UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Propuesta de optimización de la base de datos del proyecto Sistema de Informatización para la gestión de los Tribunales Populares Cubanos.

Autor: Lisandra Nuñez Aguilar.

Tutor: Ing. Darián González Ochoa

Co-tutor: Ing. Yosvany Gómez Perdomo

La Habana, junio de 2013

"Año 55 de la Revolución"

“Hay hombre que luchan un día y son buenos.

Hay otros que luchan un año y son mejores.

Hay quienes luchan muchos años y son muy buenos.

Pero hay los que luchan toda la vida:

Esos son los imprescindibles”.

Bertolt Brecht

DECLARACIÓN DE AUTORÍA

Declaración de autoría

Declaro que soy la única autora de este trabajo y autorizo a la facultad 3 y en especial al proyecto Sistema de Informatización para la gestión de los Tribunales Populares Cubanos de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Lisandra Nuñez Aguilar

Autor

_____ _____

Ing. Yosvany Gómez Perdomo

Tutor

Ing. Darián González Ochoa

Tutor

Agradecimientos

A mi mamita hermosa por apoyarme incondicionalmente y confiar siempre en mí. A ella por ser mi Ángel de la Guarda, mi gran guía y ejemplo.

A mi papá por haberme enseñado muchas cosas importantes a lo largo de mi vida, por darme una educación excelente, por alentarme a seguir adelante y por ser su joya espiritual.

Para ambos todo mi amor, mi esfuerzo y mis grandes logros.

A mis hermanos por quererme tanto y ser cada uno mi motor impulsor, por ellos soy lo que soy hoy y espero que les sirva de ejemplo para su futuro.

A mi Titi adorado por el amor, la dedicación y la confianza que ha depositado siempre en mí. Por estar siempre a mi lado y brindarme su apoyo infinito.

Al resto de mi familia por el apoyo que siempre me han dado en todo momento para que cumpliera uno de mis grandes sueños: graduarme como profesional. Especialmente, a mis abuelos por su cariño infinito, a Papi Gallego que a pesar de que no esté hace años conmigo físicamente siempre ha sido mi paradigma, a Zaily por ser como una hermana para mí y a Ramón por quererme y educarme como si fuera su hija.

A mi Virgencita de la Caridad y al padrecito Dios por escucharme y ayudarme siempre que lo necesite.

A todos mis compañeros que de una forma y otra aportaron su granito de arena para que yo pudiera realizar este sueño.

A mis tutores y educadores que han dedicado horas para convertir en mí, la mujer de bien que soy hoy en especial a Yosvany y Mavi.

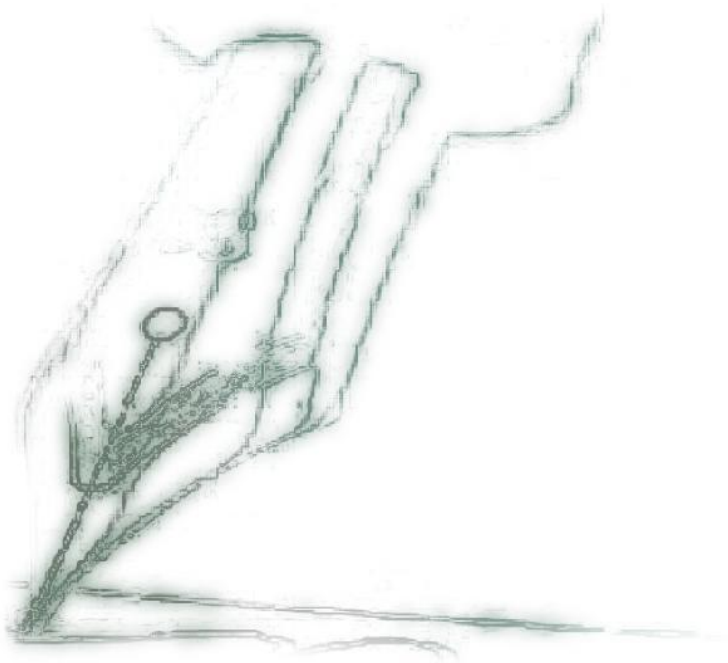
A Fidel por darme la oportunidad de ser una joven universitaria graduada como Ingeniero en Ciencias Informáticas, lo que constituye para mí un orgullo y privilegio.

