso de conexiones SSL para cifrar las comunicaciones entre los clientes y el servidor permiten incrementar la seguridad, ya que la información viaja de forma cifrada. *PostgreSQL* contiene además privilegios de acceso y restricciones aplicables para cada objeto de la BD, como por ejemplo tablas, vistas y secuencias. Para esto hace uso de las sentencias REVOKE y GRANT en donde se revocan y otorgan privilegios respectivamente. La asignación de roles es otra práctica de seguridad ejecutada en el gestor, referente al control de acceso y a la operatividad en la BD. Aunque en el

sistema se establece esta práctica a nivel de aplicación, a nivel de BD se realizó también, como por ejemplo con el establecimiento del rol de operadora (autenticación mediante su contraseña), con sus respectivos permisos sobre las tablas que opera como dCliente, dCodigoSucursal, entre otras. Al hacer un buen diseño de la BD y especificar los tipos de datos y sus dominios (ver Anexo 14), *PostgreSQL* es capaz de garantizar que en la BD pueda controlarse la redundancia a su mínima expresión, manteniendo la consistencia y así conseguir la integridad de los datos.

Para fomentar el respaldo de la información se propone realizar copias de seguridad o backups, primeramente una general y luego incremental sobre esta, que sean guardadas en otra instancia o varias instancias de almacenamiento diferentes a los servidores de BD.

Otra característica que procura seguridad de los datos, y que no por mencionarla de último es menos importante, es que el sistema de BD del SIB está ubicado en un centro de datos que se rige por experimentadas políticas de seguridad, debido a que el SAIME es una organización que gestiona inmensos paquetes de información sumamente sensibles de la sociedad venezolana. Un ejemplo de estas políticas, se evidencia cuando dicho sistema será consultado mediante redes privadas, con lo cual se le aplican todas las directivas de seguridad y permisos de un ordenador en esa red privada.

3.2 Validación funcional

La validación funcional de una BD es la parte más importante de un proceso de diseño de este tipo de herramienta, ya que se da la posibilidad de chequear, si el trabajo realizado funciona realmente. Es en esta validación donde se muestran algunos resultados obtenidos con respecto al rendimiento y al correcto funcionamiento de la misma. Además da paso, a través de muestras, a hacer valoraciones y comparaciones de los tiempos de respuestas de las consultas y otros aspectos entre la BD normalizada y la optimizada. Las pruebas que se realizarán tienen como objetivo simular una carga de producción real y observar cómo se comportan ante esta carga. De este modo se comprueba que tan bien diseñada está la BD del SIB y cuáles son los cambios que deben ser aplicados para establecer una BD optimizada que satisfaga todos los requisitos del sistema sin violar la integridad de los datos.

3.2.1 Prueba de rendimiento

Las pruebas de rendimiento son muy importantes en el desarrollo de una aplicación, pues de ellas se asume, si la aplicación funciona, es estable y sólida. Estas pueden servir para diferentes propósitos:

- ✓ Pueden demostrar que el sistema cumple los criterios de rendimiento.
- ✓ Pueden comparar dos sistemas para encontrar cuál de ellos funciona mejor.
- ✓ Pueden asegurar el posterior funcionamiento del sistema y garantizar un correcto y eficiente servicio, carente de fallas, errores y retrasos a sus usuarios.
- ✓ Pueden medir qué partes del sistema o de carga de trabajo provocan que el conjunto rinda mal (González, 2011).

Es importante tener presente que los resultados de las pruebas y el rendimiento del sistema, están estrictamente ligados al hardware de la computadora que realizará la función de servidor.
A continuación se presentan algunas de estas pruebas de rendimiento, para demostrar con más claridad la factibilidad de la solución brindada.

Pruebas de Carga y de Stress

Debido a que el cliente cuenta con el equipamiento necesario para solventar cualquier tipo de problema por cuestión de almacenamiento y a que las principales dificultades podrían estar vinculadas a los pedidos simultáneos a la BD en horarios de mucha gestión, sumado a una BD bastante poblada, se decidió efectuar pruebas de carga y de stress. Estas se llevaron a cabo para determinar la solidez de la BD en los momentos de carga extrema y para ayudar a los administradores a determinar si esta, rendirá lo suficiente en caso de que la carga real supere a la carga esperada. Para ello se utilizó el *JMeter*, un *software* de distribución gratuita, una herramienta Java para llevar a cabo simulaciones de cargas sobre cualquier recurso *software*, destacándose por su versatilidad, estabilidad y por ser de uso gratuito. En estas pruebas se trabajó con la BD del SIB normalizada y con la BD del SIB optimizada, con el objetivo de establecer comparaciones entre ellas, y validar el proceso de optimización sobre la BD de manera funcional.
Lo primero que se realizó fue llenar las tablas más críticas de las dos BD, entre ellas la tabla dPago, a través de la herramienta *Data Generator* 2005 para *PostgreSQL*, para recrear un llenado voluminoso de la BD. Dicha tabla fue llenada con más de 2 000 000 de tuplas, lo que representan la cantidad de transacciones que se realizarían en poco menos de 6 meses.

Suponiendo que en este período ya las oficinas hayan llegado a 250 en todo el país y que la cantidad de operadores promedio en cada una de estas sea de 25, entonces se puede simular que en un determinado horario del día, bajo estas condiciones, estarían efectuando una petición equis sobre esta tabla, 6250 operadores del SAIME.

La idea de la prueba, es observar la respuesta del servidor ante esta situación extrema, y analizar, entre otras cosas y como aspectos más importantes, el comportamiento de las dos BD en cuestión de rendimiento y de variación de tiempo en la obtención de resultados.

