Universidad de las Ciencias Informáticas

Facultad 3

Diseño e implementación de la base de datos de los módulos Dictámenes, Procesos con menores y Revisiones laborales.

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

Autor:

Anier Vázquez Morales.

Tutor: Ing. Yenier Figueroa Machado

Co-tutor(es): Ing. Manuel Álvarez Alonso

Ing. José Carlos Pupo Acosta

Ciudad de La Habana, Curso 2012-2013

"El futuro de nuestra patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que más estamos sembrando; lo que más estamos sembrando son oportunidades a la inteligencia (...)"

Fidel Castro Ruz.

Declaración de Autoría

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2013.

_____	_____
Firma del autor	**Firma del tutor**
Anier Vázquez Morales.	Ing. Yenier Figueroa Machado

Datos de Contacto

Datos de tutor:

Nombre: Ing. Yenier Figueroa Machado

Ingeniero en Ciencias Informáticas. 5 años de experiencia laboral. Jefe del proyecto Sistema de Informatización de Gestión de las Fiscalías fase II.

Correo electrónico: yfigueroa@uci.cu

Datos de Co-Tutor:

Nombre: Ing. Manuel Álvarez Alonso

Ingeniero en Ciencias Informáticas.

Correo electrónico: malvareza@uci.cu

Datos de Co-Tutor:

Nombre: Ing. José Carlos Pupo Acosta

Ingeniero en Ciencias Informáticas.

Correo electrónico: jcpupo@uci.cu

Datos del Autor:

Nombre: Anier Vázquez Morales.

Correo electrónico: avmorales@estudiantes.uci.cu

Agradecimientos

A toda mi familia, en especial a mi mamá y a mi papá, que me han dado todo su apoyo incondicional a lo largo de mi vida y sobre todo desde que tome esta carrera. A mi mamá por todo el amor y los sabios consejo que siempre me brindo, a mi papá por su ejemplo un digno ejemplo de sacrificio y por siempre ser mi guía e inspiración. A mis hermanos, que han sido hermanos y amigo, dignos ejemplos a seguir y por siempre darme su apoyo incondicional durante esta etapa de mi vida, estoy muy orgulloso de que estén aquí. Gracias por todo.

A Manolo y Michel que han estado conmigo en las buenas y en las malas durante el transcurso de la carrera.

A mis amigos, que más que eso han sido hermanos, al piry, argilagos, javy y a yandri que siempre me brindo su amistad desde que llegue a esta hermosa escuela.

Al negrito lindo de Abel que siempre estuvo conmigo y nunca me dijo que no durante el desarrollo de este trabajo, muchas gracias hermano.

A esos dos tutores que sin importar los momento siempre estuvieron ahí y en especial a manolito que desde el primer momento que le pregunte por un trabajo de tesis me ofreció su mano y nunca me dijo que no, a las personas de mi tribunal que con sus regaños constructivos supieron se una guía para este resultado.

A todas esas personas que siempre se preocuparon por mi tesis y siempre me ofrecieron su mano de una forma u otra para que este trabajo tuviera el resultado que hoy tiene.

A mis compañeros que me ayudaron de muchas maneras dándome su confianza para cumplir con esta meta. A los profesores que me impartieron clases en estos años.

Los compañeros de aula que durante 5 años hemos estado pasando noches sin dormir cuando teníamos prueba, a Félix, Juan Carlos y a todos los que se me quedan muchas gracias.

Al grupo de baile Los MoonWalkers, que me enseñaron que para que las cosas salgan bien llevan sacrificio y compromiso. Heiler, Alexis y todos los demás.

A la revolución por darme la oportunidad de estudiar en esta escuela.

Dedicatoria

A mi mamá por todo su amor, su cariño, le agradezco las horas de su vida que dedico a mí y que dedicado a mi formación. Ejemplo de mujer y madre para mí. La madre más linda del mundo.

A mi papichorrro, por ser mis inspiración, mi guía, un ejemplo digno a seguir, por todo su sacrificio y por todo el tiempo que dedico a mí, sin él no podría ser la persona que soy, padre y amigo, quien me ha enseñado que con sacrificio se logra todo en la vida, por lo responsable y humano que es, nunca voy a olvidar lo que significa la familia para ti.

A mis hermanos, quienes siempre me apoyaron y supieron ser una guía en mi formación, ser mi ejemplo a seguir, estoy orgulloso de tener dos hermanos como ustedes.

A toda mi familia por su apoyo y preocupación para que todo me saliera bien. A mis verdaderos amigos por estar ahí en el momento que los necesitaba, a Manolo, Michel, Piry, Argilago, Yandri, Alexis, Abel siempre los tendré presentes donde quiera que esté, gracias por ser mis amigos.

Resumen

Las aplicaciones de gestión en la actualidad tienen la necesidad de responder a la gestión de datos, los cuales deben ser almacenados de forma segura y con integridad, sin depender del volumen que presenten para las entidades. Para la Fiscalía General de la República se le desarrolla un sistema de gestión en la Universidad de las Ciencias Informáticas, Sistema de Informatización de la Gestión de las Fiscalías (SIGEF II), el cual está necesitado de una base de datos que gestione y almacene la información de los módulos Dictámenes, Proceso con Menores y Revisiones Laborales. Este trabajo tiene como objetivo desarrollar el diseño y la implementación de una base de datos para los módulos Dictámenes, Proceso con Menores y Revisiones Laborales que contribuya a la seguridad de la información y la integridad de los datos. Para llegar a la solución se toma como guía de desarrollo el framework Dbplanning (Database Planning) Framework, como herramientas, el sistema gestor de base de datos PostgreSQL 9.1, Visual Paradigm 8.0 para el modelado y el NetBeans 7.2 para describir el acceso a datos del framework para el mapeo relacional de objetos Doctrine 2.3.2.

PALABRAS CLAVE: almacenar, gestionar, integridad

Índice de Contenido

Índice de Figuras.

Índice de Tabla

Introducción

La introducción masiva de las nuevas Tecnologías de la Información y las Comunicaciones (TIC) han influenciado fuertemente en la estructura y dinámica de los procesos económicos y sociales, redefiniendo aceleradamente las formas de producir, vender y competir en prácticamente todos los sectores de bienes y servicios; así como en las nuevas formas de interacción y comunicación entre las personas y organismos de la sociedad.

Desde hace algunos años, Cuba se ha trazado la meta de informatizar la sociedad con el objetivo de elevar el nivel cultural del pueblo y así lograr una mayor eficiencia en los servicios que brinda nuestro país. La sociedad cubana ha desarrollado los sistemas informáticos con el fin de informatizar diferentes tareas sociales, políticas y económicas. La Universidad de las Ciencias Informáticas (UCI) es uno de los principales centros que ha contribuido al desarrollo informático en Cuba. Enfocado en producir aplicaciones y servicios informáticos tanto a empresas nacionales como extranjeras, a partir de la vinculación estudio-trabajo como modelo de formación, dándole solución a diferentes problemas existentes en el país en la rama de la informática.

En la UCI, el Centro de Gobierno Electrónico (CEGEL), perteneciente a la facultad 3, desde sus inicios ha dedicado sus esfuerzos a la investigación y desarrollo de sistemas informáticos para el área del gobierno electrónico, incorporando resultados importantes en el campo que contiene el contacto tecnológico y las ciencias del derecho, donde la Fiscalía General de la República (FGR), encargada de controlar el estricto cumplimiento de la constitución del país, descubriendo respuestas al ahorro de esfuerzos y tiempo, encontrándose inmersa en la tarea de mejorar sus procesos para un mejor desempeño de su función social y política, ha solicitado la creación de un sistema que le permita gestionar sus procesos.

La producción de software y la informatización de diferentes áreas del país recae principalmente en los proyectos productivos de la UCI, siendo uno de estos el Sistema de Informatización de la Gestión de las Fiscalías II (SIGEF II), creado con el objetivo de realizar una aplicación web que mejore la calidad de tramitación, supervisión y control en tiempo de los procesos fiscales cubanos teniendo como alcance obtener una fuerza fiscal óptima y eficiente con mayor economía y seguridad en cada uno de sus procesos.

El proyecto SIGEF está compuesto por subsistemas o áreas que incluyen los módulos correspondientes a los procesos que representan en la FGR. SIGEF se ha dividido teniendo en cuenta las distintas áreas con la que cuenta la fiscalía cubana. Dentro de estos subsistemas encontramos Protección de los Derechos Ciudadanos (PDC) que engloba los módulos de Reclamaciones, Quejas, Atención a la población, Procesos Civiles, Procesos Administrativos,

Proceso Laborales, Proceso de Familia y Procesos con Menores. Otro subsistema es el área de Control de la Legalidad de los Establecimientos Penitenciarios (CLEP), que es el área de la FGR que controla los aspectos relacionados con las visitas de inspección a los establecimientos penitenciarios, así como las quejas, peticiones o denuncias de los familiares del implicado en un delito, del propio interno o detenido, además de la documentación que se genera en estos procesos.

Algunos de los módulos derivados en esta área son:

- ✓ Procesos con menores.
- ✓ Procesos laborales.
- ✓ Dictámenes.

Actualmente llevar a cabo esta gama de procesos necesita una cantidad considerable de fiscales. Debido a esto, se hace engorroso el trabajo, la realización y el control de todos los procesos, por lo que requieren más tiempo del esperado, ya que se genera un gran cúmulo de documentos diariamente, viéndose afectado el análisis de los mismos de una manera más eficiente y efectiva. Por tanto la mayor problemática se enfoca en toda la información manuscrita, haciéndose difícil la búsqueda inmediata de una documentación determinada. Pasados los cinco años de permanencia de estos documentos en la fiscalía se procede a la destrucción de los que cuya información, ya no mantiene la relevancia como para su conservación, imposibilitando su acceso en tiempos futuros. Son estos los factores que hacen real la necesidad por parte de la FGR de informatizar los procesos fiscales, en este caso Quejas reclamaciones y denuncias, Dictámenes, Procesos con menores y Procesos laborales.

Por todo lo antes planteado se necesita crear un mecanismo que facilite el registro de esta información, agilice y haga más efectivo el trabajo de los fiscales a lo largo de todo el Territorio nacional. Lo que hace necesario la implementación de nuevas funcionalidades al sistema ya existente (SIGEF I), logrando así una solución eficiente a los problemas anteriormente detectados. Teniendo en cuenta que el sistema ya implementado se rige por una arquitectura definida en el proyecto, se preverá que las nuevas funcionalidades logren insertarse perfectamente en el mismo, obteniendo como resultado mejores servicios y funcionalidades haciendo uso de las TIC que juegan un papel protagónico en este campo para la informatización y automatización de la mayoría de las esferas y procesos de la producción y los servicios.

La informatización de los Procesos de Dictámenes, Procesos con menores y Procesos laborales significará una reducción del tiempo para la realización de las diligencias fiscales en estos procesos, resultando más fácil la búsqueda de documentos en formato digital y por tanto una mayor eficiencia.

En estos momentos no se cuenta con una estructura que permita la realización y el control de todos los procesos, ya que se genera un gran cúmulo de documentos diariamente, viéndose afectado el análisis de los mismos de una manera más eficiente y efectiva. Por lo que surge el siguiente **problema a resolver**: ¿Cómo garantizar el almacenamiento de la información gestionada por los módulos Dictámenes, Procesos con menores y Procesos laborales de forma que se contribuya a la seguridad de la información y la integridad de los datos?, teniendo como:

Objetivo General: Realizar el diseño y la implementación de una BD para los módulos Dictámenes, Procesos con menores y Procesos laborales, que contribuya a la seguridad de la información y la integridad de los datos, definiendo como:

Objeto de Estudio: Desarrollo de base de datos (BD) operacionales, enmarcado en el:

Campo de Acción: Desarrollo de base de datos para sistemas de gestión fiscal.

Al terminarse de analizar los elementos anteriores se plantea la siguiente idea a defender: con el desarrollo del diseño e implementación del modelo de datos de los Dictámenes, Procesos con menores y Procesos laborales, contribuyendo al almacenamiento y gestión de los datos que generan los procesos del área de Quejas reclamaciones y denuncias, Dictámenes, Procesos con menores y Procesos laborales y los datos que sean similares en otras áreas.

Para darle cumplimiento al objetivo general propuesto se definieron los siguientes Objetivos Específicos:

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Diseñar la propuesta de la base de datos para los módulos Dictámenes, Procesos con menores y Procesos laborales.
- ✓ Implementar la propuesta de la base de datos para los módulos Dictámenes, Procesos con menores y Procesos laborales.
- ✓ Validar la solución propuesta a través de pruebas de carga y rendimiento.

Para el desarrollo del trabajo, se hace necesario definir las siguientes tareas de investigación:

- ✓ Estudio del marco teórico que fundamenta el objeto de investigación.
- ✓ Selección de la herramienta de modelado.
- ✓ Selección del gestor de base de datos.
- ✓ Diseño del modelo de datos.
- ✓ Implementación de la base de datos.
- ✓ Validación de la solución propuesta.

Para la investigación se utilizaron los siguientes métodos de la investigativos.

Métodos Teóricos:

Histórico-Lógico: se utilizó para conocer todos los antecedentes que existen sobre el proceso de quejas, peticiones o denuncias dentro de la FGR, específicamente en el departamento del CLEP, mediante la documentación recogida durante el último año.

Analítico-Sintético: se utilizó para el procesamiento de toda la información relacionada con el tema de investigación, analizando los documentos que permitieron extraer los elementos más significativos relacionados con el objeto de estudio.

Modelación: posibilitó la creación de los diferentes diagramas y modelos que ayudaron a un mejor entendimiento de las funcionalidades que debe cumplir el sistema y al estudio de las relaciones entre las mismas.

Métodos Empíricos:

Experimental: posibilita la creación de las condiciones o adapta las existentes para realizar las pruebas o validación al objeto de estudio.

El presente trabajo de diploma está compuesto por 3 capítulos estructurados de la siguiente manera:

El Capítulo 1 estará basado en la fundamentación teórica estudiada para el trabajo, ubicando a los lectores en el uso importante de las bases de datos, abundando sobre sus características y funcionamiento. Desarrollando temas relacionado con las características del diseño de una base de datos y de los modelos de datos, analizando herramientas relacionadas con la administración, el modelado y la gestión de base de datos.

El Capítulo 2 se describe las características de la base de datos de los módulos Dictámenes, Proceso con Menores y Revisiones Laborales del Sistema de Informatización de la Gestión de las Fiscalías 2, así como la propuesta de solución obtenida. Los estándares de nomenclaturas utilizados, se muestran las características generales del diseño, mostrando los patrones de base de datos utilizados, presentando el modelo físico, la utilización de la normalización de las tablas y los índices. Describiendo en el mismo las pautas tomadas para el aseguramiento e integridad de los datos.

El Capítulo 3 abordará sobre los aspectos de seguridad implementados sobre la base de datos, así como las pruebas de validación realizada sobre la misma, dentro de las que se encuentran las pruebas de volumen, las cuales analizan el comportamiento de la base de datos al ser ingresadas

grandes cantidades de datos, las pruebas de rendimiento las cuales permitieron conocer los tiempos de respuesta a las consultas realizadas sobre la base de datos y las pruebas de estrés donde se simulan las consultas a la misma de manera concurrente.

Capítulo 1. Fundamentación teórica.

1.1 Introducción.

Este capítulo abordará los conceptos fundamentales utilizados para el desarrollo de la BD, ofreciendo un estudio amplio de las tecnologías y herramientas que se utilizan para realizar estas tareas y sus especificaciones, así como los sistemas gestores de bases de datos, el modelo de datos, las características del diseño de base de datos, las fases y los patrones utilizados para la misma. Se describe el uso de UML para el modelado de los datos. Abordando sobre la seguridad e integridad de los datos y los pasos generales para la normalización del modelo de datos.