Por último, quiero agradecerles a todas las personas que de una forma u otra han colaborado con la realización de este trabajo.

Muchas Gracias!

Dedicatoria

A mis familiares que sin el apoyo y el cariño que me han dado no hubiera podido cumplir este sueño. Especialmente a mis padres queridos por su amor infinito y la confianza que han depositado en mí, a mis adorados hermanos, a mis abuelos, a mi amado esposo, a Fidel, a mis tutores y compañeros de la universidad.



Resumen

Con el desarrollo de las tecnologías varias organizaciones se han dado la tarea de informatizar sus procesos así como resguardar la información generada en bases de datos. Por tales razones, emana la importancia del correcto manejo y funcionamiento que deben tener las mismas dentro de una aplicación. El proyecto Sistema de Informatización para la gestión de los Tribunales Populares Cubanos (SITPC) cuenta con una base de datos relacional para almacenar la información referente a los procesos que se desarrollan en los tribunales la cual presenta varios inconvenientes. Con el propósito de resolver dichos problemas se ha desarrollado el presente trabajo, brindando para esto una propuesta de optimización para la base de datos del proyecto. En la medida que se fue desarrollando el mismo se realizó un estudio de las temáticas referente a los patrones de diseño de bases de datos, a las herramientas y lenguajes a utilizar, a un conjunto de métodos que se emplearon en el transcurso de la investigación y a la optimización de las bases de datos haciendo énfasis en los criterios que involucra este proceso. Este trabajo permite que se minimice la redundancia e inconsistencia de la información, se reduzca el espacio de almacenamiento, se disminuyan los tiempos de respuesta para cada petición y a la vez el consumo de recursos.

Palabras Claves: bases de datos, optimización, rendimiento.

Tabla de contenido

Introducción

El vertiginoso desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC's) constituyen un singular avance de la sociedad en las últimos décadas y su impacto e importancia en todas las esferas de la vida pública las han convertido en ente catalizador del desarrollo de la humanidad. Los que tengan el poder de dominarla y usarla consecuentemente sin lugar a dudas estarán ubicados en una posición de vanguardia.

Son muchas las ventajas que ofrecen, que van desde la mejora del rendimiento y calidad de los procesos a los que son aplicadas hasta la obtención de resultados relevantes en materia de ciencia. Actualmente no se puede hablar de mejoras o novedades si no existe una aplicación y correcta utilización de la tecnología.

En Cuba se realizan grandes esfuerzos con el fin de aumentar el uso masivo de las tecnologías en el ámbito económico, político y social en aras de elevar el nivel de vida de su población. Con la premisa de lograr un contacto directo entre las tecnologías y la ciencia del derecho, el sistema judicial cubano, se ha enfrascado en lograr la informatización integral de los procesos judiciales, lo que posibilitará la rápida localización de los asuntos, el control y alerta de vencimiento de los términos, la ágil reproducción de resoluciones y la estandarización de la amplia gama de modelos utilizados en los tribunales, así como el envío digital y automático de documentos a instituciones. (Navarro 2010)

En tal sentido, la Universidad de las Ciencias Informáticas (UCI) ha participado activamente en el programa de Informatización de la Sociedad Cubana que se ha venido consolidando en estos últimos años. Esta universidad cuenta con un centro de desarrollo dedicado a implementar los preceptos de la informática jurídica, el Centro de Gobierno Electrónico perteneciente a la facultad 3, el cual está a cargo de desarrollar el proyecto Sistema de Informatización para la gestión de los Tribunales Populares Cubanos (SITPC) surgido a raíz del acuerdo de colaboración ANC11-001-001 establecido con el Tribunal Supremo Popular (TSP).