Se decidió realizar un conjunto de tres pruebas a cada BD, donde la última en cada caso sería la situación anteriormente descrita. A continuación en la Tabla 3, se muestran las propiedades de los hilos de cada una de las pruebas, así como el significado y valor definido para cada uno de ellos.

- ✓ Número de hilos: Número de usuarios a simular.
- ✓ Período de subida (en segundos): Tiempo que debiera llevarle a *JMeter* lanzar todos los hilos (si se seleccionan 10 hilos y el periodo de subida es de 1 segundo, entonces cada hilo comenzará 0,1 segundo después de que el hilo anterior haya sido lanzado).
- ✓ Contador del bucle: Número de veces a realizar la prueba.

Tabla 3: Pruebas.

Propiedades de los hilos	Prueba 1	Prueba 2	Prueba 3
Número de hilos	2080	4170	6250
Período de subida (en seg)	1	1	1
Contador del bucle	1	1	1

En los informes de cada una de las pruebas se muestran los siguientes indicadores:

- ✓ Label: El nombre de la muestra (conjunto de muestras).
- ✓ # Muestras: El número de muestras de peticiones JDBC (igual al número de hilos por el contador del bucle).
- ✓ Media: El tiempo medio transcurrido para la obtención de un conjunto de resultados.
- ✓ Mediana: Mediana aritmética[13].
- ✓ Desviación: Desviación Estándar[14].
- ✓ Mín.: El mínimo tiempo transcurrido para las muestras de peticiones JDBC.

[13] Elemento de una serie ordenada de valores crecientes de forma que la divide en dos partes iguales, superiores e inferiores a él.
[14] Diferencia entre la medida de una magnitud y el valor de referencia.

- ✓ Máx.: El máximo tiempo transcurrido para las muestras de peticiones JDBC.
- ✓ Error %: Porcentaje de las peticiones con errores.
- ✓ Rendimiento: Rendimiento medido en base a peticiones por segundo/minuto/hora.
- ✓ Kb/seg: Rendimiento medido en Kilobytes por segundo.

Tabla 4: Resultados de las pruebas.

	BD del SIB normalizada			BD optimizada (Propuesta)		
Label	**Petición JDBC**	**Petición JDBC**	**Petición JDBC**	**Petición JDBC**	**Petición JDBC**	**Petición JDBC**
No.Muestras	2080	4170	6250	2080	4170	6250
Media	766454	889565	887779	448251	591325	501276
Mediana	1011380	1011484	1011461	510451	810880	954256
Desviación	1012113	1013428	1013392	511604	914817	1013014
Mín	972	994	1000	879	905	920
Máx	1013593	1014783	1015014	512104	1015809	1015706
% Error	100%	100%	100%	100%	100%	100%
Rendimiento	2.1/seg	4.1/seg	4.0/seg	5.3/seg	5.0/seg	5.1/seg
Kb/seg	0	0	0	0	0	0

Es importante resaltar, que estas pruebas fueron realizadas sobre servidores locales ubicados en computadoras de trabajo que no fueron configuradas a nivel de hardware para mejorar el rendimiento de los mismos. Las cuales contaban con las siguientes características: tarjeta madre P5LD2, disco duro de 160 GB y memoria RAM de 1 GB. Los resultados evidencian la efectividad de los cambios realizados sobre la BD del SIB, en aras de su optimización. En el análisis representado en la Tabla 4, la media, el tiempo promedio transcurrido para la obtención del conjunto de resultados de las pruebas realizadas en la propuesta, disminuyó en cada caso con respecto a la normalizada, lo cual demuestra una gestión rápida de la información a partir del proceso de optimización.

Con respecto a la mediana aritmética y la desviación estándar (significativos indicadores estadísticos), el primero constituye el valor que ocupa el lugar central cuando el conjunto de resultados está ordenado de menor a mayor mientras que el segundo es una medida que permite determinar el promedio aritmético de variación de los datos respecto a la media, se puede decir que las cifras obtenidas en la BD optimizada son mejores en todos los casos que las obtenidas en su homóloga. De igual manera ocurre con los valores encontrados en los tiempos mínimos y máximos transcurridos para las muestras de peticiones JDBC, y en cuanto

el porcentaje de error, se observa una constancia máxima de manera general, lo que indica que se debe seguir profundizando en la optimización de la BD.

Finalmente analizando los indicadores de rendimiento (rendimiento medido en base a peticiones por segundo y rendimiento medido en Kilobytes por segundo), se puede observar que aunque son valores bajos de rendimiento, los mejores números los exhiben las pruebas realizadas a la BD optimizada. Un aspecto a destacar es que en las pruebas a medida que aumenta el número de hilos o el número de usuarios a simular, las diferencias de los dos indicadores de rendimientos entre ambas BD aumenta también y en este caso siempre favorece a la BD optimizada que se propone en el presente trabajo de diploma.

3.3 Optimización

La optimización de la BD del SIB comienza en los primeros diseños, con el análisis de cuáles eran los datos que deberían persistir en un diseño final. Después se crearon relaciones y atributos, tratando de estructurar la información de forma tal que cumpliera con los requisitos del sistema. En el transcurso de este proceso, debido a diferentes cambios de conceptos en el negocio, se fueron eliminando tablas y sustituyéndolas por otras hasta llegar a un modelo lógico estable.

La normalización como paso continuo al anterior se llevó a cabo con el objetivo de minimizar la redundancia de la información al mínimo, no tratando de eliminarla totalmente ya que puede complejizar en gran medida el diseño y afectar el rendimiento del sistema. Como buena práctica a seguir luego de la normalización, se realizó la desnormalización a algunas tablas de la BD para ganar en rendimiento.

Como parte del proceso del diseño de los datos se determinó cuáles atributos debían ser indexados y cuáles no. Por esto se tuvo presente aspectos como cuáles eran los campos de mayor frecuencia de acceso, cuáles son utilizados comúnmente como criterio de comparación en consultas *WHERE*, entre otros.