1.2 Bases de Datos y Sistemas Gestores de Bases de Datos.

1.2.1 Bases de datos (BD).

En un inicio la información era procesada de forma manual, donde los procesos de búsqueda de datos eran engorrosos al tener que buscar en archivos físicos. El surgimiento y desarrollo de las bases de datos dio solución a este problema, pues la información era almacenada en equipos de cómputo donde el acceso a la misma era más cómodo, rápido y seguro. El acelerado avance en las tecnologías de la información y las comunicaciones, unido a la necesidad de obtener mejores rendimientos en los sistemas y elevar la disponibilidad de la información, propició la ampliación de soluciones que lograran solventar las necesidades del usuario en cuanto a estos términos. Es por esto que es necesario conceptualizar elementos claves en la investigación, para poder realizar un buen diseño de la bases de datos y garantizar factores que son de vital importancia en la emisión de los documentos de pasaportes y acreditaciones.

Una base de datos se define como un conjunto de datos que se encuentran organizados y que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto, con el fin de satisfacer tratamientos de información implicados en las actividades de una empresa (Grueso, 2009-2010).

Un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo (Mato García, 2005).

Las BD tienen implícitas las siguientes características (Elsmari R., 1993):

- ✓ Independencia de los datos. Es decir, que los datos no dependen del programa y por tanto cualquier aplicación puede hacer uso de los datos.

- ✓ Reducción de la redundancia. Llamamos redundancia a la existencia de duplicación de los datos, al reducir ésta al máximo consiguiendo un mayor aprovechamiento del espacio y además evita que existan inconsistencias entre los datos. Las inconsistencias se dan cuando nos encontramos con datos contradictorios.
- ✓ Seguridad. Un SBD debe permitir tener un control sobre la seguridad de los datos.
- ✓ Permite realizar un listado de la base de datos.
- ✓ Permiten la programación a usuarios avanzados.

Clasificación de las base de datos.

- ✓ Bases de datos operacionales

Algunos tipos de base de datos que califican dentro del nivel operacional son las que se describen a continuación:

Bases de datos para el procesamiento de transacciones en línea

Bases de datos de procesamiento de transacciones diarias en tiempo real y de alta disponibilidad. Son la opción correcta para sistemas de gestión de datos y procesos de negocio. Están centradas en la ejecución eficiente de transacciones y actualizaciones, con tolerancia a fallos. La integridad de los datos es un factor clave.

- ✓ Bases de datos de propósito general

Se califica como de propósito general una base de datos que no tenga un marcado procesamiento transaccional ni un marcado procesamiento analítico. Como su nombre lo indica, tiene como propósito general el almacenamiento de información para su consulta por terceras aplicaciones. Ejemplos de bases de datos de propósito general son:

- ➢ Bases de datos documentales.
- ➢ Bases de datos jurídicas.
- ➢ Bases de datos de información o contenidos.
- ➢ Bases de datos de configuración y administración de sistemas.

- ✓ Base de datos embebida

Una base de datos embebida es una base de datos estrechamente integrada con una aplicación de software que requiere acceso a datos almacenados. Esta base de datos requiere poca o ninguna administración ya que está concebida solamente para su operación por la aplicación de software con la cual interactúa. Es actualmente una amplia categoría tecnológica que incluye sistemas de

base de datos con distintas interfaces de desarrollo, diferentes estilos arquitectónicos, modos de almacenamiento, modelos de base de datos y diferentes destinos en el mercado. Este tipo de base de datos es muy común en dispositivos modernos como teléfonos inteligentes de última generación.

Se debe considerar utilizar una base de datos embebida si es necesario:

- Almacenar los datos internamente en la aplicación o dispositivo.
- Ejecutarse desatendidamente.
- Instalar y configurar la base de datos sin intervención del usuario.
- Definir soluciones completas de aplicación y datos.

✓ Base de datos NoSQL

Conocidas como No Solo SQL (Not Only SQL) o base de datos relacional no basada en SQL, representan una reciente evolución de arquitecturas de aplicaciones empresariales.

Las bases de datos NoSQL son frecuentemente usadas para adquirir y almacenar Big Data (Oracle Corporation, 2012). Estas son ideales para estructuras de datos dinámicas y son altamente escalables. Los datos almacenados en una base de datos NoSQL son típicamente de una alta variedad ya que los sistemas de referencia solo pretenden capturar estos datos sin categorizarlos ni parsearlos. Son frecuentemente utilizadas para recolectar y almacenar datos de medios sociales.

Las bases de datos NoSQL son las bases de datos On Line Transactional Process (OLTP) del mundo del Big Data y han emergido en compañías tales como Amazon, Google, LinkedIn y Twitter en el esfuerzo de operar datos improcedentes y volúmenes de operaciones bajo ajustadas restricciones de latencia (Oracle Corporation, 2012). Están optimizadas para la rápida captura de datos no estructurados y su rápida consulta de patrón simple. Son capaces de proveer un rendimiento muy rápido ya que el dato que es capturado es rápidamente almacenado con un identificador simple en vez de ser interpretado y casteados dentro de un esquema, posibilitando almacenar un gran número de transacciones, por lo que los datos son altamente variados. Para explotar al máximo las potencialidades que brindan las soluciones NoSQL a nivel empresarial, estas se deben combinar con soluciones SQL dentro de una misma infraestructura, que corresponda con los requerimientos actuales de administración y seguridad dentro de las empresas e instituciones.

Fig. 1 Diferentes tipos de bases de datos operacionales.

- ✓ Bases de datos para la toma de decisiones

Los principales tipos de base de datos que califican dentro del nivel de toma de decisiones son las que se describen a continuación.

Almacén de datos (data warehouse - DWH)

La universalmente aceptada definición de un almacén de datos desarrollada por Bill Inmon en la década de los 1980 es que es una colección de datos orientada a temas, integrada, variante en el tiempo y no volátil, utilizada en la toma estratégica de decisiones (Claudia Imhoff, 2003). En él se integran todos los datos de la organización así como los datos históricos. Se alimenta del ODS y de la capa de Integración y Transformación. Es la fuente de datos de los Mercados de Datos (Inmon, 2005). El Almacén de Datos se construye a partir de los datos del ambiente operacional.

Mercado de datos (data mart - DM)

Un Mercado de Datos es un subconjunto personalizado de datos tomados del Almacén de Datos y es donde se realiza la mayor actividad de procesamiento analítico en un ambiente de Inteligencia de Negocio (Claudia Imhoff, 2003). Son creados a la medida de las necesidades particulares de procesamiento de datos para la toma de decisiones de determinado departamento. Contienen datos detallados y sumariados, así como una limitada cantidad de datos históricos. Es significantemente de menor tamaño que el almacén de datos y cada departamento crea su propio Mercado de Datos.

Almacén de datos operacional (operational data store - ODS)

Un Almacén de Datos Operacional es una estructura híbrida que tiene algunos aspectos de un Almacén de Datos y otros aspectos de un sistema operacional. El ODS contiene datos integrados y soporta procesamiento OLAP de sistemas para la toma de decisiones, así como también procesamiento de transacciones de alto rendimiento OLTP (Inmon, 2005). Por sus características, el ODS se encuentra ubicado entre los niveles Operacional y Almacén de Datos. El ODS es el resultado de la combinación de diseño relacional y multidimensional.

Fig. 2 Bases de datos para la toma de decisiones.

Es importante identificar qué tipo de base de datos se debe desarrollar según las características de los datos que debe almacenar y procesar, ya que cada una tiene especificaciones estructurales muy particulares que, de violar el paradigma, se incurriría en problemas futuros de diseño y arquitectura (Osorio, 2012).

Coincidiendo en que las bases de datos son estructuras integradas, persistentes y variables en el tiempo, donde se puede almacenar datos sobre cualquier aspecto del mundo real. Para su manejo e interrelación, se utilizan los Sistemas Gestores de Base de Datos (SGBD).

1.2.2 Sistemas gestores de bases de datos.

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 una capa intermedia 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. Deben presentar los siguientes servicios (Zambrano Ramírez, 2008):

- ✓ Definir y crear bases de datos.
- ✓ Manipular los datos utilizando consultas.
- ✓ Brindar acceso controlado a los datos mediante mecanismos de seguridad de acceso a los usuarios.
- ✓ Mantener la integridad de los datos.
- ✓ Controlar la concurrencia a las bases de datos.
- ✓ Poseer mecanismos de copias de respaldo y recuperación para restablecer la información en caso de fallos en el sistema.

El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado. Así se puede utilizar y actualizar los datos almacenados en una o varias base de datos por uno o varios usuarios desde diferentes puntos de vista. Además almacenar y acceder a los datos de forma rápida y estructurada. Además sirven de interfaz entre la BD, el usuario y las aplicaciones que la utilizan (Mato García, 2005).

Los SGBD son una capa intermedia entre los programas que el usuario utiliza y el sistema, por tanto son los encargados de establecer la comunicación entre ellos,

Entre los gestores más utilizados en el mundo se encuentran Oracle, MySQL, PostgreSQL, Microsoft SQL Server; y existen otros gestores menos reconocidos como: SQLite, Microsoft Access, DBase, AdvantageDatabase, Fox Pro, Paradox, Open Access, NexusDB, Sybase IQ, WindowBase, IBM Informix.

1.2.3 PostgreSQL 9.1

PostgreSQL es un sistema gestor de base de datos objeto-relacional, bajo licencia Berkeley Software Distribution (BSD). Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Se ejecuta en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc. Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios. Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, etc. Soporte de todas las características de una base de datos profesional (triggers, store procedures, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.). Altamente adaptable a las necesidades del cliente. (Postgresql Cuba, 2012).

1.2.4 Estrategia de indexado.

Las operaciones realizadas sobre una base datos, a la hora de su manejo pueden implicar gran cantidad de tiempo, que puede medirse como demasiado costoso en la operación y poco aceptable. De ahí que muchos gestores tengan creado una estrategia para eliminar estas deficiencias.

Una muy utilizada son los índices, punteros que al igual que en los libros indican exactamente donde se encuentra un dato, eliminando así la búsqueda manual entre todo lo escrito para encontrar lo que interesa. En los gestores los índices son una estructura física que permite un tipo de acceso alternativo al secuencial, es creado a partir de una o varias columnas de una tabla (Scridb, 2012). Al insertar índices en las tablas se deben tener en cuenta los siguientes factores:

- ✓ Se deben indexar los campos que son más utilizados en las búsquedas (los que aparecen en las cláusulas WHERE o JOIN) para mejorar una consulta con el operador (SELECT).
- ✓ Los índices se deben crear sobre campos con valores únicos pues funcionan de una mejor manera si el campo no tiene valores duplicados.

- ✓ Se deben indexar campos con valores de la menor longitud posible, preferiblemente enteros y si se indexa un campo de texto, se debe evitar hacerlo sobre campos de longitud variable.
- ✓ Por último se recomienda evitar crear índices innecesariamente pues estos se actualizan con cada cambio en la tabla asociada y pueden ralentizar las modificaciones de la misma.

El gestor utilizado PostgreSQL utiliza varios tipos de índices (Postgresql Cuba, 2012). Están los Hash; que son una tabla hash o mapa hash es una estructura de datos que asocia llaves o claves con valores, la operación principal que soporta de manera eficiente es la búsqueda. También están los GIST, Generalized Search Tree (Árbol de Búsqueda Generalizada), que no son un solo tipo de índice, sino más bien una infraestructura dentro de la cual muchas estrategias diferentes de indexación se pueden aplicar. Otro de los tipos son los GIN, Generalized Inverted Index (Índice Invertido Generalizado), que no son más que índices invertidos que pueden controlar los valores que contienen más de una clave, por ejemplo las matrices; al igual que con GIST, GIN puede soportar muchas estrategias de indexación diferentes definidas por el usuario.

La opción estándar de índices del SGBD PostgreSQL es la estrategia “B-tree”, Árbol-B (árbol balanceado) aplicada en la solución de la base datos de los módulos Dictámenes, Proceso con Menores y Revisiones Laborales, es utilizada para encontrar un único valor o para explorar en un área de distribución la búsqueda de los valores claves mediante el empleo de los operadores: <, <=, =, >, =>. En este tipo de estrategia la búsqueda de datos es utilizando las llaves primarias y foráneas, poseen índices de este tipo, lo que implica que cualquier búsqueda que se realice utilizando las llaves se optimizará mediante este método. (Postgresql Cuba, 2012)

1.3 Modelo de datos.

En la informática un modelo es utilizado para representar por medios de abstracciones la realidad, enfocando ciertas partes de un sistema, las de interés, y restándole importancia a otras, de esta forma se aleja al usuario de funciones complejas que realiza por detrás la computadora para obtener algún resultado sobre cierta acción (ICT, 2012). Basado en este concepto, los modelos de datos se convierten un elemento esencial para estas operaciones. Varios autores han afirmado que:

El modelo de datos, como abstracción del universo de discurso, es el enfoque utilizado para la representación de las entidades y sus características dentro de la BD. (Batini, 2004)

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una BD: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. En general, un modelo no es capaz de expresar todas las propiedades de una realidad determinada, por lo que hay que añadir afirmaciones que complementen el esquema. Los modelos de datos

contienen también un conjunto de operaciones básicas para realizar lecturas y actualizaciones de los datos. Además, los modelos de datos más modernos incluyen conceptos para especificar comportamiento, permitiendo especificar un conjunto de operaciones definidas por el usuario. (Marqués, 2009)

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 base de datos.

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 Base de Datos 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 Base de Datos 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).

En el siguiente epígrafe se analizarán el **modelo relacional** y el **modelo entidad-relación.**

1.3.1 Modelo relacional.

El modelo relacional de base de datos de gestión es un modelo de base de datos basado en la lógica de predicados de primer orden , primero formulado y propuesto en 1969 por Edgar F. Codd . En el modelo relacional de una base de datos, todos los datos se representan en términos de tuplas o filas, agrupados en las relaciones. Una base de datos organizada en términos del modelo relacional es una base de datos relacional.

- ✓ El desarrollo de un lenguaje SQL, convertido en el lenguaje estándar de los sistemas relacionales.
- ✓ La producción de varios SGBD relacionales durante los años ochenta, como IBM DB2 y ORACLE de ORACLE Corporation.

El modelo relacional de datos supuso un gran avance con respecto a los modelos anteriores. Este modelo está basado en el concepto de relación. Una relación es un conjunto de n-tuplas. Una tupla, al contrario que un segmento, puede representar tanto entidades como interrelaciones. Los lenguajes matemáticos sobre los que se asienta el modelo relacional, el álgebra y el cálculo relacionales, aportan un sistema de acceso y consultas orientado al conjunto. La repercusión del modelo en los SGBD comerciales actuales ha sido enorme, estando hoy en día la gran mayoría de los gestores de bases de datos basados en mayor o menor medida en el modelo relacional. Este permite al diseñador de base de datos para crear una representación coherente y lógico de la información . La consistencia se consigue incluyendo declaradas limitaciones en el diseño de bases de datos, que generalmente se hace referencia como el esquema lógico. La teoría incluye un proceso de normalización de base de datos que puede obtenerse un diseño con ciertas propiedades deseables seleccionado de un conjunto de lógicamente equivalentes alternativas. Los planes de acceso y otras aplicaciones y detalles de operación son manejadas por el SGBD del motor, y no se reflejan en el modelo lógico. Esto contrasta con la práctica común de bases de datos SQL en el que el ajuste de rendimiento a menudo requiere cambios en el modelo lógico. El

principio básico del modelo relacional es el principio de información : toda la información está representada por valores de datos en las relaciones.

1.3.2 Modelo entidad–relación.

Con el objetivo de erradicar las deficiencias presentadas en el modelo relacional para representar los datos de una manera más real, el Dr. Peter Chen presentó en 1976 el modelo entidad-relación, convirtiéndose en la técnica más utilizada en el diseño de bases de datos (Orallo, 2002)

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).

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 cardinalidad 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).

1.4 Diseño de bases de datos a través de Dbplanning Framework.

El diseño de una BD es un proceso complejo que abarca decisiones a muy distintos niveles. Existen varios factores importantes que influyen en el diseño de BD:

- ✓ Mantenerse comunicado con los usuarios
- ✓ Seguir una metodología de diseño.