SITPC inicia en esta universidad como necesidad del TSP del país de cumplir con una de sus proyecciones estratégicas para el año 2009: continuar desarrollando la infraestructura tecnológica de los tribunales populares y asegurar su gestión, administración y explotación; basándose en los principios de la seguridad informática. (Pérez 2012)

INTRODUCCIÓN

El Tribunal Supremo Popular tiene como función ejercer la máxima autoridad judicial en el país. A través de su Consejo de Gobierno, ejerce la iniciativa legislativa en cuanto a la administración de justicia y la potestad reglamentaria. Toma decisiones y dicta normas generales de obligado cumplimiento por todos los tribunales y, sobre la base de la experiencia de estos, imparte instrucciones de carácter obligatorio para establecer una práctica judicial uniforme en la interpretación y aplicación de la Ley. (Quesada 1997)

Los Tribunales Populares Cubanos (TPC) están estructurados por tres instancias, las cuales se clasifican en: Tribunal Municipal Popular, Tribunal Provincial Popular y Tribunal Supremo Popular, cada uno de los cuales cuenta con diferentes procedimientos distribuidos en cinco materias (Pérez 2012):

- Materia Administrativa.
- Materia Civil.
- Materia Económica.
- Materia Penal.
- Materia Laboral.

El sistema está compuesto por 7 subsistemas que corresponden a cada una de las materias así como el subsistema de Administración y Gobierno (AG) y Común. La información recogida en cada uno de estos subsistemas es almacenada en una base de datos relacional la cual presenta varios inconvenientes. De manera general el diseño de la misma está más enfocado y condicionado a los conocimientos y habilidades de los desarrolladores que a la necesidad de los requerimientos del cliente. En consecuencia, se evidencia que el manejo de temas tan sensibles como: minimización del espacio de almacenamiento, disminución de los tiempos de respuesta y del consumo de recursos, no han constituido prioridades durante el proceso de diseño de la base de datos. Además, se debe mencionar que el empleo de patrones de diseño, se ha realizado de manera casual y no siempre de forma correcta. La documentación y artefactos generados durante el proceso de diseño e implementación de la misma son insuficientes, debido a que no permite conocer a ciencia cierta el diseño realizado. También son abordadas en términos muy generales lo que imposibilita tener un enfoque preciso de la magnitud y conformación del diseño propuesto.

Los criterios anteriormente expuestos demuestran la existencia de errores de diseño en el modelo de datos, problemática que trae como consecuencia que existan redundancias e inconsistencias en la información y la necesidad de incrementar el rendimiento de la base de datos del proyecto, con el propósito de aumentar sus potencialidades y beneficios.

Tomando como antecedente la situación descrita, se identifica el siguiente **problema a resolver**: ¿Cómo incrementar el rendimiento en la base de datos del proyecto Sistema de Informatización para la gestión de los Tribunales Populares Cubanos de forma tal que contribuya a agilizar la ejecución de los procesos en la aplicación?

Por consiguiente el **objeto de estudio** de la investigación es: Optimización de bases de datos relacionales. Como **objetivo general** de la misma se define: Realizar una propuesta de optimización de la base de datos del proyecto SITPC, mediante la implementación de criterios de optimización. Como **campo de acción** se tiene: Optimización de la base de datos relacional del proyecto SITPC, lo que trae consigo la siguiente **idea a defender**: con la implementación de criterios de optimización en la base de datos del proyecto SITPC se contribuirá a incrementar el rendimiento y agilidad en la ejecución de los procesos a gestionar.

Para el correcto cumplimiento del objetivo general trazado en la investigación, se desglosan los siguientes **objetivos específicos**:

1. Realizar un estudio del marco teórico referente a incrementar el rendimiento de bases de datos relacionales.
2. Elaborar una propuesta para incrementar el rendimiento de la base de datos del proyecto Sistema de Informatización para la gestión de los Tribunales Populares Cubanos.
3. Validar la propuesta de solución mediante la realización de pruebas que evalúen el rendimiento de la base de datos.

Para alcanzar los objetivos propuestos se utilizaron un conjunto de métodos científicos, tales como:

Métodos Teóricos:

Analítico – sintético: Este método se utilizó con el propósito de dividir la investigación en dos procesos fundamentales: el análisis y la síntesis. El análisis permitió adquirir

una mejor comprensión de la estructura y del momento histórico en el que se encuentra el proyecto SITPC. Además de realizar un estudio de temas relacionados con las herramientas y lenguajes a utilizar, patrones de diseño de bases de datos existentes, entre otros aspectos. Mediante la síntesis se unieron las partes previamente analizadas para realizar la propuesta de optimización de la base de datos del proyecto.