La utilización de consultas preparadas, es una posibilidad que da *PostgreSQL* que permite tener procedimientos almacenados con la intensión de ahorrar una serie de pasos al ejecutar consultas. En el caso de la BD en cuestión esto se refleja con la utilización de *Triggers*, lo cual contribuye notablemente a la mejora del rendimiento.

La configuración objetiva de los parámetros del archivo de *postgresql.conf*, constituyó otra práctica de optimización para mejorar el rendimiento de la BD del SIB. Esta configuración tiene que ver con las características de hardware de la PC en que estará instalado el servidor, así

como con el SO con el que se operará. La configuración realizada sobre este fichero, contribuyó a mejorar el rendimiento expuesto en el proceso de pruebas. Se configuró el *shared_buffers* al 30% de la capacidad de la RAM (memoria de acceso aleatorio, igual a 1GB), este indica el número de bloques de memoria o buffers de 8KB que *PostgreSQL* reservará como zona de trabajo en el momento del arranque para procesar las consultas. Se llevó el *max_connections* hasta 100 conexiones permitidas en un mismo tiempo (máximo). El *effective_cache_size*, es el que le dice al optimizador de *PostgreSQL,* cuanta memoria tiene asignada el gestor para el almacenamiento de caché de datos y ayuda en el momento de la determinación de si crear o no un índice, se le dio hasta dos tercio de la RAM y el *work_men*: memoria destinada a cada usuarios concurrente, se valoró por debajo de la RAM entre el número de conexiones. Independientemente de que para el proceso de prueba resultó ventajosa esta configuración, al enfocarse en el caso en cuestión donde se utiliza el SO *Linux*, se deben tomar otras acciones. *Linux*, de forma predeterminada, tiene un segmento de memoria compartida de 32MB, lo cual para un servidor de BD al que se le va a dedicar 4GB de memoria compartida, significa un problema, ya que el espacio que requiere el número de buffers es superior al tamaño del segmento. Por ejemplo:

Segmentos: 4GB = 4294967296 KB shared_buffers: (4294967296 /8192) = 524288 KB
32MB = 33554432 KB : (33554432 /8192) = 4096 KB

La solución está en modificar el tamaño máximo del segmento de memoria compartida, y una forma de hacerlo es asignando un nuevo valor al parámetro del *kernel* SHMMAX a través de la siguiente línea de comando o con la posterior línea al archivo *sysctl.conf*:

```
shell> sysctl -w kernel.shmmax=4294967296
# sysctl -w kernel.shmmax=4294967296
```

Para hacer un cambio permanente, se debe añadir la siguiente línea al archivo *sysctl.conf* (su configuración puede variar). Este archivo se utiliza durante el proceso de arranque.

```
# echo "kernel.shmmax=4294967296" >> /etc/sysctl.conf
```

O también con la siguiente línea de código:

```
sysctl -p /etc/sysctl.conf
```

El planificador de consultas de *PostgreSQL* es quien se encarga de decidir cuál será la manera más eficiente de ejecutar una consulta y hacerlo en un tiempo lo suficientemente rápido. Para esto se traza planes de búsquedas aplicando métodos que pueden ser alterados a través de

sus parámetros, con el objetivo de establecer alternativos planes de búsquedas más rentables. Por ejemplo:

- ***enable_seqscan(boolean)****:* este parámetro habilita o deshabilita el uso de planes que utilicen búsquedas secuenciales. Por defecto su valor es *on*. Al situar este parámetro en *off*, simplemente se evita que el planeador de ejecución de consultas de *PostgreSQL* utilice búsquedas secuenciales para ejecutar una consulta si existe otra vía.

La previa práctica de manera eficiente de estas acciones, influye positivamente en la optimización de la BD del SIB. Pero optimizar una BD de este tipo, con sus características de seguridad, desempeño y confidencialidad, no debe limitarse a un pequeño grupo de medidas cuando existen otras prácticas. Es por eso que se proponen las siguientes para contribuir a la configuración adecuada de la BD en busca de rendimiento:

- ✓ **Optimizar las características del hardware del servidor:** esta es una práctica importante ya que un servidor de BD en su acción gestionar cientos y miles de transacciones para los usuarios, de manera que se permita la satisfacción de los mismos, está destinado a consumir grandes recursos para poder ofrecer sus servicios.
- ✓ **Utilizar conexiones persistentes o "*pooling software*"**: con un buen hardware, *PostgreSQL* puede soportar unos pocos cientos de conexiones con un costo considerable, lo que implica que establecer otras tantas, podría provocar la caída del sistema. La propuesta del Pgpool-II (*software* de agrupación de conexiones), es una solución a las miles de conexiones que si lugar a duda, permitirían que lo anterior pasara. Esta herramienta actúa como intermediario entre un grupo de servidores web y el servidor de BD, de manera que los servidores de conectan al *software* y este al servidor, el cual interpreta esto como una sola conexión, por lo que se nota una mejora en el rendimiento.
- ✓ **Hacer más con cada consulta**: los sistemas en *PostgreSQL* tienen un mejor rendimiento si están implementados con algunas consultas de gran tamaño y no con muchas pequeñas.
- ✓ **Conocer las particularidades de rendimiento entre las sentencias equivalentes del lenguaje**: usar la sentencia NOT IN es mejor que NOT EXISTS, no usar la cláusula HAVING si la condición deseada se puede expresar en la cláusula WHERE, evitar el uso de

la cláusula ORDER BY que puede ser costosa en tablas temporales, mejor rendimiento con la cláusula BETWEEN que con la cláusula LIKE, entre otras.