- ✓ Emplear una técnica centrada en datos.
- ✓ Poner consideraciones de integridad dentro de los modelos.
- ✓ Utilizar diagramas para representar los modelos.
- ✓ Utilizar un lenguaje de diseño de BD.

La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas independientemente en fases.

1.4.1 Fases del diseño de base de datos.

El diseño de una base de datos no es un proceso sencillo. Habitualmente, la complejidad de la información y la cantidad de requisitos de los sistemas de información hacen que sea complicado. Por este motivo, cuando se diseñan bases de datos es interesante aplicar la vieja estrategia de divide y vencerás. Por lo tanto, conviene descomponer el proceso del diseño en varias etapas: Diseño conceptual, Diseño lógico y Diseño físico. (Dataprix, 2013).

Diseño conceptual:

Como resultado de esta fase se obtiene el esquema conceptual de la BD, donde quedan descritas las especificaciones de requisitos de usuario. Se representan los elementos que pertenecen a una BD, que reciben el nombre de entidades, las cuales se corresponden con el concepto de clase de la Programación Orientada a Objeto (POO) y donde cada tupla de una futura relación representaría un objeto de la POO.

Diseño lógico:

En esta parte el resultado del diseño conceptual se transforma de tal forma que se adapte a la tecnología que se debe emplear. Más concretamente, es preciso que se ajuste al modelo del SGBD con el que se desea implementar la BD. Por ejemplo, si se trata de un SGBD relacional, esta etapa obtendrá un conjunto de relaciones con sus atributos, claves primarias y claves foráneas. Esta etapa parte del hecho de que ya se ha resuelto la problemática de la estructuración de la información en un ámbito conceptual, y permite concentrarnos en las cuestiones tecnológicas relacionadas con el modelo de BD.

Diseño físico:

En esta fase se transforma la estructura obtenida en la etapa del diseño lógico, con el objetivo de conseguir una mayor eficiencia; además, se completa con aspectos de implementación física que dependerán del SGBD utilizado. Por ejemplo, si se trata de una BD relacional, la transformación de la estructura puede consistir en lo siguiente: tener almacenada alguna relación que sea la

combinación de varias relaciones que se han obtenido en la etapa del diseño lógico, partir una relación en varias, añadir algún atributo calculable a una relación. Los aspectos de implementación física que hay que completar consisten normalmente en la elección de estructuras físicas de implementación de las relaciones, la selección del tamaño de las memorias intermedias (buffers) o de las páginas.

1.4.2 Patrones de diseño de bases de datos.

Al partir de la definición de un patrón, que no es más que un fragmento de un modelo que es recurrente, lo cual deriva como una solución a un problema específico que se ha mantenido a pesar del tiempo proporciona una solución probada a un problema común, documentada en un formato coherente (Erl, 2009).Por lo cual en el diseño y construcción de una BD se requiere de mayor esfuerzo y análisis posible, ya que a partir de este se crean las bases de datos que en la actualidad suelen ser muy grandes y utilizar patrones de diseño en su modelado asegura un mejor resultado. (Blaha, 2010)

Patrón: árbol fuertemente codificado. (Hardcoded tree).

Los árboles son sumamente utilizados en los diseños actuales, por lo que existen varios patrones de diseño de BD asociados a los mismos: En este tipo de patrón las entidades son asociadas a un nivel del árbol. Normalmente constituyen relaciones de 1 a muchos (n). Normalmente utilizado para representar jerarquías donde es bien conocida la estructura y es importante representar la correspondencia, por ejemplo las estructuras organizacionales. Es importante señalar que este patrón debe utilizarse sólo en los casos en que los cambios en la estructura a representar sean poco probables. Así como aclarar que el patrón admite tantos niveles como requiera la jerarquía que se vaya a representar.

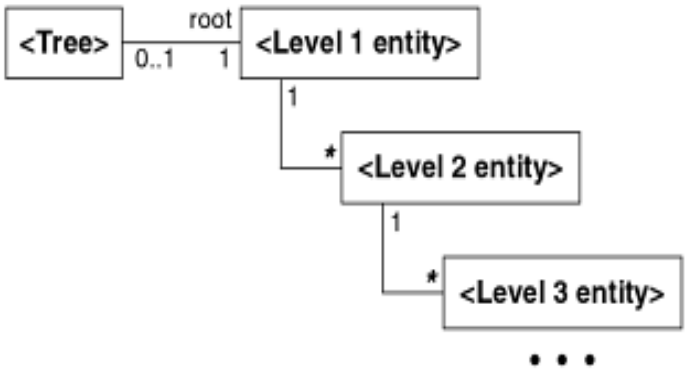

Fig. 3 Árbol fuertemente codificado

Patrón llave subrogada.

Este patrón es muy utilizado pues facilita la interacción con la BD en un futuro. El mismo plantea que se genere una llave primaria única para cada entidad, en vez de usar un atributo identificador en el contexto dado. Normalmente se usan enteros en columnas identity o GUID (Global Unique Identifier) que están demostradas que no se repiten o con una probabilidad extremadamente baja. Esto permite que las tablas sean más fáciles de consultar a partir del identificador, pues todos tienen el mismo tipo en cada una de las tablas.

1.4.3 Diseño de base de datos a través de Dbplanning Framework.

Tomando las características del Framework Dbplanning, el cual propone una guía de desarrollo e implementación de base de datos. Agrupando las fases generales de diseño de una base de datos, además posibilita la documentación de sus actividades.

El dbplanning (databaseplanning) es un framework de desarrollo de base de datos para proyectos de desarrollo de software, el cual posibilita al equipo de desarrollo establecer una línea base para el desarrollo de bases de datos. No plantea ningún elemento innovador de forma individual, sino un seguimiento del desarrollo de base de datos dentro del ciclo de desarrollo de software con el objetivo de ganar en productividad. (Osorio, 2012)

La utilización de este marco de trabajo está fundamentada en cubrir las lagunas de diseño de base de datos que no describen específicamente metodologías, tanto ágiles como tradicionales, a la hora de concebir el desarrollo de software. De ello cuenta que la metodología Rational Unified Process (Proceso Unificado de Rational) aborda el desarrollo de base de datos como una tarea de diseño, careciendo de especificaciones necesarias en todo el esfuerzo de desarrollar y mantener una base de datos; por otro lado la Agile Data Method (Método de Datos Ágil)define buenos enfoques prácticos desde el punto de vista ágil, pero el seguimiento sobre cómo se evoluciona no está definido; y la metodología XP trata el tema dejando esta tarea en manos de los programadores, lo cual comúnmente propicia malas decisiones en la actividad.

1.4.4 Características generales de Dbplanning Framenwork.

El framework transita por las fases de Inicio, Desarrollo y Despliegue. Plantea 4 actividades: Modelado de Datos, Configuraciones, Implementación y ADTP (Acceso a Datos, Tuning (optimización por su traducción del inglés) y Prueba), las cuales se desarrollan durante las 3 fases definidas. Las actividades establecen relaciones las cuales deben de ser cumplidas.

En la fase de inicio es definida la arquitectura del sistema a partir de los requisitos funcionales y no funcionales; en la fase de desarrollo se es desarrollada la solución de software y la tercera fase la

de despliegue, es la puesta en marcha y es el soporte del sistema, incluyendo en esta la etapa de pruebas de liberación.

En las actividades se responde a: estructuración del modelo de datos (Modelado de Datos); configuraciones de la Arquitectura de Datos (Configuraciones); estructuración de todas las funcionalidades de base de datos (Implementación); creación y prueba de la interfaz de comunicación entre la base de datos y los usuarios (ADTP).

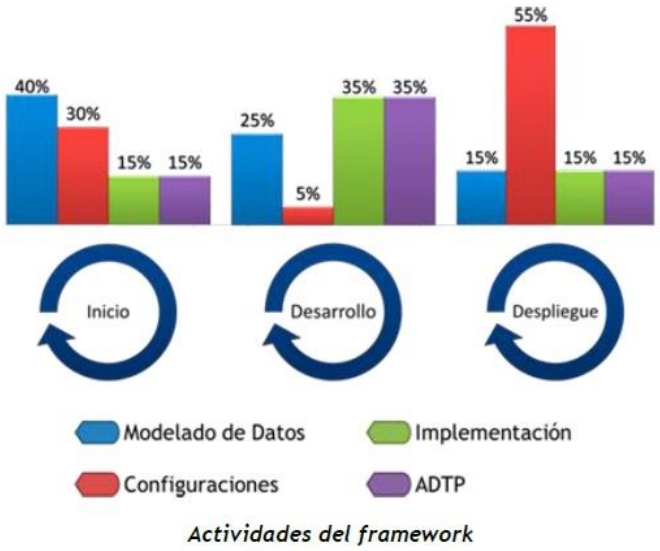

Fig. 4 Relación de Fases y Actividades de dbplanning framework.

1.5 Normalización de bases de datos.

Obtener un diseño de BD eficiente evitaría los problemas de actualización de la información, por ello la teoría de normalización de BD se encarga de obtener modelos eficaces que eviten anomalías en la BD.

La normalización es la expresión formal del modo de realizar un buen diseño. Provee los medios necesarios para describir la estructura lógica de los datos en un sistema de información. (Mato García, 2005). Las ventajas de aplicar la misma son:

- ✓ Eliminar la redundancia o repetición de los datos en el sistema.
- ✓ Descarta las inconsistencias de actualización de datos como resultado de las actualizaciones parciales y la redundancia de la información.
- ✓ Mejora la independencia de los datos, permitiendo realizar las extensiones de la BD, las cuales afectan muy poco o nada a los programas que acceden a los datos.
- ✓ Evita las anomalías de borrado o pérdidas no intencionadas de datos.

- ✓ Elimina las anomalías de inserción o imposibilidad de adicionar datos en la BD debido a la ausencia de otros datos.

La normalización se divide en varias fases, llamadas formas normales (FN), las cuales suponen el haber cumplido la anterior en su totalidad para poder alcanzar la próxima (Mato García, 2005). Estas son:

Primera Forma Normal (1FN):

Una relación se encuentra en primera forma normal cuando no hay grupos repetitivos en sus atributos. Todos los dominios de los atributos contienen únicamente valores atómicos. Es una restricción inherente al modelo relacional, por lo que su cumplimiento es obligatorio. Se eliminan los atributos multivariados. Esta forma normal elimina los valores repetidos dentro de una BD.

Segunda Forma Normal (2FN):

Una relación está en 2FN si además de estar en 1FN todos los atributos que no forman parte de ninguna clave candidata tienen dependencia funcional completa respecto de cada una de las claves. Toda relación cuya clave está formada por un solo atributo está en 2FN. Se eliminan las dependencias funcionales no totales. Siempre es posible transformar un esquema de relación que no esté en 2FN en esquemas de relación 2FN, sin pérdida de información o de dependencias.

Tercera Forma Normal (3FN):

Una relación está en 3FN si además de estar en 2FN, los atributos que no forman parte de ninguna clave candidata facilitan información solo acerca de las claves y no acerca de otros atributos. Cada atributo no clave es dependiente no transitivamente de la clave primaria. Se eliminan las dependencias funcionales transitivas. Siempre es posible transformar un esquema de relación que no esté en 3FN en esquemas de relación 3FN, sin pérdida de información o de dependencias.

Forma Normal Boyce Codd (FNBC):

Una relación está en FNBC si lo está en 3FN y si además el conocimiento de las claves permite averiguar todas las relaciones existentes entre los datos de la relación. Las claves candidatas deben ser los únicos descriptores sobre los que se facilita información por cualquier otro atributo. Cada determinante (atributo con el cual otro atributo tiene dependencia funcional total) debe ser una clave candidata. Se eliminan claves candidatas compuestas que se solapan. No siempre es posible transformar un esquema de relación en FNBC sin que se produzca pérdida de dependencias funcionales. Si se puede hacer sin pérdida de información.

Cuarta Forma Normal (4FN):

La cuarta forma normal (4NF) es una forma normal usada en la normalización de bases de datos. La 4NF se asegura de que las dependencias multivaluadas independientes estén correcta y eficientemente representadas en un diseño de base de datos. La 4NF es el siguiente nivel de normalización después de la forma normal de Boyce-Codd (Pérez, 2011).

Quinta Forma Normal (5FN)

La quinta forma normal (5FN), también conocida como forma normal de proyección-unión (PJ/NF), es un nivel de normalización de bases de datos designado para reducir redundancia en las bases de datos relacionales que guardan hechos multivalores, aislando semánticamente relaciones múltiples relacionadas. Una tabla se dice que está en 5NF si y sólo si está en 4NF y cada dependencia de unión (join) en ella es implicada por las claves candidatas. (Pérez, 2011)

Al realizar el proceso de normalización hasta la 3FN se puede asegurar, en gran medida, la no ocurrencia de los problemas de redundancia en la actualización. La BD propuesta como solución al problema que resuelve este trabajo está normalizada hasta esta fase, lo cual advierte sobre algunos aspectos a considerar a la hora de realizar el proceso de normalización, para un trabajo más correcto y efectivo.

Desnormalización:

Es el proceso de procurar optimizar el desempeño de una base de datos por medio de agregar datos redundantes. A veces es necesaria porque las actuales DBMS implementan el modelo relacional pobremente. Una verdadera DBMS relacional debe permitir una base de datos completamente normalizada a nivel lógico, mientras proporciona el almacenamiento físico de los datos afinado para alto rendimiento. La desnormalización no es una opción muy elegante pero esta puede reducir enormemente el esfuerzo y el tiempo de respuesta en términos de consultas a las bases de datos. (Marcelo D. Vinjoy, 2010).

Un modelo de datos desnormalizado no es lo mismo que un modelo de datos que no ha sido normalizado, y la desnormalización debe tomar lugar solamente después de que haya ocurrido un nivel satisfactorio de normalización y de que hayan sido creadas las restricciones y/o reglas requeridas para ocuparse de las anomalías inherentes en el diseño. Por ejemplo, que todas las relaciones estén en la tercera forma normal y cualquier relación con dependencias de unión (join) y multivalor sea manejado apropiadamente.

1.6 Integridad de los datos.

Cuando los contenidos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes. Pueden añadirse datos no válidos a la BD, tales como un pedido que especifica un producto inexistente. Los cambios en la BD pueden perderse debido a un error del sistema o a un fallo en el suministro de energía. Una de las funciones importantes de un sistema gestor de base de datos relacional es preservar la integridad de sus datos almacenados en la mayor medida posible.

Las restricciones que concretan los estados de consistencia de los datos definen la integridad de los datos en la BD, por lo cual se tendría una corrección y completitud de la información. La integridad de los datos consiste en garantizar la no contradicción entre los datos almacenados de modo que en cualquier momento los datos almacenados sean correctos, es decir, que no se detecte inconsistencia entre los datos. Está relacionada con la minimización de la redundancia, ya que es más fácil garantizar la integridad si se elimina la redundancia. (Mato García, 2005)

Existen cuatro restricciones fundamentales que sustentan la integridad de los datos, las mismas se mencionan a continuación: (UCV, 2010).

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

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

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

1.7 Seguridad en las bases de datos.

Las bases de datos hoy en día son de gran importancia en cada una de las aplicaciones que la requieran, es por esto que la seguridad para su acceso y manejo de la información se restringe en una jerarquía de usuarios. Los SGBD permiten mantener la seguridad mediante el establecimiento de claves para identificar al personal autorizado a utilizar la base de datos. Las autorizaciones se pueden realizar a nivel de operaciones, de modo que un usuario puede estar autorizado a consultar

ciertos datos pero no a actualizarlos. En el caso específico el gestor de bases de datos PostgreSQL 9.1 presenta una serie de características que facilita el trabajo para tener un alto nivel de seguridad.