Análisis Histórico – lógico: Su utilización se sustenta en la necesidad de analizar y estudiar la trayectoria referente a cómo elevar el rendimiento de base de datos relacionales y acontecimientos en el decursar de su historia, elaborando para esto, el estudio del estado del arte de dicho tema.

Modelación: Este método teórico permite la construcción de modelos, descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio. Se utilizó para la creación de abstracciones que explican la realidad, específicamente cuando se realizó el modelo lógico de la base de datos. Además se ve presente en la construcción del diseño físico de la misma.

Métodos Empíricos:

Entrevista: Mediante este método se realizaron entrevistas a especialistas con experiencias en el área de conocimiento de base de datos, con el propósito de profundizar en las diferentes temáticas a tratar para lograr un correcto desarrollo de la investigación.

Medición: Se utiliza con el objetivo de obtener información numérica referente al comportamiento del rendimiento de la base de datos del proyecto SITPC mediante el uso de diferentes procedimientos que permiten determinar las tendencias en cuanto al mismo.

El presente trabajo de diploma consta de 3 capítulos, a continuación se evidencia una breve descripción de cada uno de ellos:

Capítulo 1. Fundamentación teórica: En este capítulo se especifican un conjunto de enfoques teóricos e investigativos sobre el lenguaje y las herramientas a utilizar. Además se abordan temas relacionados con los criterios de optimización y patrones de diseño que se utilizan para incrementar el rendimiento de bases de datos relacionales.

Capítulo 2. Propuesta de solución: En este capítulo se realiza básicamente la propuesta de solución partiendo del estudio realizado en el capítulo anterior. Además se actualizan los artefactos Entorno de Desarrollo, Configuraciones y la Vista de Datos del documento de Arquitectura.

Capítulo 3. Validación de la propuesta: En este capítulo se valida la propuesta de solución desarrollada mediante la aplicación de varias pruebas de rendimiento.

Capítulo 1: Fundamentación teórica.

1.1. Introducción.

La fundamentación teórica constituye una de las partes imprescindible para el desarrollo de la investigación, pues orienta su dirección, amplía el horizonte investigativo y provee un marco de referencia para interpretar los resultados. En el presente capítulo se especifican un conjunto de enfoques teóricos e investigativos sobre los lenguajes y las herramientas a utilizar; así como la arquitectura de la base de datos del proyecto. Además se abordan temas relacionados con los criterios y patrones de diseño que se utilizan para elevar el rendimiento de bases de datos relacionales.

1.2. Conceptos asociados al dominio del problema.

Con el propósito de lograr una mejor comprensión en las temáticas a abordar en el presente trabajo, se mostrará un conjunto de términos correspondiente a los Sistemas Gestores de Bases de Datos (SGBD) y a la optimización de bases de datos relacionales.

1.2.1. Base de datos.

La Licenciada Rosa María Mato García expresa que una base de datos es: *"un conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, una base de datos puede considerarse una colección de datos variables en el tiempo".* (García 1999)

Teresa Garzón Pérez las define como: *"un sistema computarizado para mantener información de un individuo o de una organización y hacer que esté disponible cuando se solicite".* (Pérez 2010)

Otros autores como Silvia López Riquelme y Mirtha Cepero Morales expresan que una base de datos no es más que: *"un conjunto ordenado e interrelacionado de los datos de una organización cualquiera, que tiene relación con su actividad operativa como con su proyección táctica como estratégica".* (Riquelme and Morales 2010)

1.2.2. Sistema Gestores de Base de Datos.

Para definir, crear y mantener las bases de datos, así como proporcionar y controlar el acceso a estas, son usados los Sistemas Gestores de Bases de Datos (SGBD). Varios autores, han planteado diferentes definiciones acerca de estos, por ejemplo, Sara Álvarez, expresa que: *"un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos".* (Álvarez 2010)

Estos permiten la utilización y la actualización de los datos almacenados en una o varias bases de datos por uno o varios usuarios desde diferentes puntos de vista. Además permiten almacenar y acceder a los datos de las bases de datos de forma rápida y estructurada. Sirven de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. (García 2005)

Los SGBD deben presentar los siguientes servicios (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.

Mediante el análisis de los conceptos antes mencionados se ha llegado a la conclusión que los SGBD son sistemas que se encargan entre otras funciones, de gestionar de manera legible, la información contenida en las bases de datos.