- ✓ **Programación de un *VACUUM FULL***: debido a que en la versión 8.4 de *PostgreSQL* está activado el *Autovacuum* por defecto y que no es aconsejable realizar el *VACUUM FULL* de manera cotidiana, se debe programar, por lo menos una vez al mes, pudiendo ser algún día no laborable por los bancos, un *VACUUM FULL* para realizar un mantenimiento de tablas más actualizado.
- ✓ **Reindexación:** es recomendable realizar esta práctica cotidianamente en los horarios muy poca gestión sobre la BD, para reorganizar los índices que son alterados por los procesos de limpieza (*Autovacuum, VACUUM FULL*).
- ✓ **Realizar *ANALIZE* de Tablas:** aunque el proceso de *Autovacuum* ejecuta este comando de manera automática, la realización de esta operación manualmente por parte del administrador sobre las tablas de mayor actividad en cuanto inserciones y actualizaciones, contribuye a enriquecer los planes de consultas para las tablas de la BD y en consecuente el rendimiento de la misma.

3.4 Conclusiones parciales

Un buen diseño lógico de BD puede ser la base de un rendimiento óptimo de la aplicación y de la BD (MSDM Library, 2012). Con el objetivo de validar la normalización realizada y garantizar la integridad de los datos almacenados de la BD del SIB, se realizaron las pruebas de integridad de entidad, de dominio y referencial. Las cuales arrojaron como resultado un buen diseño de tablas y relaciones, como el visto en la prueba de integridad referencial y una buena validación de datos, como el visto en la prueba de integridad de dominio. Tras el análisis de estos procesos se decretó que la BD contará con la mínima redundancia de información posible, debido a que por temas de rendimiento no se encuentra completamente a un mismo nivel de normalización. Se determinó que a pesar que la seguridad de la información y de la BD, dependerá en sentido general de factores externos, existen buenas estrategias que garantizarán la estabilidad de esta propiedad, así como del acceso a datos de la información. Las pruebas funcionales demostraron la disminución de los tiempos de respuestas y el mejoramiento del rendimiento de la BD tras el proceso de optimización del diseño realizado, así como que la misma, está lista para resistir situaciones de cargas extremas en horarios picos.

Conclusiones Generales

Se logró el diseño de una BD relacional, guiado por principios y fases de diseño, que responde a las exigencias del proyecto Integración Bancos con el SAIME. Se estableció el uso de herramientas adecuadas para desarrollo del trabajo que fuesen preferentemente libres, primero para establecer la línea base del proyecto y segundo para no estar en contradicción con las leyes del país en que reside el cliente. Se usaron patrones de diseño que permitirán la resistencia a los cambios que sufren los modelos, la disminución del riesgo de generar un mal diseño y de los tiempos de búsquedas de soluciones a problemas de este tipo.

Se lograron establecer niveles adecuados de seguridad ante el acceso a datos de los usuarios, el intercambio de datos entre equipos y ante la ejecución de operaciones sobre tablas, así como reglas de trazabilidad para registrar estas últimas. Por último, se logró optimizar la BD diseñando una buena estructura de datos, la cual fue validada teóricamente mediante pruebas de integridad, con el objetivo establecer garantías a la integridad y persistencia de los datos almacenados. También fue validada funcionalmente mediante pruebas de stress y de cargas, las cuales arrojaron como resultados una BD resistente y con buen rendimiento.

Recomendaciones

Al finalizar la investigación y cumplir el objetivo general de la misma, se recomienda:

- ✓ Utilización de *tablespaces*, para separar las tablas más utilizadas de las de menos gestión o de las que no varían en contenido, con el objetivo de agilizar la gestión de las mismas.
- ✓ Emplear una SAN (*Storage Área Network o* red de área de almacenamiento) para gestionar toda la información que se procesará en el SIB en busca de mejorar el rendimiento, la confidencialidad del sistema, la reducción de fallos por almacenamiento por parte de la BD, aumentar la disponibilidad, entre otros.
- ✓ Realizar un seguimiento de cómo se comporta el rendimiento con el tiempo y analizar cómo mejorarlo si es requerido, fundamentalmente en los momentos en que la BD alcance altos nivel de almacenado.
- ✓ Aplicar el particionamiento horizontal cuando existan tablas con gran volumen de datos, para contrarrestar el bajo rendimiento y realizar consultas con un ahorro significativo en el tiempo de respuesta.

Referencias Bibliográficas

Aguilar, Carlos Proal. 2011. Interactive and Cooperative Technologies Lab. *Modelado de datos.* [En línea] 2011. http://ict.udlap.mx/people/carlos/is341/bases02.html.

Alvarez, Sara. 2007. desarrolloweb.com. *Paso del modelo E/R al modelo relacional.* [En línea] 2007. http://www.desarrolloweb.com/articulos/paso-tablas-entidad-relacion.html.

buenastareas. 2012. buenastareas.com. [En línea] 2012. http://www.buenastareas.com/ensayos/Http-Www-Buenastareas-Com-Ensayos-Guia-2243910-Html/3638585.html.

Cano, Jeimy J. 2005. ISACA. *Trazabilidad de las Operaciones Electrónicas.* [En línea] 2005. http://www.isaca.org/Journal/Past-Issues/2005/Volume-6/Pages/JOnline-Trazabilidad-de-las-Operaciones-Electronicas-Un-Reto-para-la-Gerencia-de-Tecnologias-de-Informacion1.aspx. volumen 6.

Chávez, Mallelin Bolufe. 2011. monografias.com. *Frameworks para el desarrollo de aplicaciones con PHP.* [En línea] 2011. http://www.monografias.com/trabajos70/frameworks-desarrollo-aplicaciones-php/frameworks-desarrollo-aplicaciones-php.shtml.