La seguridad de la base de datos está implementada en varios niveles:

- ✓ Protección de los ficheros de la base de datos, ya que todos los ficheros almacenados en la base de datos están protegidos contra escritura por cualquier cuenta que no sea la del súper usuario de Postgres.
- ✓ Las conexiones de los clientes se pueden restringir por dirección IP y/o por nombre de usuario mediante el fichero *pg_hba.conf* situado en PG_DATA.
- ✓ Las conexiones de los clientes pueden ser autentificadas mediante otros paquetes externos.
- ✓ A cada usuario de Postgres se le asigna un nombre de usuario y (opcionalmente) una contraseña. Por defecto, los usuarios no tienen permiso de escritura a bases de datos que no hayan creado.
- ✓ Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse en base a esos grupos

1.8 Herramientas y entorno de desarrollo.

Las prestaciones de las herramientas marcan en el desarrollo de un software rapidez, seguridad y estabilidad en los resultados. En el mundo actual las herramientas se miran desde varios puntos de vistas: las privativas, que presentan el gran problema de pago de licencias altamente costosas; las libres, que son de libre uso y sin ningún tipo de costo. A estos aspectos se le suma, que su uso sea multiplataforma, capacidad del software de ejecutarse en varios sistemas operativos, y puedan ser de código abierto, lo cual sería la posibilidad de tener acceso al código fuente del software y de este utilizar alguna parte o modificarla para uso propio.

La arquitectura base del proyecto SIGEF II para el desarrollo de la aplicación tiene definido un conjunto de software específicos, los cuales cumpliendo con las políticas de desarrollo del centro CEGEL, son software libre y multiplataforma.

1.8.1 UML 2.0.

Los lenguajes de modelado son utilizados para facilitar el diseño de un software, tanto a la hora de desarrollar, dar mantenimiento o integrar con otro software, todo ello es posible gracias a que son un conjunto estandarizado de símbolos y sus distintas combinaciones.

El lenguaje de modelado unificado (por sus siglas en inglés *Unified Modeling Language*), permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software. Proporciona una forma estándar de representar los planos de un sistema, y comprende tanto elementos conceptuales, como los procesos de negocio y las funciones del sistema. Permite una comunicación sencilla y rápida entre desarrolladores y clientes del software que se desarrolla y simplifica el proceso complejo de análisis y diseño de software. (Pressman, 2005). El diagrama que mejor que describe las relaciones entre clases y la noción de atributos claves que relacionan entre sí las tablas unas con otras, es la extensión de UML, Diagrama de Relación de Entidad (ER diagram), el cual sirve como fiel elemento para seleccionar este lenguaje.

1.8.2 VISUAL PARADIGM 8.0.

Herramienta CASE que permite generación de código y la base de datos a partir de los diagramas UML realizados. Principalmente es utilizada en la modelación de negocio y en el diseño es de gran ayuda a los analistas porque pueden visualizar el flujo central y detallado de cada proceso mediante diagramas, además posibilita crear un conjunto amplio de artefactos utilizados con mucha frecuencia durante las disciplinas análisis, diseño, implementación o el despliegue. Esta herramienta permite a múltiples usuarios trabajar sobre el mismo proyecto. Visual Peradigm apoya plenamente la última versión del estándar BPMN (Business Process Model and Notation) y la característica Animación, posibilita a los usuarios ver el flujo de trabajo en acción, dando así una perspectiva más amplia del funcionamiento del negocio (Visual Paradigm, 2013)

1.8.3 RAPIDSVN 0.12.1.

RapidSVN es un cliente gráfico de Subversión multiplataforma. Que se distribuye bajo la Licencia Pública General de GNU (Linux Six Blog, 2012).

Características

- ✓ **Simple** - proporciona una interfaz fácil de usar para las características de Subversión
- ✓ **Eficiente** - simple para los principiantes pero lo suficientemente flexible como para aumentar la productividad para los usuarios de Subversion con experiencia
- ✓ **Portable** - se ejecuta en cualquier plataforma en la que Subversion y wxWidgetspuede ejecutar: Linux, Windows, Mac OS / X, Solaris, etc.
- ✓ **Rápido** - completamente escrito en C + +

- ✓ **Multilingüe** - ha sido traducido a muchos idiomas: alemán, francés, italiano, portugués, ruso, ucraniano, chino simplificado, japonés
- ✓ **Soporte** completo para Unicode

1.8.4 ORM Doctrine 2.3.2

Es un marco de trabajo de mapeo de objeto relacional (ORM) para *PHP* 5.3.0 (Personal Home Page) o versiones superiores, el cual proporciona persistencia transparente de objetos PHP, el cual se sitúa en la parte superior de una poderosa capa de abstracción de base de datos (DBAL por sus siglas DataBase Abstraction Layer). La principal tarea de los ORM para PHP es la traducción transparente entre objetos (PHP) y las filas relacionales de la base de datos. (Ing. Blanco, 2012)

1.8.5 IDE NetBeans 7.2

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso (Netbeans, 2012).

Entre las novedades principales de esta versión destaca mejoras en el soporte para la nueva sintaxis de Java 7, soporte de HTML5, mejoras al editor Java, mejoras en la integración con WebLogic y soporte para PHP 5.3.

1.8.6 Subversion 1.5.

Para el control de versiones según las necesidades del proyecto se usa Subversion, es un software libre bajo una licencia de tipo Apache/BSD y se le conoce también como SVN por ser el nombre de la herramienta utilizada en la línea de comando.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer porque la calidad del mismo vaya a verse afectada, si se ha hecho un cambio incorrecto a los datos, simplemente deshace ese cambio (Postgresql Cuba, 2012).

Entre otras ventajas por las que se selecciona esta herramienta están:

- ✓ Se sigue la historia de los archivos y directorios a través de copias y renombrados.

- ✓ Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- ✓ Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL (Strutured Query Language), LDAP (Lightweight Directory Access Protocol), PAM (Pluggable Authentication Modules)).

1.8.7 PGAdmin III 1.14.0.

Es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres. (Ubuntu, 2008)

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas Unix), y puede encriptarse mediante SSL para mayor seguridad. (Ubuntu, 2008)

1.9 Herramientas para las pruebas.

1.9.1 EMS Data Generator 2005 for PostgreSQL 2.21.

Es una potente herramienta para la generación de datos de prueba a tablas de bases de datos en PostgreSQL. Permite definir tablas y campos para la generación de datos, establecer rangos de valores, generar campos char por la máscara, obtener listas de valores de las consultas SQL y muchas otras características para generar datos de prueba de forma sencilla y de manera directa. También proporciona aplicación de consola, lo que le permite generar datos en un solo toque mediante el uso de plantillas de generación. Las principales características que posee son (PostgreSQL, 2005):

- ✓ Fácil de usar interfaz de asistente
- ✓ Seis idiomas disponibles: Inglés, francés, alemán, italiano, ruso y español.
- ✓ Generación de datos a varias tablas de bases de datos diferentes en un host.
- ✓ Todos los tipos de datos PostgreSQL, incluyendo Array, direcciones de red y tipos geométricos.

- ✓ Los diferentes tipos de generación para cada campo, incluyendo lista, la generación incremental de datos al azar.
- ✓ Capacidad para utilizar los resultados de consultas SQL como lista de valores para la generación de datos.
- ✓ El control automático sobre la integridad referencial para las tablas vinculadas generación de datos.
- ✓ Gran variedad de parámetros de generación para cada tipo de campo.
- ✓ Capacidad para establecer los valores NULL para cierto porcentaje de los casos.
- ✓ Posibilidad de guardar todos los parámetros de generación, creado en la sesión del asistente actual.
- ✓ Utilidad de línea de comandos para generar datos utilizando el archivo de plantilla.

1.9.2 Apache JMeter 2.9.

Es un software de código abierto realizado en Java, puede ser utilizado para probar el rendimiento tanto en recursos estáticos y dinámicos (archivos, Servlets, scripts de Perl, Java Objects, bases de datos y consultas, servidores FTP y mucho más). Se puede utilizar para simular una carga pesada en el servidor, de red o un objeto para probar su resistencia o para analizar el rendimiento general bajo diferentes tipos de carga. Se puede utilizar para hacer un análisis gráfico de rendimiento o para probar su servidor / script / comportamiento del objeto bajo carga pesada concurrentes. Sus principales características son (Jmeter, 2013):

- ✓ Se puede cargar y probar diferentes tipos de servidores de rendimiento.
- ✓ Portabilidad completa y pureza Java 100%.
- ✓ Permite el muestreo simultáneo de muchas discusiones y toma de muestras simultáneas de diferentes funciones de los grupos de hilos separados.
- ✓ Su cuidadoso diseño GUI permite un funcionamiento más rápido y los tiempos más precisos.
- ✓ Almacenamiento en caché y el análisis / Reproducción de resultados de las pruebas fuera de línea.
- ✓ Varias estadísticas de carga pueden ser elegidos con temporizadores que pueden ser conectados.
- ✓ Análisis de datos y plugins (extensiones) de visualización permiten una gran extensibilidad y personalización.

1.10 Conclusiones parciales

Realizado el estudio de los conceptos esenciales de BD, de las principales características, tocando los temas importantes como son los sistemas gestores de base de datos, la metodologías de diseño y las herramientas tanto para el modelado como para el mantenimiento de la BD se puede concluir que:

El framework dbplanning, ofrece características importantes para el desarrollo del sistema. Las herramientas definidas por el equipo de desarrollo son de alta calidad para desarrollar las tareas de este trabajo, cumpliendo estas con los requisitos de ser un software libre y multiplataforma. Así como un amplio estudio de los patrones a utilizar en el diseño de los modelos, los cuales ayudan a un mejor entendimiento de los mismos.

Capítulo 2. Análisis de la solución propuesta.

2.1 Introducción

Este capítulo aborda, especificaciones importantes del desarrollo de la base datos, como son las configuraciones de trabajo y las normas aplicadas al modelo para más claridad en el resultado final. A partir del resultado obtenido se realiza una descripción sobre los datos, las cuales incluyen patrones utilizados, las formas de acceso, la optimización realizada sobre estos y otras estrategias para mejor manejo de la base de datos en el SGBD.

2.2 Configuración del entorno de desarrollo.

Para desarrollar el modelo de datos de los módulos Dictámenes, Proceso con Menores y Revisiones Laborales de SIGEF II se utiliza la siguiente configuración.

Herramienta	Función
Subversion 1.5	Repositorio para el versionado del modelo
RapidSVN 0.12.1	Gestión de versiones del repositorio
Visual Paradigm 8.0	Diseño del modelo de datos
PGAdmin III 1.14.0	Cliente de base de datos
PostgreSQL 9.1	Gestor de base de datos
IDE NetBeans 7.2	Desarrollo del acceso a datos

Tabla 1 Relación de herramientas y funciones en el entorno de desarrollo.

Las herramientas de modelado y desarrollo, NetBeans y Visual Paradigm, incluyen opciones de conexión al repositorio de versiones, posibilitando el control cambios desde la misma herramienta. La herramienta RapidSVN se utilizó para el proceso de control de versiones.

2.2.1 Usuarios y privilegios.

El control y seguridad de la BD se realizó de dos formas, una mediante el patrón RBAC (Control de acceso basado en roles), que implementa el SGBD. El cual es una función de seguridad para controlar el acceso de usuarios a tareas que normalmente están restringidas al superusuario. Mediante la aplicación de atributos de seguridad a procesos y usuarios, RBAC puede dividir las capacidades de superusuario entre varios administradores. La gestión de derechos de procesos se implementa a través de privilegios. La gestión de derechos de usuarios se implementa a través de RBAC.(Oracle,

2013), y la otra forma es mediante el fichero pg_hba.conf el cual define como, donde y desde que sitio un usuario puede utilizar dependiendo del tipo de conexión y del tipo de autentificación (Linux.org).

Para el trabajo con la base de datos del proyecto SIGEF II a través de este patrón el servidor PostgreSQL se creó diferentes roles. El rol administrador tiene permisos para administrar la base de datos en su totalidad, y los usuarios con privilegios a la actualización, inserción y eliminación de datos.

2.3 Arquitectura de Datos

En la Fiscalía Nacional de la República como previo desarrollo de la aplicación SIGEF II, para la gestión de su información, contará con base de datos independientes en cada una de sus fiscalías provinciales y municipales, contando con un servidor central de BD en la Fiscalía General donde se almacenará toda la información relevante de cada una de las entidades fiscales del país. En cada una de estas entidades se realizará una backup completo de la BD mensualmente y se guardarán copias de las bases de datos anteriores, en dependencia de los recursos tecnológicos y las políticas de protección de la información de la organización. Estableciendo una política de reemplazo en consonancia con la cantidad de copias establecidas, las cuales van a ser 12 copias (Acosta, 2013).

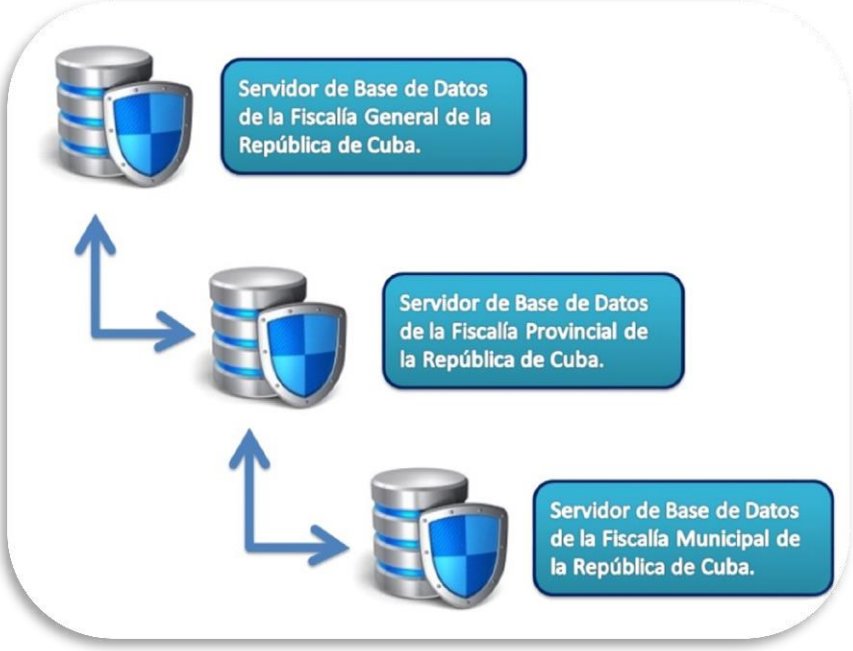

Fig. 5 Distribución de datos.

2.3.1. Réplica para la base de datos de la aplicación SIGEF II.

Se entiende como proceso de replicación como el proceso de compartir objetos y datos de una base de datos a múltiples bases de datos. La replicación de datos es utilizada para el mantenimiento automático de copias de datos en múltiples servidores, por ejemplo el almacenamiento de datos provenientes de servidores Web en la memoria caché. Es una técnica para la mejora de los servicios que ofrecen los sistemas distribuidos porque proporciona una mejora del rendimiento de los servicios e incrementa su disponibilidad y lo hace tolerante a fallos (Martínez, 2013).

En las soluciones de los procesos de réplica generalmente deben tenerse en cuenta los siguientes principios (Acosta, 2013):

- ✓ Es importante llegar a un equilibrio en la cantidad de veces que se replica y la cantidad de información a transferir en función de la satisfacción del cliente.
- ✓ La réplica debe verse como un proceso normal en el funcionamiento de un sistema de software. Con frecuencia esta no constituye el núcleo del funcionamiento de los subsistemas ni de la interacción con los clientes y por tanto no debe afectar significativamente el rendimiento de estos.
- ✓ La réplica como parte de la solución de software debe además satisfacer los requisitos de los clientes que de ella dependan.

Básicamente existen dos tipos o entornos de réplicas: el de solo lectura o maestro-esclavo (master-slave en inglés) que permite al nodo maestro realizar consultas de lectura/escritura, mientras que los nodos esclavos solo realizan consultas de lectura lectura; y el otro entorno es el multimaestro donde muchos nodos maestros interactúan entre sí facilitando la sincronización de los cambios.