Los mismos deben cumplir los siguientes objetivos (González and Díaz 2011):

- **Abstracción de la información.** Ahorran a los usuarios detalles acerca del almacenamiento físico de los datos.

- **Independencia.** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Consistencia.** En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad que tiene determinadas condiciones.

- **Seguridad.** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentre segura de permisos a usuarios y grupos de usuarios.

- **Manejo de transacciones.** Una transacción es un programa que se ejecuta como una sola operación. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

- **Tiempo de respuesta.** Lógicamente, es deseable minimizar el tiempo que el SGBD demora en proporcionar la información solicitada y en almacenar los cambios realizados.

1.2.3. Sistemas Gestores de Bases de Datos Relacionales.

Las bases de datos se clasifican entre otros factores de acuerdo a su modelo de administración de datos, entre los que se destacan jerárquico, red, relacional y orientada a objeto. El modelo relacional fue presentado por Edgar Codd en una publicación en 1970, refiriéndose a un modelo de datos en específico, que se caracterizaba por contar con relaciones como único objeto de tratamiento en el modelo, así también definió un álgebra como lenguaje de consulta a la que llamó

Álgebra Relacional (AR), pero no contaba con ninguna manera de expresar actualizaciones, restricciones y/o cálculos sobre el modelo. (Diego 2006)

Este es uno de los más utilizados en la actualidad para modelar problemas reales y administrar datos dinámicamente. Una de las ventajas que posee dicho modelo es que los datos se almacenan, al menos conceptualmente, de un modo que los usuarios entienden con mayor facilidad.

Autores como Hernández y G.W. Hansen han expresado las potencialidades que brinda el modelo antes explicado. Entre otras ventajas, ambos manifiestan que provee herramientas que evitan la duplicidad de registros, garantizan la integridad referencial. De esta manera al eliminar un registro se eliminan todos los registros relacionados dependientes y favorece la normalización por ser más comprensible y aplicable. (Hernández 1997), (G.W. Hansen 1997)

El Sistema Gestor de Base de Datos Relacional (SGBDR) se encarga de administrar, como su nombre lo indica, a las Bases de Datos Relacionales (BDR), esta notoriedad se debe a dos factores fundamentales:

- Ofrece sistemas simples y eficaces para representar y manipular los datos.
- Se basa en el modelo relacional con sólidas bases teóricas.

Estos sistemas son una herramienta efectiva que permiten a varios usuarios acceder simultáneamente a los datos, empleando lenguajes de alto nivel, fáciles de usar. Brindan facilidades eficientes y un grupo de funciones con el objetivo de garantizar la confidencialidad, la calidad, la seguridad y la integridad de los datos que contienen. (Bertino and Martino 1995)

1.3. Herramientas a utilizar.

1.3.1. Embarcadero (ER) Studio.

Embarcadero (ER) Studio es una herramienta de modelado de datos, fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel lógico y físico. Direcciona las necesidades diarias de los administradores de datos, desarrolladores y

arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejas.

Es reconocida en el mercado mundial como la herramienta líder para el modelado de datos, desarrollada por la empresa Embarcadero Technologies. Soporta un gran número de SGBD como: *Oracle, Microsoft SQL Server, MySQL, PostgreSQL*, *Microsoft Access*, entre otros. (Martín 2009)

Seguidamente se muestran sus principales características (Technologies 2009):

Características de producto	Descripción
Ambiente de diseño y modelado altamente productivo	
Gráficos y diseños avanzados.	Automáticamente crea diagramas altamente navegables y legibles.
Capacidades de diseño multinivel.	Permite muchos diseños físicos desde una arquitectura central lógica.
Transformaciones automatizadas y personalizadas.	Facilita la derivación de un diseño físico desde uno lógico y chequea la compatibilidad con una plataforma de base de datos objetivo.
Múltiples formatos de presentación.	Publica modelos y reportes en una variedad de formatos incluyendo HTML, RTF, esquemas XML.
Soporte completo al ciclo de vida de la base de datos	
Ingeniería hacia adelante.	Genera código fuente para algunos diseños de base de datos.
Ingeniería inversa.	Construye modelos gráficos desde una base de datos o esquema existente.
Gestión de modelado empresarial	
Integración de metadatos.	Importa y exporta metadatos desde una variedad de fuentes incluyendo plataformas