Cidran, Javier. 2010. estudios-universidad. *MODELOS DE BASES DE DATOS.* [En línea] octubre de 2010. http://estudios-universidad.blogspot.com/2010_10_01_archive.html.

Cooperativa Centro de Estudios para la Educación Popular. 2009. cepep. *El Software Libre en Venezuela y la Soberanía Tecnológica.* [En línea] junio de 2009. http://es.scribd.com/doc/92632645/Software-Libre-en-Vzla-y-Soberania-Tecnologica.

duiops. 2009. duiops.net. *Manual de Microsoft Access 2003.* [En línea] 2009. http://www.duiops.net/manuales/access/access3.htm.

Ecured. 2010. Ecured. [En línea] 2010. http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos.

Figueroa, Daniel Sebastian Lopez. 2010. mitecnologico.com. [En línea] 2010. http://www.mitecnologico.com/Main/ModeloErYNormalizacion.

Figueroa, Henry Rafael Mendoza, Lemus,Roberto Alexander Posada. 2011. Bases de Datos de la Universidad de El Salvador. [En línea] 2011. http://basesdedatosues.blogspot.com/2011/06/nhibernate.html.

GARCIA, ROSA MARIA MATO. 2005. *Sistemas de bases de datos.* Ciudad de la Habana : Editorial Pueblo y Educación, 2005.

González, Irael Pérez, Díaz, Madelaine Sosa. 2011. *Propuesta de optimización de la base de datos para el proyecto Sistema de Informatización de la Gestión de las Fiscalías.* Cuidad de la Habana : s.n., 2011.

González, Maikel Hugo Álvarez. 2009. *DISEÑO DE LA BASE DE DATOS DEL SISTEMA DE INFORMACIÓN DE PERFORACIÓN DE POZOS (SIPP).* Ciudad de la Habana : s.n., 2009.

Graells, Dr. Pere Marquès. 2010. *INTRODUCCIÓN A LA INFORMÁTICA.* Barcelona : s.n., 2010.

Greg Smith, Robert Treat, Christopher Browne. 2012. PostgreSQL Wiki. [En línea] enero de 2012. http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server.

Guardado, Iván. 2010. web.Ontuts. [En línea] julio de 2010. http://web.ontuts.com/tutoriales/utilizando-doctrine-como-orm-en-php/.

Hoyo, Ireisy Bermúdez, Martínez, Lázaro Rubén García. 2011. *Propuesta de diseño de la base de datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela.* Ciudad de la Habana : s.n., 2011.

Infociberland. 2012. Infociberland. [En línea] 2012. http://infociberland.comxa.com/spring/config.php.

Jacobson, Ivar,Grady Booch y James Rumbaugh. 2000. *El proceso unificado de desarrollo del software.* Madrid : Pearson educacion SA, 2000, 2000.

Librosweb. 2009. libroweb.es. *Capítulo 1. Introducción a Symfony.* [En línea] 2009. http://www.librosweb.es/symfony_1_1/capitulo1/symfony_en_pocas_palabras.html.

liz555. 2011. buenastareas.com. [En línea] 2011. http://www.buenastareas.com/ensayos/Restricciones-De-Integridad/2070297.html.

Macías, Dayron Fernández. 2008. *Diseño de una base de datos para controlar la información de Unión de Jóvenes Comunistas en la UCI.* 2008.

masadelante. 2012. masadelante.com. [En línea] 2012. http://www.masadelante.com/faqs/base-de-datos.

Mata, Manel Pérez. 2009. TecnoRetales. [En línea] julio de 2009. http://www.tecnoretales.com/programacion/que-es-doctrine-orm/comment-page-1/.

Microsoft . 2007. Microsoft Office . [En línea] 2007. http://office.microsoft.com/es-es/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx.

MSDM Library. 2012. msdm. *Cómo funciona la integridad referencial con una base de datos de Oracle.* [En línea] 2012. http://msdn.microsoft.com/es-es/library/aa290252%28v=vs.71%29.aspx.

Obregón, Jesús Humberto Martínez. 2008. monografias.com. *Administración y diseño de bases de datos .* [En línea] abril de 2008. http://www.monografias.com/trabajos59/administracion-diseno-db/administracion-diseno-db.shtml.

Pacheco, Nacho. 2012. *Symfony, Libro.* [En línea] 2012. http://gitnacho.github.com/symfony-docs-es/book/doctrine.html.

Perdomo, Mariluz Hernández, Fernández, Enrique José González. 2009. *Guía para la optimización de servidores de bases de datos de PostgreSQL.* Cuidad de la Habana : s.n., 2009.

Pérez, Sergio Ernesto Gastón. 2005. mailxmail.com. *Curso componentes PC's. Base de datos.* [En línea] 2005. http://www.mailxmail.com/curso-componentes-pc-s/base-datos.

robealex1. 2011. buenastareas.com. [En línea] 2011. http://www.buenastareas.com/ensayos/Temmario-De-Administracion-De-Base-De/1713538.html.

Rodríguez, Héctor Julio. 2009. Portal web de la Pontificia Universidad Javeriana, Bogotá. [En línea] 2009. http://recursostic.javeriana.edu.co/wiki/index.php/Historia_de_las_bases_de_datos_en_Ciencia_de_la_Informaci%C3%B3n.

Silberschatz, Abraham. 2002. *Fundamentos de bases de datos 4ta Ed.* 2002. ISBN: 8448136543.

SpringSource. 2012. springsource. *The Standard for Enterprise Java Development.* [En línea] 2012. http://www.springsource.com/developer/spring.

The PHP Group. 2012. php. [En línea] 2012. http://php.net/manual/es/faq.passwords.php.