De forma tal que las réplicas entre los servidores de bases de datos del sistema SIGEF II, se realizarán de forma multimaestro, en el cual se tendrán dos variantes. Una forma fundamentalmente orientada a la réplica de los datos entre el servidor central y las instancias inferiores y la segunda forma desde las instancias inferiores hacia las superiores según las configuraciones establecidas de que se debe subir y que no (Acosta, 2013).

2.4 Nomenclatura y normas para los modelos de datos.

El entendimiento de la base de datos para todos los involucrados en el desarrollo de una aplicación proporciona un mejor manejo y gestión de los datos, de ahí que se defina una nomenclatura para todas las entidades y sus atributos en el modelo.

2.4.1 Generalidades

- ✓ Todas las entidades y sus atributos estarán escritos con letra minúscula.
- ✓ En los nombres de los atributos y entidades los caracteres como tildes, puntos, la ñ no están permitidos.

2.4.2 Nomenclatura para entidades.

- ✓ Los nombres deben ser concisos y claros con el negocio no excederán los 35 caracteres, además de adaptarse a los nombres definidos en el diagrama de clases persistente.
- ✓ Los nomencladores estarán descritos inicialmente por una n y a continuación el nombre identificativo de la entidad especificada en el diagrama de clases persistente.
- ✓ Las tablas de datos variados tendrán de comienzo una d y a continuación el nombre identificativo de la entidad, especificada en el diagrama de clases persistente.
- ✓ Las relaciones de mucho a mucho donde se genera una nueva tabla, nombre estará compuesto por el nombre de las dos tablas separados por un guión bajo.
- ✓ Los nombres compuestos, se escribirán juntos.

2.4.3 Nomenclatura para atributos.

- ✓ Los atributos tendrán nombres identificativos y claros de lo que representan.
- ✓ Los nombres no excederán los 35 caracteres.
- ✓ Las llaves primarias comenzarán con id seguido de un guión bajo y el nombre de la tabla.
- ✓ Para tablas con nombres compuestos la nomenclatura será id guión bajo primer nombre guión bajo segundo nombre, tratando siempre de no superar los 35 caracteres.
- ✓ Los nombres de los atributos compuestos estarán separados por un guión bajo.

2.5 Descripción general del modelo entidad-relación.

El resultado final del modelo físico obtenido para los módulos Dictámenes, Revisiones Laborales y Proceso con Menores, cuentan para el primero de estos con 11 tablas de datos y 8 nomencladoras (ver Anexo 1), para el segundo 4 tablas de datos y 1 nomencladora (ver Anexo 2), y para el tercer módulo 19 tablas de datos y 9 nomencladoras (ver Anexo 3), además se utilizaron las tablas dproceso, datos, dpersona, ddocumento, dnota, dentidad, ntipopromovente, dtrabajadorfiscalia,

dresolucionjudicial, ntipodesicion, nprovincia, nmunicipi y nasaltribunal, pertenecientes al módulo Comunes.

2.5.1 Descripción de las principales tablas del módulo Dictámenes.

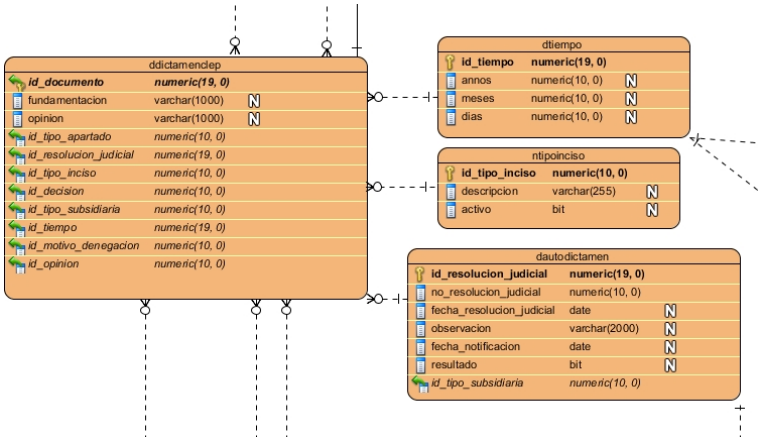

Fig. 6 Tablas principales del módulo Dictámenes.

Entidad	ddictamenclep	
Descripción	Datos generales del dictamen	
Atributos	**Tipo**	**Descripción de los atributos**
id_documento	numeric (19)	Identificador del documento.
id_tipo_apartado	numeric (10)	Identificador del apartado relacionado con el documento.
id_resolucion_judicial	numeric (19)	Identificador de la resolución por la que se realiza el dictamen.
Id_tipo_inciso	numeric (10)	Identificador del inciso por el cual se realiza el dictamen.
Id_decision	numeric (10)	Identificador del tipo de decisión que se tomará.
Id_tipo_subsidiaria	numeric (10)	Identificador de la subsidiaria a la que está relacionada el dictamen
Id_tiempo	numeric (10)	Identificador del tiempo que se demora el proceso del dictamen.
Id_motivo_denegacion	numeric (10)	Identificador del motivo por el cual se deniega o no el dictamen.
Id_opinion	numeric (10)	Identificador de la opinión que se recoge en el dictamen.
fundamentacion	varchar (255)	Fundamento por el cual se realiza el dictamen.
opinion	varchar (255)	Opinión recogida en el dictamen.

Tabla 2 Entidad ddictamenclep del módulo Dictámenes.

Entidad		dautodictamen
Descripción		Datos que contiene el autodictamen
Atributos	**Tipo**	**Descripción de los atributos**
id_resolucion_judicial	numeric(19)	Identificador de la resolución.
Id_tipo_subsidiaria	numeric (10)	Identificador de la subsidiaria ha la que está relacionada el dictamen.
no_resolucion_judicial	varchar(255)	Número de la resolución por el que se realiza el autodictamen.
fecha_ resolucion_judicial	date	Fecha en la que se elaboró la resolución judicial
observacion	varchar (255)	Observaciones que se realizaron para el autodictamen.
fecha_notificacion	date	Fecha en la que se notificó el autodictamen.
resultado	boolean	Resultado del autodictamen.

Tabla 3 Entidad dautodictamen del módulo Dictámenes.

Entidad		dtiempo
Descripción		Datos del tiempo de demora para el dictamen.
Atributos	**Tipo**	**Descripción de los atributos**
id_tiempo	numeric(19)	Identificador del tiempo.
anos	numeric(10)	Años que durará el proceso.
meses	numeric(10)	Meses que durará el proceso
dias	numeric(10)	Días que durará el proceso

Tabla 4 Entidad dtiempo del módulo Dictámenes.

Entidad		ntipoinciso
Descripción		Incisos por los cuales se realizan los dictámenes
Atributos	**Tipo**	**Descripción de los atributos**
id_tipo_inciso	numeric(19)	Identificador del inciso.
descripcion	Varchar(255)	Denominación del tipo de inciso.
activo	boolean	Si la clase está activa en la base de datos o no.

Tabla 5 Entidad ntipoinciso del módulo Dictámenes.

2.5.2 Descripción de las principales tablas del módulo Revisiones Laborales de PDC.

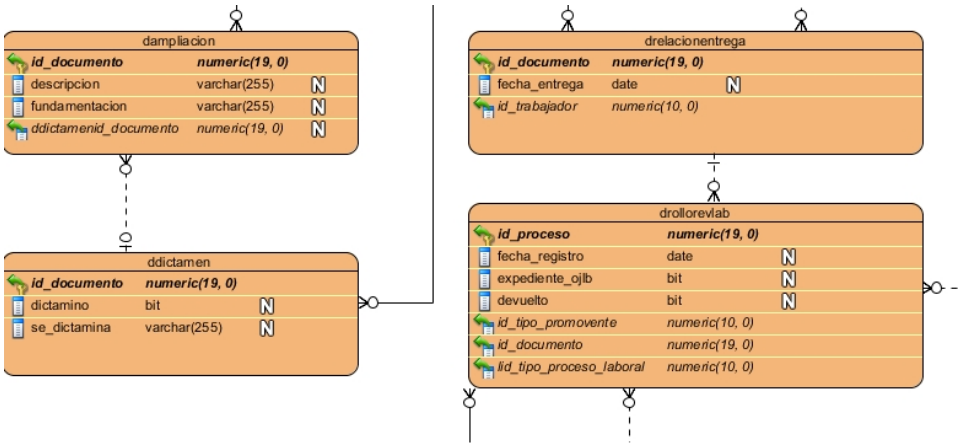

Fig. 7 Tablas principales del módulo Revisiones Laborales de PDC.

Entidad	ddictamen	
Descripción	Datos a registrar sobre un dictamen.	
Atributos	**Tipo**	**Descripción de los atributos**
id_documento	numeric(19)	Identificador del dictamen.
dictamino	boolean	Si se realizó o no el dictamen.
se_dictamino	varchar(255)	Razón por la que no se dictamina o lo que se dictamina.

Tabla 6 Entidad ddictamen del módulo Revisiones Laborales de PDC.

Entidad	drollorevlab	
Descripción	Proceso para las revisiones laborales	
Atributos	**Tipo**	**Descripción de los atributos**
id_proceso	numeric(19)	Identificador del proceso para las revisiones laborales.
id_tipo_promovente	numeric(10)	Ente que promueve el proceso.
id_documento	numeric(19)	Identificador del documento.
id_tipo_proceso_laboral	numeric(10)	Identificador del tipo de proceso laboral.
fecha_registro	date	Fecha en la que se registró el proceso.
expediente_ojb	boolean	Define si los expedientes cuentan con expedientes OJB.
devuelto	boolean	Define si el expediente fue devuelto o no.

Tabla 7 Entidad drollorevlab del módulo Revisiones Laborales de PDC.

Entidad		drelaciónentrega
Descripción		Relación para la entrega de la documentación para el proceso
Atributos	**Tipo**	**Descripción de los atributos**
id_documento	numeric(19)	Identificador del documento que se entrega.
id_trabajador	numeric(10)	Identificador del trabajador que recibe el documento.
fecha_entrega	date	Fecha en la que se hace entrega del documento.

Tabla 8 Entidad drelaciónentrega del módulo Revisiones Laborales de PDC.

Entidad		dampliacion
Descripción		Datos a plantear sobre una queja.
Atributos	**Tipo**	**Descripción de los atributos**
id_documento	numeric(19)	Identificador del documento que se ampliará.
id_dictamen_documento	numeric(19)	Identificador de la ampliación del dictamen.
descripcion	varchar(255)	Descripción de la ampliación.
fundamentacion	varchar(255)	Fundamentación de una ampliación realizada.

Tabla 9 Entidad dampliacion del módulo Revisiones Laborales de PDC.

2.5.3 Descripción de las principales tablas del módulo Proceso con Menores de PDC.

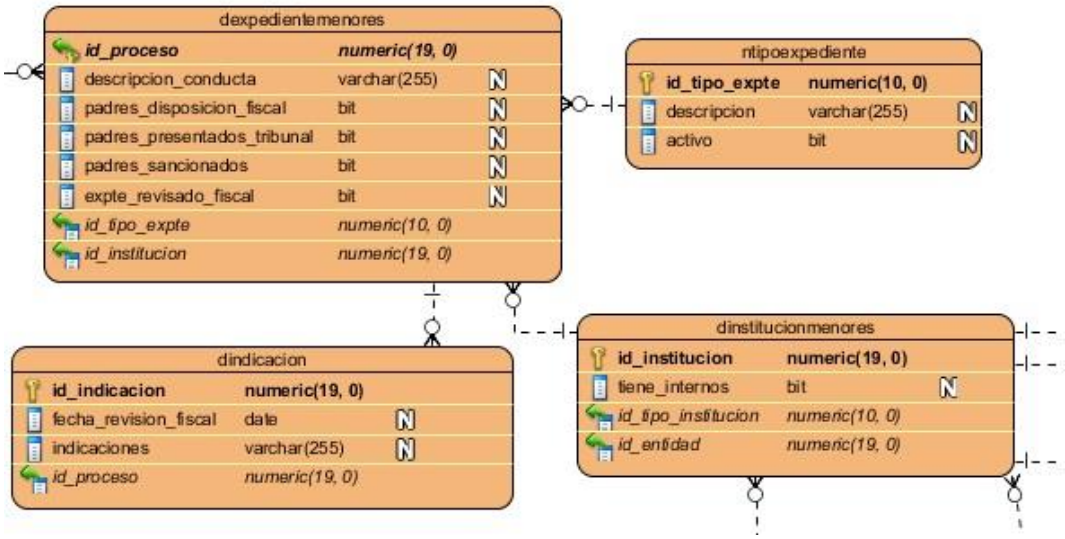

Fig. 8 Tablas principales del módulo Proceso con Menores de PDC.

Entidad		dexpedientemenores
Descripción		Representa el proceso de abrir un expediente
Atributos	**Tipo**	**Descripción de los atributos**
id_proceso	numeric(19)	Identificador del proceso de abrir un expediente.
id_tipo_expte	numeric(10)	Identificador del tipo de expediente
id_institucion	numeric(19)	Identificador de la institución.
descripcion_conducta	varchar(255)	Descripción de la conducta o delito.
padres_disposicion_fiscal	boolean	Define que los padres han sido puestos a disposición del fiscal.
padres_presentados_tribunal	boolean	Define que los padres fueron presentados ante el tribunal.
padres_sancionados	boolean	Define que los padres han sido sancionados.
expte_revisado_fiscal	boolean	Define que el expediente ha sido revisado por el fiscal.

Tabla 10 Entidad dexpedientemenores del módulo Proceso Menores de PDC.

Entidad		dindicacion
Descripción		Conlleva el proceso de revisar un expediente de un menor de edad por el fiscal y las observaciones que sobre él realiza.
Atributos	**Tipo**	**Descripción de los atributos**
id_indicacion	numeric(19)	Identificador de la clase.
id_proceso	numeric(19)	Identificador del proceso
fecha_revision_fiscal	date	Fecha en que el fiscal revisa el expediente.
indicaciones	varchar(255)	Descripción de las indicaciones que el fiscal hace sobre el expediente.

Tabla 11 Entidad dindicacion del módulo Proceso Menores de PDC.

Entidad		dinstitucionmenores
Descripción		Clase que surge de los atributos adicionales que genera la relación de una persona (menor) con la institución a la que pertenece.
Atributos	**Tipo**	**Descripción de los atributos**
id_institucion	numeric(19)	Identificador de la clase.
id_tipo_institucion	numeric(10)	Identificador del tipo de institución
id_entidad	Numeric(19)	Identificador de la entidad
tiene_internos	boolean	Si tiene o no internos

Tabla 12 Entidad dinstitucionmenores del módulo Proceso Menores de PDC.

Entidad		ntipoexpediente
Descripción		Nomenclador del tipo de expediente
Atributos	**Tipo**	**Descripción de los atributos**
id_tipo_expte	numeric(19)	Identificador de la clase

descripcion	Varchar(255)	Denominación del tipo de expediente.
activo	boolean	Si la clase está activa en la base de datos o no.

Tabla 13 Entidad ntipoexpediente del módulo Proceso Menores de PDC.

2.5.4 Patrones utilizados en el diseño.

Los patrones utilizados en la solución dada del modelo, complementan el objetivo de encontrar formas comunes para solventar problemas de diseño.

- ✓ El Patrón llave subrogada: Este patrón se encuentra dentro de los más utilizados en el diseño de base de datos, es usado para generar una llave primaria única en cada una de las tablas del modelo. Está creada de forma numérica (*numeric*) de tamaño 19, ejemplo de esta tenemos idmedidaimpuesta_medida del módulo Proceso con Menores de PDC.