	de BI, UML y soluciones de modelado de datos y esquemas XML.
Generación de Esquemas XML.	Asegura que los proyectos XML tales como aquellos utilizando Arquitectura Orientada a Servicio (SOA) están basados en los mismos estándares y metadatos que los modelos al modelarlos en ER/Studio y generar XSD.
Diseño de bases de datos de calidad	
Validación del modelo completo.	Automatiza las revisiones de modelos, y refuerza estándares con más de 50 chequeos para validar modelos lógicos y físicos para definiciones de objetos faltantes, dominios no utilizados, índices únicos e idénticos y relaciones circulares.
Migración automática de llaves foráneas.	Mantiene claves externas para asegurar la integridad referencial en los diseños.
Diseño y seguridad	
Clasificación de datos.	Categoriza y etiqueta datos y objetos de acuerdo al nivel de seguridad y privacidad que deben ser aplicados a la información.
Gestión de permisos.	Permite modelado de usuarios, roles y permisos en niveles lógico y físico.

Tabla 1: Algunas características del producto "Embarcadero ER / Studio".

Embarcadero (ER) Studio en su versión 7.5 es la herramienta que se utilizó para el diseño del modelo lógico y físico de la base de datos del proyecto SITPC.

1.3.2. PostgreSQL.

PostgreSQL es un Sistema Gestor de Base de Datos objeto-relacional, que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Se distribuye

bajo licencia BSD[1] y con su código fuente disponible libremente. Utiliza el modelo Cliente/Servidor así como procesos en vez de múltiples hilos de ejecución. Un fallo en uno de los procesos no afectará al resto y el sistema continuará funcionando. Se desempeña muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo simultáneamente al sistema. (Sánchez 2009)

PostgreSQL ha logrado reunir las siguientes características (González and Díaz 2011)**:**

- Aproxima los datos a un modelo objeto-relacional y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son las consultas SQL declarativas, el control de concurrencia multiversión, el soporte multiusuario, la optimización de consultas así como de herencia y arreglos.
- Altamente Extensible. Permite operadores funcionales, métodos de acceso y tipos de datos definidos por el usuario.
- Incluye características avanzadas tales como las uniones (más conocidas como JOINS por el uso de la terminología en inglés).
- Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- Posee soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL[2].
- Cuenta con habilidad para usar Perl, Python o TCL[3] como lenguaje procedural embebido.

A continuación se mencionan algunos de los programas usados para administrar PostgreSQL (González and Díaz 2011)**:**

- **PgAdmin III:** Herramienta gráfica que permite ver la estructura de las bases de datos, realizar operaciones SQL, ver datos, operaciones de administración.

[1] BSD (Berkeley Software Distribution) o Distribución de Software Berkeley es un sistema operativo derivado del sistema Unix.

[2] PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language) es un lenguaje imperativo provisto por el gestor de base de datos PostgreSQL.

[3] TcL o lenguaje de herramientas de comando se utiliza principalmente para el desarrollo rápido de prototipos, aplicaciones "script", interfaces gráficas y pruebas.

Diseñada para ejecutarse en muchos sistemas operativos como Windows, Linux y MacOS.

- **PhppgAdmin III:** Es una poderosa herramienta de administración basada en una interfaz Web para bases de datos PostgreSQL. Equivalente a la anterior herramienta, pero está realizada en una interfaz web con PHP. Tiene la ventaja de que no requiere la instalación en los clientes, así como se puede centralizar la conexión a las bases de datos, se impide el acceso desde estaciones de trabajo y a la vez, se facilita el trabajo a los potenciales usuarios porque no tienen que dedicar tiempo a configurar conexiones. Dispone de soporte para procedimientos almacenados y vistas.

- **PgAccess:** Es una interfaz gráfica para el gestor de bases de datos PostgreSQL escrito por Constantin Teodorescu en el lenguaje Tcl/Tk. Permite al usuario interactuar con PostgreSQL de una manera similar a muchas aplicaciones de bases de datos, con menús de opciones y diversas herramientas gráficas. Esto significa que el usuario puede evitar la línea de comandos para la mayoría de las tareas. Tiene opciones para creación de formularios e informes.