Torre, Andrés de la. 2010. monografias.com. [En línea] 2010. http://www.monografias.com/trabajos79/base-datos-orientadas-objetos/base-datos-orientadas-objetos.shtml.

Vargas, Guillermo de Jesús Saldivar. 2005. monografias.com. [En línea] 2005. http://www.monografias.com/trabajos30/base-datos/base-datos.shtml.

Bibliografía

Aguilar, Carlos Proal. 2011. Interactive and Cooperative Technologies Lab. *Modelado de datos.* [En línea] 2011. http://ict.udlap.mx/people/carlos/is341/bases02.html.

Alvarez, Sara. 2007. desarrolloweb.com. *Paso del modelo E/R al modelo relacional.* [En línea] 2007. http://www.desarrolloweb.com/articulos/paso-tablas-entidad-relacion.html.

ArPUG. *Grupo de usuarios Postgres SQL de Argentina.* [En línea] http://www.arpug.com.ar/trac/wiki/PgAdmin.

Batista, Arcel Labrada, Damir Góngora Mora. 2007. *Sistema de gestión de informacion de la Facultad 8. Diseño de la Base de Datos.* Ciudad de La Habana : s.n., 2007.

buenastareas. 2012. buenastareas.com. [En línea] 2012. http://www.buenastareas.com/ensayos/Http-Www-Buenastareas-Com-Ensayos-Guia-2243910-Html/3638585.html.

2012. BuenasTareas.com. [En línea] 2012. http://www.buenastareas.com/ensayos/Base-De-Datos-Multidimensionales/190823.html.

C. Díaz. 2005. *CIFI-INFORMATICA– PROCURADURIA GENERAL DE LA NACION.* Colombia : s.n., 2005.

Cano, Jeimy J. 2005. ISACA. *Trazabilidad de las Operaciones Electrónicas.* [En línea] 2005. http://www.isaca.org/Journal/Past-Issues/2005/Volume-6/Pages/JOnline-Trazabilidad-de-las-Operaciones-Electronicas-Un-Reto-para-la-Gerencia-de-Tecnologias-de-Informacion1.aspx. volumen 6.

CAVSI.com. [En línea] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

Chávez, Mallelin Bolufe. 2011. monografias.com. *Frameworks para el desarrollo de aplicaciones con PHP.* [En línea] 2011. http://www.monografias.com/trabajos70/frameworks-desarrollo-aplicaciones-php/frameworks-desarrollo-aplicaciones-php.shtml.

Cidran, Javier. 2010. estudios-universidad. *MODELOS DE BASES DE DATOS.* [En línea] octubre de 2010. http://estudios-universidad.blogspot.com/2010_10_01_archive.html.

Cooperativa Centro de Estudios para la Educación Popular. 2009. cepep. *El Software Libre en Venezuela y la Soberanía Tecnológica.* [En línea] junio de 2009. http://es.scribd.com/doc/92632645/Software-Libre-en-Vzla-y-Soberania-Tecnologica.

Documentación de Referencia de Hibernate. Version: 3.0.5.

duiops. 2009. duiops.net. *Manual de Microsoft Access 2003.* [En línea] 2009. http://www.duiops.net/manuales/access/access3.htm.

Ecured. 2010. Ecured. [En línea] 2010. http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_de_bases_de_datos.

Espinoza, Humberto. 2005. Scribd.com. *PostgreSQL. Una Alternativa de DBMS Open Source.* [En línea] 2005. http://es.scribd.com/doc/60476868/PresentacionES-PSQL.

Figueroa, Daniel Sebastian Lopez. 2010. mitecnologico.com. [En línea] 2010. http://www.mitecnologico.com/Main/ModeloErYNormalizacion.

Figueroa, Henry Rafael Mendoza, Lemus,Roberto Alexander Posada. 2011. Bases de Datos de la Universidad de El Salvador. [En línea] 2011. http://basesdedatosues.blogspot.com/2011/06/nhibernate.html.

Free Download Manager. 2007. Free Download Manager. [En línea] 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

GARCIA, ROSA MARIA MATO. 2005. *Sistemas de bases de datos.* Ciudad de la Habana : Editorial Pueblo y Educación, 2005.

González, Irael Pérez, Díaz, Madelaine Sosa. 2011. *Propuesta de optimización de la base de datos para el proyecto Sistema de Informatización de la Gestión de las Fiscalías.* Cuidad de la Habana : s.n., 2011.

González, Maikel Hugo Álvarez. 2009. *DISEÑO DE LA BASE DE DATOS DEL SISTEMA DE INFORMACIÓN DE PERFORACIÓN DE POZOS (SIPP).* Ciudad de la Habana : s.n., 2009.

Graells, Dr. Pere Marquès. 2010. *INTRODUCCIÓN A LA INFORMÁTICA.* Barcelona : s.n., 2010.

Greg Smith, Robert Treat, Christopher Browne. 2012. PostgreSQL Wiki. [En línea] enero de 2012. http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server.

Guardado, Iván. 2010. web.Ontuts. [En línea] julio de 2010. http://web.ontuts.com/tutoriales/utilizando-doctrine-como-orm-en-php/.

Hoyo, Ireisy Bermúdez, Martínez, Lázaro Rubén García. 2011. *Propuesta de diseño de la base de datos para los módulos de solicitud y enrolamiento del Sistema de Emisión de Pasaportes Diplomáticos, de Servicio y Acreditaciones de la República Bolivariana de Venezuela.* Ciudad de la Habana : s.n., 2011.

Infociberland. Infociberland. [En línea] http://infociberland.comxa.com/spring/config.php.

Jacobson, Ivar,Grady Booch y James Rumbaugh. 2000. *El proceso unificado de desarrollo del software.* Madrid : Pearson educacion SA, 2000, 2000.