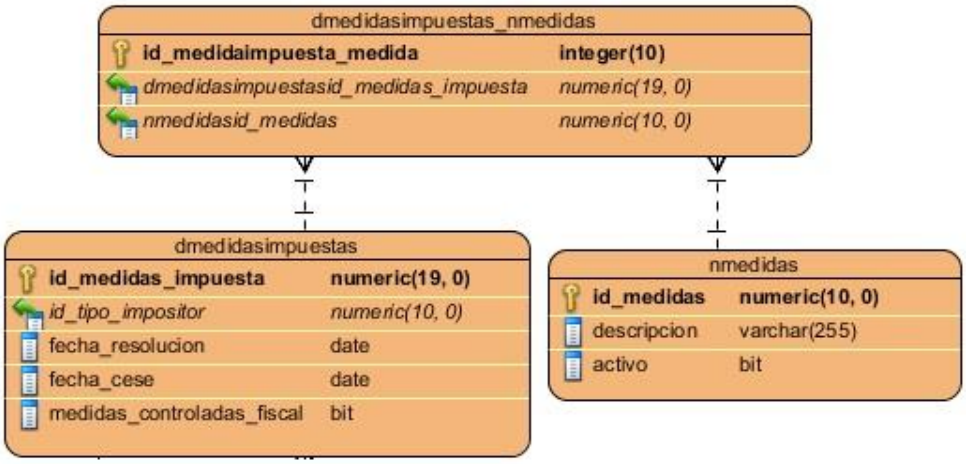

Fig. 9 Patrón Llave Subrogada

- ✓ Patrón árbol fuertemente codificado: En este tipo de patrón las entidades son asociadas a un nivel del árbol. Normalmente constituyen relaciones de 1 a muchos (n). Normalmente utilizado para representar jerarquías donde es bien conocida la estructura y es importante representar la correspondencia. Este patrón es utilizado donde los cambios estructurales que sean representados sean poco probables. Además admite tantos niveles como requiera la jerarquía que se vaya a representar.

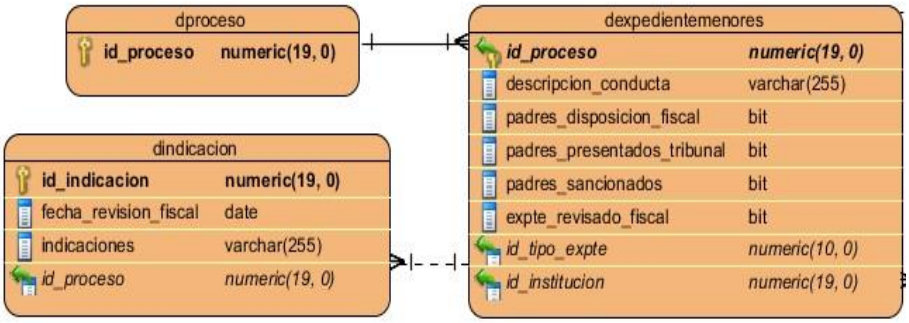

Fig. 10 Patrón Árbol Fuertemente Codificado

2.6 Optimización.

Las estrategias para optimizar una base de datos son de gran variedad, parte de ellas están dirigidas al funcionamiento del gestor de base de datos, puntualizando configuraciones de acceso, optimizando las consultas, y otras estarán centradas en la limpieza de los datos que se obtenga del modelo seleccionado.

2.6.1 Normalización del modelo.

Los modelos obtenidos para los módulos Dictámenes, Proceso con Menores y Revisiones Laborales, se encuentran en primera forma normal pues se puede asegurar que cada tupla contiene exactamente un valor para cada atributo de las tablas de la base de datos, o sea, no existen campos multivaluados, ni compuestos, además tener un único identificador . Además se puede decir que están también en segunda forma normal, pues se encuentran en primera forma normal y todos los atributos que no son claves en las tablas, dependen totalmente de la clave primaria. Además está en tercera forma normal, al encontrarse en segunda forma normal y no tener dependencias transitivas los atributos no primos en una misma relación.

2.6.2 Mantenimiento del SGBD (VACUUM).

Esta opción se basa en el comando VACUUM de Postgres. Este comando no es propio del estándar SQL92. Esta opción escaneará la BD o tabla por filas. Si una fila es modificada o eliminada, su contenido previo no será reemplazado, pero será marcado como no válido y no se podrá usar. El nuevo dato será insertado en la BD, este si será accesible. Se necesita regularmente realizar una recolección de basura, para asegurarse que la BD no contenga demasiados datos sin uso y se afecte el rendimiento de la misma (Ulacia., 2011) .

VACUUM sirve para dos propósitos en Postgres como medio para reclamar almacenamiento, y también para recolectar información para el optimizador. VACUUM abre cada clase en la base de datos, limpia los registros de transacciones ya pasadas y actualiza las estadísticas en los catálogos del sistema. Las estadísticas mantenidas incluyen el número de tuplas y el número de páginas almacenadas en todas las clases. La ejecución de VACUUM periódicamente aumentará la velocidad de la base de datos al procesar las consultas del usuario (Ulacia., 2011). Este comando además permite redefinir los índices que han quedado desbalanceados dentro de la BD.

La utilización del comando VACUUM en la solución propuesta es justificable después de haber realizado un constante trabajo de modificaciones sobre alguna de las tablas de los módulos. La aplicación del comando se realizará fuera de las horas laborales.

2.7 Acceso a datos.

La utilización de framework de mapeo relacional de objetos (ORM), facilita a los desarrolladores un mejor manejo de los datos en la aplicación, una mejor gestión y entendimiento.

El framework ORM Doctrine 2, para el mapeo de la base datos, utiliza anotaciones en las clases php que identifican con igual nombre y atributos a sus similares en la base de datos.

Ejemplo:

```
/**
 * @ORM\Entity(repositoryClass = "PDC\MenoresBundle\Entity\ResolucionRepository")
 * @ORM\Table(name = "pdc.dresolucionmenores")
 * Documentation: Representa el documento Resoluci□n usado por varios procesos.
 */
class ResolucionMenores extends Documento {

    /**
     * @ORM\Column(name="numero_resolucion", type="decimal", nullable = false)
     * @Assert\NotNull()
     * Documentation: el n□mero de resoluci□n
     */
    private $numero_resolucion;

    /**
     * @ORM\Column(name="numero_expediente", type="decimal", nullable = false)
     * @Assert\NotNull()
     * Documentation: n□mero del expediente al que pertenece la resoluci□n
     */
```

Fig. 11 Código del ORM Doctrine 2 sobre clases entidades.

Para la manipulación y acceso a los datos, el framework utiliza su propio lenguaje de consulta, *Doctrine Query Language*, lenguaje de consulta de Doctrine, por sus siglas DQL. La utilización de esta vía media de una clase repositorio creada a partir de identificar la entidad con más fuerza en algún proceso. Ejemplo de alguna consulta:

```
public function listarRolloPaso($id_paso_actual, $lista_paso_anteriores) {
    $qb = $this->getEntityManager()->createQueryBuilder();
    $qb->select('pq')
            ->from('RevLaboralBundle:RolloRevLab', 'pq')
            ->innerJoin('pq.tipo_proceso_laboral', 'q')
            ->leftJoin('pq.personas_proceso', 'pp')
            ->leftJoin('pq.documentos', 'doc')
            ->leftJoin('doc.tipo_documento', 'tdoc')
            ->leftJoin('pq.entidades_proceso', 'enp')
            ->leftJoin('pp.persona', 'pr')
            ->leftJoin('pp.tipo_persona', 'tp')
            ->leftJoin('pq.expedientes', 'e')
            ->leftJoin('e.municipio', 'm')
            ->leftJoin('m.provincia', 'pv')
            ->leftJoin('e.tipo_tribunal', 'tt')
            ->leftJoin('pq.tipo_promovente', 'p')
            ->innerJoin('pq.estado', 'ep')
            ->innerJoin('ep.estado', 'ee')
            ->innerJoin('ep.paso', 'ps');
    if ($id_paso_actual == ConfigUtil::paso_ampliar_dictamen_rl) {
                $qb ->from('RevLaboralBundle:Dictamen', 'dic')
                 ->andWhere(($qb->expr()->eq('dic.id_documento', 'doc.id_documento')))
                 ->andWhere($qb->expr()->eq('dic.dictaminc', 'false'));
```

Fig. 12 Código DQL del ORM Doctrine 2.

2.7.1 Asignación de Llaves de ORM.

Las llaves autogeneradas en las entidades son controladas a través de secuencias, para evitar que se inserten valores iguales en las tuplas. El identificador estará conformado por un número de 19 dígitos, que inicialmente se le asigna un prefijo que define la fiscalía donde se genera la información y los restantes números están definidos por el incremento en 1 del dato anterior. Para controlar los números que la secuencia debe generar, se implementó la clase FiscaliaIdGenerator.php, la cual se encarga de controlar la secuencia de llaves que se generan, capturando la cantidad de ceros en la secuencia, el id de la fiscalía donde se está trabajando y el próximo número de la secuencia.

Un ejemplo de identificador generado para tabla ddictamenclep se muestra en la siguiente imagen.

Fig. 13 Identificador generado para una tupla en la entidad ddictamenclep.

La imagen que a continuación se presenta, representa la implementación del método encargado de generar el identificador en la clase FiscaliaIdGenerator.php.

```php
public function generate(EntityManager $em, $entity) {
    $conn = $em->getConnection();
    $class = $em->getClassMetadata(get_class($entity));
    //Obtener la cantidad de Ceros a adicionar a la secuencia
    $strlength = $class->fieldMappings[$class->getSingleIdentifierFieldName()]['precision'] == 0 ? 7 : 16;
    //Obtener la fiscalia desde donde se autentico el user
    $user = $this->container->get('security.context')->getToken()->getUser();
    $fiscalia = $user->getTrabajadorFiscalia()->getFiscalia();
    $id_fiscalia = $fiscalia->getIdOficina();
    //Obtener proximo valor de la secuencia.
    $sequence = '';
    if (empty($class->parentClasses)) {
        $sequence = $class->sequenceGeneratorDefinition['sequenceName'];
    } else {
        $metadata = $em->getClassMetadata($class->parentClasses[count($class->parentClasses) - 1]);
        $sequence = $metadata->sequenceGeneratorDefinition['sequenceName'];
    }
    $sql = $conn->getDatabasePlatform()->getSequenceNextValSQL($sequence);
    //Obtener proximo ID para la fiscalia a salvar
    $max = $conn->fetchColumn($sql);
    //Generar el Identificador con el IdFiscalia.SecuenciasDeCeros.ValorSecuencia
    $str = $id_fiscalia . str_repeat('0', $strlength - strlen($max)) . $max;

    return $str;
}
```

Fig. 14 Método de asignación de llaves a través del ORM.

2.8 Conclusiones parciales.

En este capítulo se realiza una detallada descripción de la solución obtenida en el diseño e implementación de los módulos Dictámenes, Proceso con Menores y Revisiones Laborales de la aplicación SIGEF II. Donde se explican las configuraciones usadas con las herramientas de desarrollo, la arquitectura que se usó, la nomenclatura definida para implementar bases de datos; así como la descripción general de las principales tablas del modelo y patrones que se usaron. Como parte de la solución se explican las técnicas de optimización utilizadas para dar mayor calidad al resultado obtenido.

Capítulo 3. Validación y pruebas.

3.1 Introducción

En el presente capítulo se realizan la validación y pruebas a la bases datos, obtenida en la investigación. Abordado mediante el uso de técnicas de validación teórica, como son los tipos de restricciones de integridad para los datos. Además de las pruebas de volumen, rendimiento, carga y estrés para complementar un alto rigor con el cual medir la validación funcional de la solución.

3.2 Validación Teórica.

Obtener un modelo encaminado a las necesidades del usuario final, por sí solo no asegura un correcto funcionamiento de la base de datos, pues velar por aspectos importantes como consistencia, integridad y seguridad, corregirían aquellos posibles errores de datos. Aplicar técnicas desde la aplicación hasta la base de datos, para asegurar los datos se convierte en uno de los mejores procesos a realizar.

3.2.1 Integridad de la base de datos.

La integridad de los datos se gestiona a través de dos elementos fundamentales, el control de transacciones que provee el framework Doctrine y las reglas establecidas para velar por la consistencia de los datos requeridos en las tablas. En el caso del primero evita la pérdida de información a la hora de realizar una transacción de una entidad a otra y el segundo chequea la validez y unicidad de determinados atributos, incluyendo las restricciones impuestas para los tipos de llaves primarias y foráneas. Fueron determinadas de la siguiente forma:

Integridad de entidad: las llaves primarias (Primary Key) de cada entidad son atributos no nulos y únicos.

Integridad Referencial: las llaves foráneas agregadas en una tabla tomaran valores referentes y con concordancia de donde derivan.

Datos Requeridos: Se definieron atributos no nulos, con la cláusula (not null), en las tablas por la importancia que contienen a la hora de agregar datos. Se puede tomar como ejemplo el campo fecha_resolucion_judicial de la tabla dautodictamen.

Chequeo de Validez: Se definieron restricciones sobre los tipos de datos utilizados en las columnas, para chequear que los valores agregados cumplan con un dominio.

Dominio	Tipo de Datos	Campos donde aparece
Descripción	Varchar(255)	nombre, descripción, primer_apellido, recomendaciones
Comentario	Texto	Comentario, fundamentación, dictamen.
id_tabla	Numeric(19)	Para identificadores de tablas de datos.
id_nomenclador	Numeric(10)	Para identificadores de tablas nomencladoras.
fecha	Date	fecha, fecha_notificación, fecha_desde, fecha_hasta
Bool	Bit	provisional, activo, actual

Tabla 14 Descripción de integridad de dominio

3.2.2 Transacciones.

La arquitectura del proyecto define que el control de transacciones realizado desde la aplicación hacia la base de datos se haga a través del framework Doctrine 2, encargado de velar por las operaciones de escritura (INSERT/UPDATE/DELETE). El mismo ofrece dos formas de realizar esta acción: La primera es ubicándolas en una cola, hasta que se invoca EntityManager#flush () el cual envuelve todos estos cambios en una sola transacción (enfoque implícito); y la otra opción es realizar este control desde un punto vista propio (enfoque explícito) (Doctrine, 2013). Ejemplos:

```php
<?php
// $em es instancia de EntityManager
$user = new User;
$user->setName('George');
$em->persist($user);
$em->flush();
```

Fig. 15 Ejemplo de Enfoque Implícito

```php
<?php
// $em es instancia de EntityManager
$em->getConnection()->beginTransaction(); // susp
try {
    //... realiza alguna tarea
    $user = new User;
    $user->setName('George');
    $em->persist($user);
    $em->flush();
    $em->getConnection()->commit();
} catch (Exception $e) {
    $em->getConnection()->rollback();
    $em->close();
    throw $e;
}
```

Fig. 16 Ejemplo de Enfoque Explícito

El tipo de enfoque utilizado para controlar las transacciones en SIGEF II, es el segundo; ya que permite definir parámetros específicos sobre los datos a insertar, actualizar o eliminar.

3.3 Validación funcional.

La capacidad funcional que pueda brindar una base de datos, respecto al modelo obtenido, el gestor utilizado y las características de hardware que ayudan al proceso, siempre ha sido útil probarlas, pues de la misma podrían partir estrategias para mejorar el rendimiento final. Pruebas factibles para medir resultados del funcionamiento de una base de datos son las de volumen, rendimiento y las pruebas de carga y estrés.

3.3.1 Pruebas de volumen.

Estas pruebas buscan dar una estimación de hasta dónde se puede llegar cargando el sistema antes de que sea inutilizable (GlobeTesting, 2013). Esta se convierte en una de las pruebas más comunes para sistemas que utilizan bases de datos, por lo que es necesario realizar un grupo de investigaciones sobre la cantidad máxima de datos que puede almacenar el gestor de bases de datos.

Para tener una medida de la cantidad de datos que se generan en las fiscalías, es utilizada la tabla ddictamenclep del modelo del módulo Dictámenes, pues gran parte de las actividades de inserción de datos en el proceso dentro del módulo hacen referencia a esta entidad. Se estima un total aproximado de 20000 procesos anuales entre todas las fiscalías del país, Por lo que se estima que no varíen durante los próximos 5 años. Por consiguiente el total de tuplas a insertar en prueba serían 100000.