Por las características anteriormente argumentadas es seleccionado como Sistema Gestor de Base de Datos, PostgreSQL en su versión 9.1 para ser utilizado en el proyecto SITPC y PgAdmin III como herramienta para administrarlo.

Según Charles Fan: "…esta actualización de la base de datos de código abierto líder ofrece tecnología innovadora, extensibilidad sin igual, y nuevas características como replicación sincrónica, método de indexado K-Nearest Neighbor, y conectores de datos foráneos. PostgreSQL 9.1 provee algunas de las más avanzadas características empresariales de cualquier base de datos de código abierto, y es soportado por una entusiasta e innovativa comunidad con probados casos de éxito. Está muy bien preparada para construir y correr aplicaciones en la nube[4]…". (Group 2009)

[4] Las aplicaciones en la nube son aquellas a las que se puede acceder desde el navegador de tal modo que la información y el contenido no se guardar en la estación de trabajo, sino en un servidor remoto en línea.

1.3.3. Apache JMeter.

Herramienta de código abierto, desarrollada con el lenguaje de programación Java en el proyecto Jakarta. JMeter fue diseñada inicialmente para probar aplicaciones Web. Actualmente puede ser utilizado para probar el rendimiento en bases de datos y consultas, servidores FTP, entre otros. Se destaca por su facilidad de su uso, versatilidad, estabilidad y la variedad de funcionalidades que brinda. (Foundation 1999-2013)

Apache JMeter en su versión 2.4 es la herramienta seleccionada para realizar la prueba de Carga y estrés a la base de datos.

1.3.4. EMS Data Generator 2005 para PostgreSQL.

Herramienta poderosa utilizada para generar datos de prueba a tablas de bases de datos para PostgreSQL. Permite definir tablas y campos para generar datos y configurar valores de rangos. Entre sus características se encuentra que provee una interfaz amigable, permite la generación de datos a varias tablas desde diferentes bases de datos en un solo ordenador anfitrión (host). Soporta todos los tipos de datos de PostgreSQL, incluyendo los tipos array, network address (dirección de red) y geometric (geométricos). (Solutions 1999 - 2013)

EMS Data Generator 2005 para PostgreSQL es la herramienta seleccionada para realizar la prueba de Volumen a la base de datos.

1.4. Lenguaje de programación.

1.4.1. PL/pgSQL.

Procedural Language/PostgreSQL Structured Query Language es un lenguaje provisto para el gestor de base de datos PostgreSQL. Permite ejecutar comandos SQL mediante un lenguaje de sentencias imperativas y uso de funciones, dando mucho más control automático que las sentencias SQL básicas. Puede ser usado para crear funciones y procedimientos almacenados. Añade estructuras de control al lenguaje SQL. Permite realizar cálculos complejos y crear nuevos tipos de datos de usuario.

Ventajas de usar PL/pgSQL:

- *Mayor rendimiento:* permite agrupar un conjunto de consultas dentro del servidor de bases de datos, teniendo así la potencia de un lenguaje procedural y la sencillez de uso del SQL, pero ahorrando una gran cantidad de tiempo porque no tiene la sobrecarga de una comunicación cliente/servidor.
- *Soporte SQL:* añade a la potencia de un lenguaje procedural la flexibilidad y sencillez del SQL. Puede usar todos los tipos de datos, columnas, operadores y funciones de SQL.
- *Portabilidad:* Debido a que las funciones PL/pgSQL corren dentro de PostgreSQL, estas funciones trabajarán en cualquier plataforma donde PostgreSQL corra. Así podrá reusar el código y reducir costes de desarrollo.**Optimización de bases de datos relacionales.**

En el desarrollo de una base de datos, la optimización, juega un papel primordial en lo referente al incremento del rendimiento de la misma. Esta depende en gran medida de un correcto diseño en la fase inicial, tanto lógico como físico. Tiene entre sus objetivos minimizar la redundancia de los datos e inconsistencia en los mismos, reducir el espacio de almacenamiento, minimizar los tiempos de respuesta para cada petición y disminuir el consumo de recursos.