2009. Jotadeveloper Blog. [En línea] julio de 2009. http://blog.jotadeveloper.com/2009/07/03/symfony-el-struts-e-hibernate-del-php/.

Librosweb. 2009. libroweb.es. *Capítulo 1. Introducción a Symfony.* [En línea] 2009. http://www.librosweb.es/symfony_1_1/capitulo1/symfony_en_pocas_palabras.html.

liz555. 2011. buenastareas.com. [En línea] 2011. http://www.buenastareas.com/ensayos/Restricciones-De-Integridad/2070297.html.

Macías, Dayron Fernández. 2008. *Diseño de una base de datos para controlar la información de Unión de Jóvenes Comunistas en la UCI.* 2008.

Maldonado, Daniel Martin. 2008. AplicacionesEmpresariales.com. *Gestionando las Bases de Datos MySQL con phpMyAdmin.* [En línea] abril de 2008. http://www.aplicacionesempresariales.com/gestionando-las-bases-de-datos-mysql-con-phpmyadmin.html.

Marañón, Gonzalo Álvarez. 1999. Criptonomicón. [En línea] 1999. http://www.iec.csic.es/criptonomicon/cookies/ejemplos.html.

masadelante. 2012. masadelante.com. [En línea] 2012. http://www.masadelante.com/faqs/base-de-datos.

Mata, Manel Pérez. 2009. TecnoRetales. [En línea] julio de 2009. http://www.tecnoretales.com/programacion/que-es-doctrine-orm/comment-page-1/.

Microsoft . 2007. Microsoft Office . [En línea] 2007. http://office.microsoft.com/es-es/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx.

MiTecnologico.com. *Diseño Bases De Datos.* [En línea] http://www.mitecnologico.com/Main/Dise%F1oBasesDeDatos.

MSDM Library. 2012. msdm. *Cómo funciona la integridad referencial con una base de datos de Oracle.* [En línea] 2012. http://msdn.microsoft.com/es-es/library/aa290252%28v=vs.71%29.aspx.

MSDN Library. 2012. msdn. *Normalización.* [En línea] 2012. http://msdn.microsoft.com/es-es/library/ms191178%28v=sql.105%29.aspx.

MultiMania. MultiMania. *DISEÑO DE BASES DE DATOS RELACIONALES.* [En línea] http://usuarios.multimania.es/cursosgbd/UD4.htm.

NETRONYCS. NETRONYCS.COM. [En línea] http://www.netronycs.com/modelos_de_base_de_datos.html.

Obregón, Jesús Humberto Martínez. 2008. monografias.com. *Administración y diseño de bases de datos .* [En línea] abril de 2008. http://www.monografias.com/trabajos59/administracion-diseno-db/administracion-diseno-db.shtml.

Online, Centro de Soporte y Documentación. 2011. Centro de Soporte y Documentación Online. *Asistencia técnica entregada por profesionales.* [En línea] 2011. http://www.superhosting.cl/manuales/sesiones-en-php-ii.html.

Oracle. 2011. oracle.com. *Oracle Database.* [En línea] 2011. http://www.oracle.com/us/products/database/overview/index.html.

Pacheco, Nacho. 2012. *Symfony, Libro.* [En línea] 2012. http://gitnacho.github.com/symfony-docs-es/book/doctrine.html.

Paré, Rafael Camps. 2011. *Introducción a las bases de datos.* Barcelona,España : s.n., 2011.

Pecos, Daniel. 2002. PostGreSQL vs. MySQL. [En línea] 2002. http://danielpecos.com/docs/mysql_postgres/x159.html.

Perdomo, Mariluz Hernández, Fernández, Enrique José González. 2009. *Guía para la optimización de servidores de bases de datos de PostgreSQL.* Cuidad de la Habana : s.n., 2009.

Pérez, Sergio Ernesto Gastón. 2005. mailxmail.com. *Curso componentes PC's. Base de datos.* [En línea] 2005. http://www.mailxmail.com/curso-componentes-pc-s/base-datos.

Peter. Rob.Coronel, Carlos. *Sistemas de Bases de Datos.Diseño, implementación y administración.5ta ed. s.l. : Thomson Learning. Inc, 2002. 9706862862.*

robealex1. 2011. buenastareas.com. [En línea] 2011. http://www.buenastareas.com/ensayos/Temmario-De-Administracion-De-Base-De/1713538.html.

Rodríguez, Héctor Julio. 2009. Portal web de la Pontificia Universidad Javeriana, Bogotá. [En línea] 2009. http://recursostic.javeriana.edu.co/wiki/index.php/Historia_de_las_bases_de_datos_en_Ciencia_de_la_Informaci%C3%B3n.

Santiesteban, Elizabeta Rojas ,Angel Erlys Boloy Boado. 2010. *Arquitectura y diseño de la Base de Datos Única de los Sistemas de Identificación, Inmigración y Extranjería de la República de Cuba.* Ciudad de la Habana : s.n., 2010.

sentido web. 2007. sentido web. Referencias y explicaciones sobre desarrollo web, PHP, Ajax, XHTML, MySQL . [En línea] 2007. http://sentidoweb.com/2007/03/29/phppgadmin-administra-postgresql-desde-la-web.php.

Silberschatz, Abraham. 2002. *Fundamentos de bases de datos 4ta Ed.* 2002. ISBN: 8448136543.

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

Softline. Un catálogo de software. **Mauricio Cuéllar Rojas, Igor Petlyakov. 2011.** Bogotá, Colombia : s.n., 2011.

Soto, Lauro. 2010. mitecnologico.com. *Especificaciones De Requerimientos.* [En línea] 2010. http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

SpringSource. 2012. springsource. *The Standard for Enterprise Java Development.* [En línea] 2012. http://www.springsource.com/developer/spring.