El ambiente de este tipo prueba se restringe a condiciones inferiores a las existentes en la Fiscalía General de la República. Utilizándose una máquina con microprocesador Intel(R) Core(TM) 2 Duo, a 2.20 GHz, que utiliza a su vez una Memoria de Acceso Aleatorio (RAM por sus siglas en inglés) de 1 gigabyte.

La herramienta de generación de datos aleatorios Datanamic Data Generator for PostgreSQL V5, permite generar la cantidad de datos aleatorios necesarios para la prueba, prestando la posibilidad de configurar los tipos de datos, para que los resultados se asemejen a lo esperado por el equipo de desarrollo.

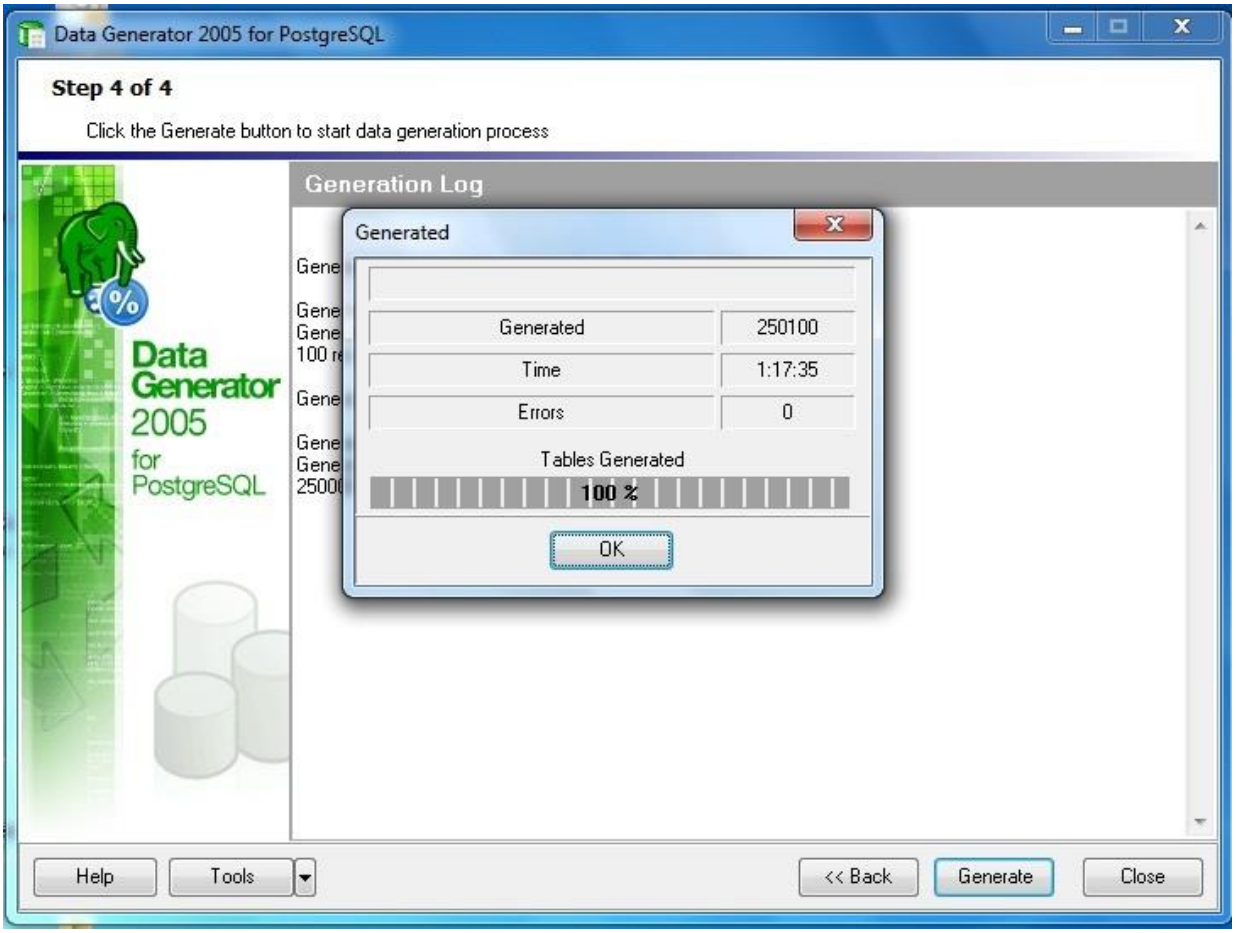

Fig. 17 Prueba de volumen

Como resultado de la prueba realizada se concluye, que la base datos puede soportar la carga de datos que se genera en las fiscalías del país, sin presentar límites de capacidad, desbordamiento de columnas, atributos o tipos de datos. Para esta prueba se tuvo en cuenta también las características de almacenamiento del gestor de base de datos que se muestran en la siguiente tabla.

Límites	Valor
Tamaño máximo de la base datos	Ilimitado
Tamaño máximo de la tabla	32 Terabyte
Tamaño máximo de la fila	1.6 Terabyte
Tamaño máximo del campo	1 Gigabyte
Número máximo de filas por tabla	Ilimitado
Número máximo de columnas por tabla	250-1600 de acuerdo al tipo de dato de la columna
Número máximo de índices por tabla	Ilimitado

Tabla 15 Capacidad de Almacenamiento de PostgreSQL 9.1

3.3.2 Pruebas de rendimiento.

Las pruebas de rendimiento se ejecutan tanto para determinar cómo responde un sistema ante una cierta carga, como para validar otros atributos relacionados con la calidad, como pueden ser la escalabilidad o el uso de recursos entre otros (Testhouse, 2013). En una base de datos se busca medir el tiempo de ejecución de consultas al enfrentar la búsqueda de información frente a un gran grupo de información almacenada.

Por lo que se define un rango de tiempos aceptables para las respuestas de las consultas, el cual se define de la siguiente manera.

Tipo de respuesta	tiempo
Aceptable	Menor a los 3 segundos
Superior	Mayor a los 3 segundos

Tabla 16Tiempos para consultas

Para realizar las consultas se tomó como carga inicial parte de la generada en la prueba de volumen, en la cual las siguientes tablas quedaron con las siguientes cantidades de tuplas:

Entidad	Cantidad de tuplas
ddictamenclep	250000
ddecision	25000
ntipodecision	10
dtiempo	10000

Tabla 17 Datos cargados para prueba de rendimiento

Para la prueba se realizó una consulta, bajo dos tipos de situaciones, y medir su tiempo de ejecución mediante el comando EXPLAIN ANALYZE, que devuelve el resultado en milisegundos (ms). La primera situación es realizar la consulta sobre campos sin indexar, y la segunda a su vez con campos indexados.

Resultados:

Ambiente sin indexado. Tiempo: 32.210ms.

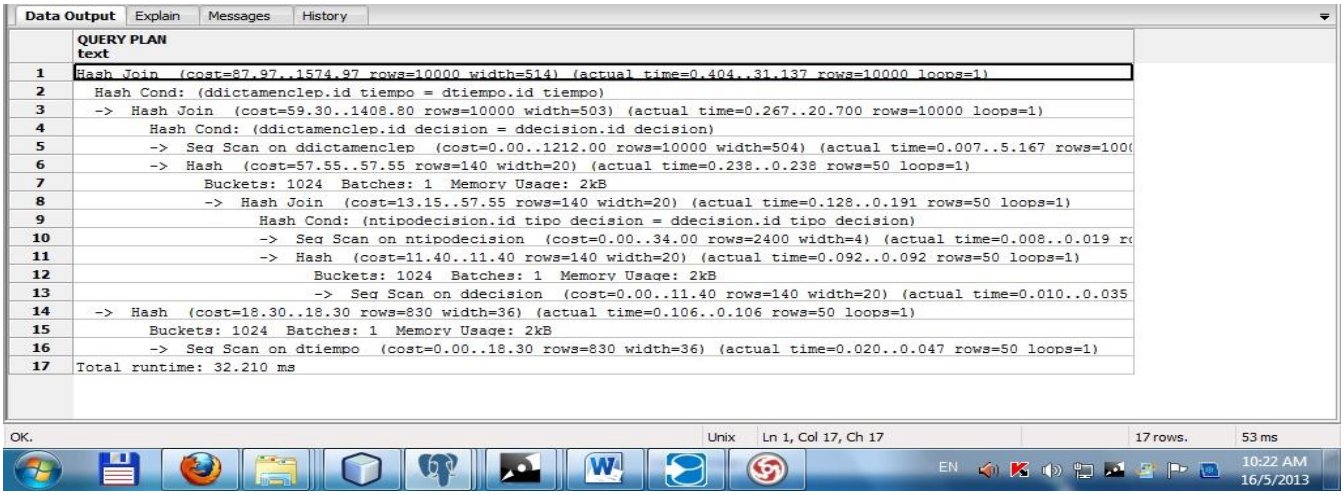

Fig. 18 Resultado de consulta sin indexado.

Ambiente con campos de llaves primarias, llaves foráneas, y columnas id_desicion, id_tipo_desicion e id_tiempo.

Ambiente sin indexado. Tiempo: 2.543ms.

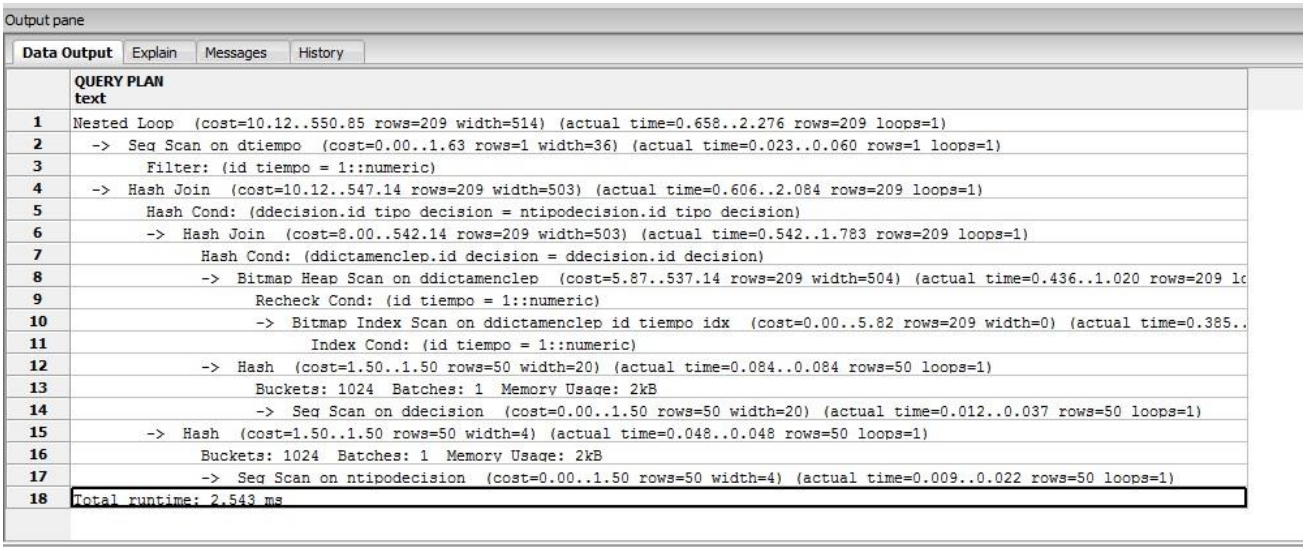

Fig. 19 Resultado de consulta con indexado.

3.3.3 Pruebas de estrés y carga.

El objetivo de estas pruebas es obtener datos, sobre la carga del sistema, que ayuden a realizar el dimensionamiento del sistema, generando carga en el sistema mediante la simulación de concurrencia que se acerque con fiabilidad al esperado en la explotación real. Las pruebas de estrés son uno de los últimos tipos de pruebas que se deben ejecutar, ya que, por su carácter poco realista, podría darse el caso de que la situación de carga simulada nunca se diera en la vida real. (GlobeTesting, 2013).

En las fiscalías de todo el país la cantidad de trabajadores es diferenciada dependiendo de su tipo (Machado, 2012).

Tipo de Fiscalía	Cantidad de trabajadores
Municipal	3-15
Provincial	15-25
Fiscalía General de la Republica	100

Tabla 18 Cantidad de trabajadores por tipo de fiscalía

De acuerdo a estos datos, las pruebas estarán centradas en cubrir el máximo de trabajadores en concurrencia que puedan estar realizando peticiones a la base de datos desde la aplicación. Tomándose entonces la cantidad posible en las fiscalías provinciales y en la Fiscalía General de la República como casos más críticos, por ser los tipos de entidades donde puede acumularse una mayor carga de trabajo. Las pruebas se realizarán utilizando la herramienta de distribución gratuita JMeter versión 2.9, desarrollada con el lenguaje de programación Java en el proyecto Jakarta, la cual permite realizar pruebas sobre aplicaciones web, sobre diferentes bases de datos. JMeter se destaca por su facilidad de uso, versatilidad, estabilidad y la variedad de funcionalidades que brinda.

De las dos pruebas definidas a continuación se muestran las propiedades de los hilos de cada una, 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 período 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 el test.

Valores definidos	Prueba 1	Prueba 2
Numero de hilos	25	100
Periodo de subida	0	0
Contador de bucle	5	5

Tabla 19 Propiedades de los hilos de las pruebas

El informe que muestra como resultado la herramienta define los siguientes campos (Jmeter, 2013).

Etiqueta: Nombre de la etiqueta de muestra.

 # Muestras: El número de muestras con la misma etiqueta.

Media: El tiempo promedio de un conjunto de resultados.

Mediana: La mediana es el tiempo en el medio de un conjunto de resultados.

90% Línea: 90% de las muestras no tardó más de este tiempo.

Min: El tiempo más corto para las muestras con la misma etiqueta

Max: El tiempo más largo para las muestras con la misma etiqueta

Error%: Porcentaje de solicitudes con errores

Rendimiento: El rendimiento se mide en solicitudes por segundo / minuto / hora. La unidad de tiempo se elige de modo que la tasa de muestra es al menos 1,0. Cuando el rendimiento se guarda en un archivo CSV, que se expresa en las peticiones / segundo, es decir, 30.0 peticiones / minuto se guardan como 0.5.

Kb / s: El rendimiento se mide en kilobytes por segundo

Resultados:

	Prueba1	Prueba 2		Prueba 1	Prueba 2
Etiqueta	JDBC Request	JDBC Request	**Mínimo**	1	1
#Muestras	125	500	**Máximo**	1235	1550
Media	245	318	**%Error**	0.00%	0.00%
Mediana	7	107	**Rendimiento**	78.1/sec	243.1/sec
Línea de 90%	1175	1245	**Kb/Sec**	9.5	27.3

Tabla 20 Respuesta de Jmeter Prueba 1 y 2.

Los resultados demuestran que la base de datos puede atender una de las peticiones realizadas en la primera prueba en poco más de un ¼ de segundo en concurrencia y para la segunda se puede aproximar por exceso a medio segundo. Además agrega, que para los dos casos la petición que más se tardaría en atender no superaría por mucho el segundo y medio. Tiempos que a su vez son los esperados como aceptables entre los tiempos de respuestas en las consultas.

3.4 Conclusiones parciales

Después de realizadas las validaciones tanto teóricas como funcionales se puede asegurar que se han cumplido los objetivos propuestos, logrando que la base de datos cumpla teóricamente con restricciones de integridad que posibiliten la calidad de los datos, donde las transacciones serán manejadas desde el ORM Doctrine 2, de la forma explícita. Además se pudo comprobar que los tiempos de respuesta respecto a la carga y consulta de datos son los requeridos como aceptables por el proyecto.

Conclusiones Generales

Luego de realizar el presente trabajo se puede concluir que:

- ✓ A partir del diseño se obtuvo el modelo de datos como artefacto fundamental para el desarrollo de la base de datos de los módulos.
- ✓ Los problemas en el diseño que se presentaron fueron solucionados con la normalización del modelo de datos y la aplicación de diferentes patrones.
- ✓ Las pruebas de volumen, rendimiento, carga y estrés realizadas, demostraron que la base de datos responderá de manera satisfactoria ante gran cantidad de información, las respuestas a las solicitudes son relativamente rápidas y soportan un gran número de conexiones y peticiones de manera concurrente.
- ✓ Se cumplió el objetivo de diseñar e implementar la base de datos para almacenar y gestionar la información de las entidades fiscales de forma segura y con integridad, para los módulos Dictámenes, Revisiones Laborales y Proceso con Menores.

Recomendaciones

- ✓ Continuar utilizando la estrategia de trabajo en los restantes módulos del proyecto SIGEF II.
- ✓ Realizar un trabajo regular de soporte y mantenimiento.
- ✓ Recurrir al modelo de trabajo utilizado para proyectos similares.

Bibliografía

Acosta, Jose Carlos Pupo. 2013. *0120_4 Arquitectura Vista de Datos.* s.l. : Universidad de las Ciencias Informáticas, 2013.

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

Batini, Carlos. 2004. 2004. *Diseño conceptual de bases de datos. 2004.* 2004.

Blaha, M. 2010. *Patterns of Data Modeling.* Estados Unidos : CRC Press, 2010.

Claudia Imhoff, 2003. 2003. *Claudia Imhoff, Nicholas Galemmo, Jonathan G. Geiger. 2003. Mastering Data Warehouse Design. Relational and Dimensional Techniques. Indianapolis : Wiley Publishing, Inc, 2003. ISBN 0-471-32421-3.* Indianapolis : s.n., 2003.

Dataprix. 2013. Dataprix. *Dataprix.* [Online] 2013. [Cited: enero 15, 2012.] http://www.dataprix.com/11-etapas-diseno-bases-datos.

Doctrine. 2013. Doctrine Project. [Online] 2013. [Cited: 4 9, 2013.] http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/transactions-and-concurrency.html.

Elsmari R., Navathe. 1993. *Sistemas de Bases de Datos, Conceptos fundamentales.* s.l. : Addison-Wesley, 1993.

Erl, Thomas. 2009. Revista Universidad EAFIT. [Online] 2009. [Cited: noviembre 27, 2012.] http://publicaciones.eafit.edu.co/index.php/revista-universidad-eafit/article/view/8.

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

GlobeTesting. 2013. GlobeTesting. [Online] 2013. [Cited: 04 26, 2013.] http://www.globetesting.com/pruebas-de-rendimiento/.

—. 2013. GlobeTesting. [Online] 2013. [Cited: 04 23, 2013.] http://www.globetesting.com/pruebas-de-rendimiento/.

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.

Grueso, Cesar David Fernandez. 2009-2010. slideshare.net. [Online] 2009-2010. [Cited: 3 24, 2013.] http://www.slideshare.net/senaticscesar/bases-de-datos-conceptos-basicos.

ICT. 2012. Interactive and Coorporative Technologies Lab. [Online] 2012. [Cited: noviembre 20, 2012.] http://ict.udlap.mx/people/carlos/is341/bases02.html.

Ing. Blanco, Héctor Fuentes. 2012. *Documento Arquitectura de Software SIGEF II.* 2012.

Inmon, W. H. 2005. 2005. *Building the Data Warehouse, Fourth Edition. Indianapolis : Wiley Publishing, Inc., 2005. ISBN: 978-0-7645-9944-6.* Indianapolis : s.n., 2005.

Jmeter, Apache. 2013. The Apache Software Fundation. [Online] 01 22, 2013. www.apache.org.

Linux Six Blog. 2012. Linux Six Blog. [Online] 2012. [Cited: diciembre 5, 2012.] http://linuxsix.blogspot.com/2011/10/rapidsvn-svn-en-linux.html.

Linux.org. http://www.linux-es.org. [Online] Linux. [Cited: 04 2013, 15.] http://www.linux-es.org/node/660.

Machado, Ing. Yenier Figueroa. 2012. *Proyecto Tecnico SIGEF II, CEGEL.* 2012.

Marcelo D. Vinjoy, Et al. 2010. Catedra de Base de Datos . [Online] Facultad de Informática-Universidad de Morón, 2010. [Cited: diciembre 10, 2012.] http://bdatos.wordpress.com/base-de-conocimiento/oracle-perfomance/oracle-considerando-la-introduccion-de-redundancia-desnormalizacion/.

Marqués, M. 2009. *Bases de Datos.* Valencia, España : s.n., 2009.

Martínez, Dr. David Luis la Red. 2013. Facultad de Ciencias Exactas y Naturales y Agrimensura. [Online] 2013. http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/MonogSO/REPLIC02.html.

Mato García, Rosa María. 2005. *Sistemas de Base de Datos.* 2005.

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

Netbeans. 2012. Netbeans. [Online] 2012. [Cited: enero 12, 2013.] http://netbeans.org..

Oracle Corporation. 2012. Oracle Corporation. [Online] 2012. [Cited: 04 25, 2013.] http://www.oracle.com/us/industries/public-sector/public-sector-big-data-br-1676649.pdf.

Oracle. 2013. Oracle. *Guía de administración del sistema: servicios de seguridad.* [Online] 01 22, 2013. http://docs.oracle.com/cd/E24842_01/html/E23286/rbac-1.html.

Orallo, José Hernández. 2002. *La Disciplina de los Sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas.* s.l. : Dep. de Sistemas Informaticos y Computación., 2002.

Osorio, Alain. 2012. *Dbplanning framework.* 2012.

Pérez, M., Valero, E.,Zavala, M. 2011. *Base de Datos Ingeniería de Sistemas.* Universidad politécnica de la fuerza Armada Bolivariana, Zulia, Venezuela : s.n., 2011.

Postgresql Cuba. 2012. Postgresql Cuba, Comunidad Tecnica. [Online] 2012. [Cited: diciembre 2, 2012.] http://postgresql.uci.cu/node/63.

—. 2012. SVN Book. [Online] 2012. [Cited: enero 15, 2013.] http://svnbook.spears.at/nightly/es/svn-ch-1-sect-3.html.

PostgreSQL. 2005. PostgreSQL. [Online] 04 07, 2005. [Cited: 05 02, 2013.] http://www.postgresql.org/about/news/309/.

Pressman. 2005. *Aprendiendo UML en 24 horas.* 2005.

Robealex1. 2011. buenastareas.com. [Online] 2011. http://www.buenastareas.com/ensayos/Temmario-De-Administracion-De-Base-De/1713538.html.

Scridb. 2012. Scridb. [Online] 2012. [Cited: diciembre 5, 2012.] http://es.scribd.com/doc/36636137/Tutorial-Visual-Paradigm.

Testhouse. 2013. Testhouse. [Online] 2013. [Cited: 04 22, 2013.] http://www.es.testhouse.net/pruebas-de-rendimiento/.

Ubuntu, Guía. 2008. Portal-Guía Ubuntu. [Online] 2008. [Cited: diciembre 9, 2012.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

UCV. 2010. Universidad Central de Venezuela. [Online] 2010. [Cited: diciembre 5, 2012.] www.ciens.ucv.ve:8080/genasig/Taller%203%20(Integridad).doc.

Ulacia., Juan Carlos Larrinaga. 2011. *Diseño y configuración de la base de datos para el procedimiento Ordinario de la materia Civil en los Tribunales Municipales Cubanos.* La Habana : Universidad de las Ciencias Informaticas, 2011.

Visual Paradigm. 2013. Visual Paradigm. [Online] 2013. [Cited: enero 14, 2013.] http://www.visual-paradigm.com/aboutus/10reasons.jsp.

Zambrano Ramírez, R. 2008. *Sistemas Gestores de Bases de Datos.* Cordoba : s.n., 2008.

Anexos

Anexo 1

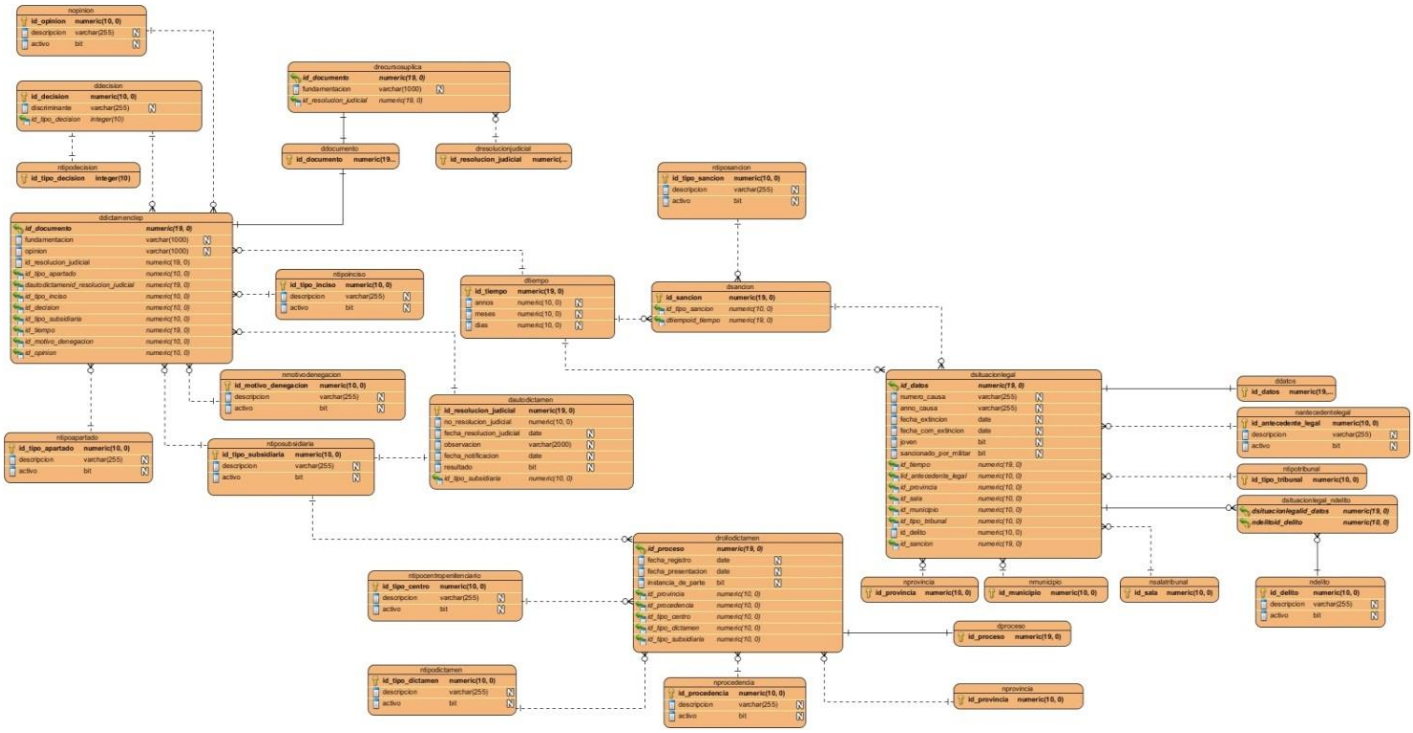

Fig. 20 Modelo de datos de módulo Dictámenes de CLEP.

Anexo 2

El resultado final del modelo físico obtenido para el módulo Revisiones Laborales es el siguiente.

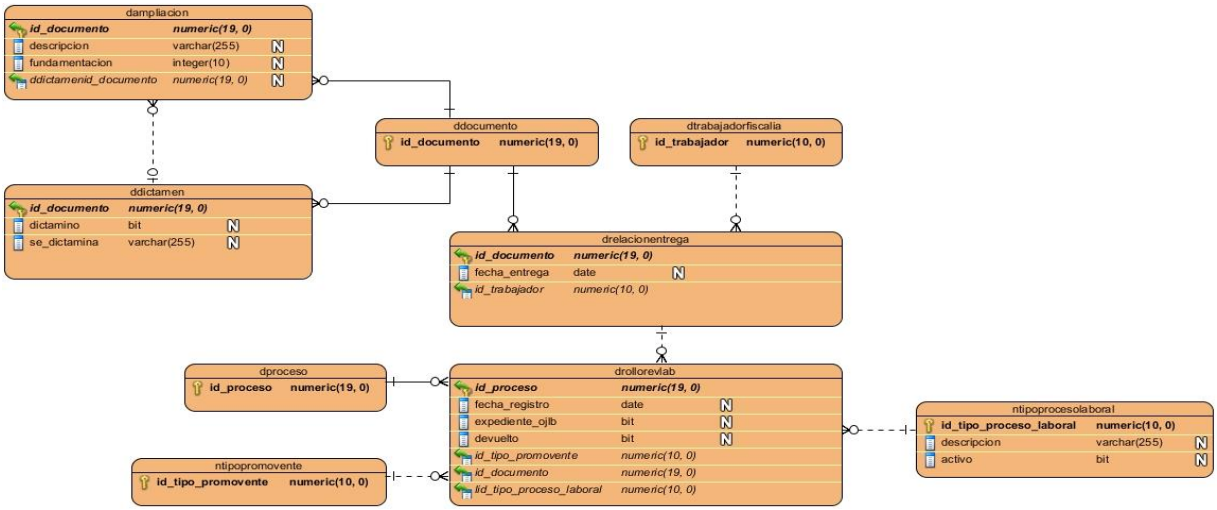

Fig. 21 Modelo de datos de módulo Revisiones Laborales de PDC.

Anexo 3

El resultado final del modelo físico obtenido para el módulo Proceso con Menores es el siguiente.

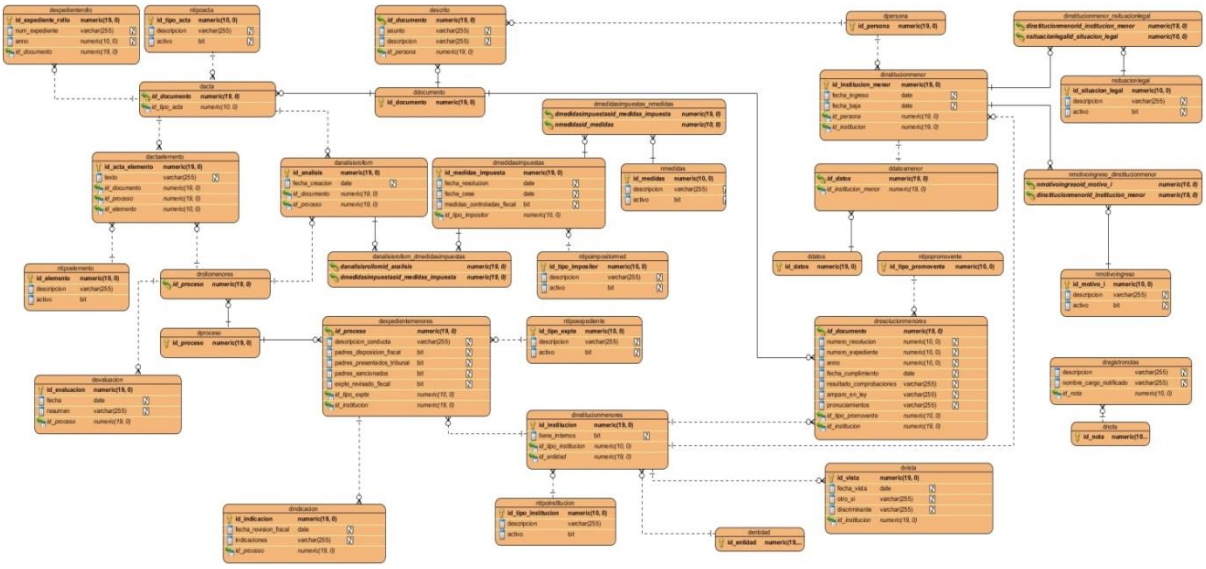

Fig. 22 Modelo de datos de módulo Proceso con Menores de PDC.