The PHP Group. 2012. php. [En línea] 2012. http://php.net/manual/es/faq.passwords.php.

TiendaLinux. 2003. TiendaLinux.com. *Ventajas de PostgreSQL.* [En línea] mayo de 2003. http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html.

Torre, Andrés de la. 2010. monografias.com. [En línea] 2010. http://www.monografias.com/trabajos79/base-datos-orientadas-objetos/base-datos-orientadas-objetos.shtml.

Ubuntu. 2008. Guía Documentada para Ubuntu. [En línea] marzo de 2008. http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

Vargas, Guillermo de Jesús Saldivar. 2005. monografias.com. [En línea] 2005. http://www.monografias.com/trabajos30/base-datos/base-datos.shtml.

Glosario de Términos

ACID es un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. Así pues, si un sistema de gestión de BD es ACID *compliant*, quiere decir que el mismo cuenta con las funcionalidades necesarias para que sus transacciones tengan las características ACID. En concreto ACID es un acrónimo de *Atomicity, Consistency, Isolation and Durability*: Atomicidad, Consistencia, Aislamiento y Durabilidad en español.

API o Interfaz de programación de aplicaciones (del inglés *Application Programming Interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objeto) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas vulgarmente "librerías").

El **CSRF** (del inglés *Cross-site request forgery* o falsificación de petición en sitios cruzados) es un tipo de programa informático malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía. Esta vulnerabilidad es conocida también por otros nombres como XSRF, enlace hostil, ataque de un *click*, cabalgamiento de sesión, y ataque automático.

E. Codd: científico informático inglés (1923 - 2003), conocido por sus aportes a la teoría de BD relacionales. Inventó el modelo relacional, el modelo de BD más usado hoy en día y para muchas personas, el único que conocen.

Framework: en el desarrollo de *software*, un *framework* o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de *software* concretos, con base a la cual otro proyecto de *software* puede ser más fácilmente organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

front-end y back-end*:** en diseño de *software* el ***front-end es parte del *software* que interactúa con el o los usuarios y el ***back-end*** es la parte que procesa la entrada desde el *front-end.* La separación del sistema en "*front-ends*" y "*back- ends*" es un tipo de abstracción que ayuda a

mantener las diferentes partes del sistema separadas. La idea general es que el *front-end* sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y procesarlas de una manera conforme a la especificación que el *back-end* pueda usar. La conexión del *front-end* y el *back-end* es un tipo de interfaz. En el diseño web se hace referencia a la visualización del usuario navegante por un lado (*front-end*), y del administrador del sitio con sus respectivos sistemas por el otro (back-end).

GUI: la interfaz gráfica de usuario, conocida también como GUI (en inglés *graphical user interface*) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

IBM: es una empresa multinacional estadounidense que fabrica y comercializa *hardware* y *software* para computadoras, y ofrece servicios de infraestructura, alojamiento de internet, y consultoría en una amplia gama de áreas relacionadas con la informática.

JDBC: *Java Database Connectivity*, más conocida por sus siglas **JDBC** , es una API que permite la ejecución de operaciones sobre BD desde el lenguaje de programación *Java*, independientemente del SO donde se ejecute o de la BD a la cual se accede, utilizando el dialecto SQL del modelo de BD que se utilice.

***Kernel*:** es un *software* que constituye la parte más importante del SO, es el encargado de gestionar recursos a través de servicios de llamada al sistema y también se encarga de decidir qué programa podrá hacer uso de un dispositivo de *hardware* y durante cuánto tiempo.

MS Visio: *Microsoft Visio* es un *software* de dibujo vectorial para *Microsoft Windows*. *Visio* comenzó a formar parte de los productos de *Microsoft* cuando fue adquirida la compañía *Visio* en el año 2000.

OpenSource: *software* que se define por la licencia que lo acompaña, que garantiza a cualquier persona el derecho de usar, modificar y redistribuir el código libremente. *Open Source* es una marca de certificación propiedad de la *Open Source Initiative*. Los desarrolladores que diseñan *software* para ser compartido, mejorado y distribuido libremente, pueden usar la marca registrada *Open Source* si sus términos de distribución se ajustan a los términos de la marca.

Oracle: es un SGBD objeto-relacional, desarrollado por *Oracle Corporation*. Se considera como una de los sistemas de BD más completos y destacado, por su soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma.

ORM o (*Object Relation Mapper, en inglés*): es una técnica de programación que nos permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una BD relacional, es decir, las tablas de cualquier BD pasan a ser clases y los registros objetos que se pueden manejar con facilidad (Mata, 2009).

Procedimiento almacenado: es un programa (o procedimiento) el cual es almacenado físicamente en una BD.

SFTP: *SSH File Transfer Protocol* (también conocido como SFTP o *Secure File Transfer Protocol*) es un protocolo del nivel de aplicación que proporciona la funcionalidad necesaria para la transferencia y manipulación de archivos sobre un flujo de datos fiable. Se utiliza comúnmente con SSH para proporcionar la seguridad a los datos, aunque permite ser usado con otros protocolos de seguridad. Por lo tanto, la seguridad no la provee directamente el protocolo SFTP, sino SSH o el protocolo que sea utilizado en su caso para este cometido.

UUID: identificador universalmente único (o *UUID*) es un identificador estándar usado en el desarrollo de *software*. Un *UUID* es un número de 16-byte (128-bit). En su forma canónica, un UUID consiste de 32 dígitos hexadecimales. Por ejemplo: **550e8400-e29b-41d4-a716-446655440000**

XSS: del inglés *Cross-site scripting* es un tipo de inseguridad informática o agujero informático, basado en la explotación de vulnerabilidades del sistema de validación de HTML incrustado.