Para lograr un proceso exitoso de rendimiento es preciso tener presente los criterios que involucra el proceso de optimización durante todo el ciclo de desarrollo, y no solo hacerlo una vez que el sistema ya está desarrollado. Teniendo como referencia el momento en que se comienza a realizar, el afinamiento se puede clasificar en:

- **Afinamiento Proactivo:** es cuando el afinamiento se realiza mientras se desarrolla el sistema.
- **Afinamiento Reactivo:** es el que se realiza para mejorar sistemas en explotación.

De los métodos anteriormente presentados el más efectivo, en gran medida, es el Afinamiento Proactivo. Este requiere para su desarrollo que en la etapa inicial del ciclo de vida del software los analistas y clientes sean capaces de establecer metas reales de rendimiento. Por consiguiente, en la etapa de análisis y diseño los arquitectos y diseñadores del sistema y de base de datos, puedan determinar qué criterios aplicar

para incrementar el rendimiento. Además establecerán qué configuraciones de hardware serán necesarias para cumplir con estas dichas metas.

A pesar de que el Afinamiento Proactivo es más eficiente y menos costoso que el Afinamiento Reactivo, esto no quiere decir que sea posible prescindir de este último. Esto es posible dada la capacidad que posee el mismo de identificar y evitar riesgos a los que está expuesto un sistema bien diseñado. Es posible mejorar un sistema en explotación reactivamente comenzando por el paso final e ir subiendo poco a poco a los otros, encontrando y arreglando la mayor cantidad posible de cuellos de botella[5].

Una de las cosas que no debería ocurrir es que se trate todo lo relacionado a la optimización luego de que los usuarios planteen quejas e inconformidades sobre el rendimiento y los tiempos de respuesta del sistema. Sino que deben trazarse estrategias en función de mitigar esta problemática durante todo el proceso de desarrollo.

En la actualidad existen diversas estrategias para optimizar bases de datos relacionales. Sin embargo, aún no se encuentra en la bibliografía consultada, una guía completa para realizar dicho proceso. Seguidamente se proponen un conjunto de criterios a tener en cuenta en el proceso antes mencionado:

- Empleo de patrones de diseño de bases de datos.
- Normalización de la base de datos.
- Índices.
- Optimización de consultas SQL.
- Optimización en PostgreSQL.

[5] Cuellos de botella en un proceso proactivo, es una fase de la cadena de producción más lenta que otras, que ralentiza el proceso de producción global.

1.6. Criterios de optimización.

En el presente epígrafe se realizará una explicación detallada de los elementos mencionados anteriormente para llevar a cabo el proceso de optimización desde la fase del diseño de la base de datos:

1.6.1. Empleo de patrones de diseño de bases de datos.

En la elaboración de una base de datos lo primero que se debe tener presente es realizar un adecuado diseño, de esto depende la eficacia del cumplimiento de los objetivos definidos para la base de datos, razón por la cual es lógico y necesario el empleo del tiempo que sea requerido para aprender los principios y aspectos fundamentales de un correcto diseño.

I. Fases en el diseño de una Base de Datos Relacional.

El proceso de diseño de bases de datos relacionales está compuesto por cuatro fases, las cuales son: (Andrés 2001)

1. Diseño conceptual de la base de datos.
2. Elección de un SGBD.
3. Diseño lógico.
4. Diseño físico.

Fase 1: Diseño conceptual de la base de datos.

La primera fase, produce una representación abstracta y de alto nivel de realidad creando un esquema conceptual, independiente del SGBD, presenta una gran capacidad semántica, además parte de especificaciones de requerimientos que describan la realidad. El objetivo fundamental del diseño conceptual es la compleción y expresividad de los esquemas conceptuales locales. (Navate 2008)

Fase 2: Elección de un SGBD.

La elección de un SGBD se debe realizar teniendo en cuenta un grupo de factores que inciden sobre la misma, como lo son: posibilidades económicas y tecnológicas de los clientes, experiencias de los mismos, entre otros.

Fase 3: Diseño lógico.

En esta fase se convierte la representación realizada en la primera fase en especificaciones que pueden implantarse en un sistema de cómputo y ser procesadas por él. El objetivo de este diseño es obtener una representación que use, del modo más eficiente posible, los recursos que el modelo de SGBD posee para estructurar los datos y para modelar las restricciones. (Navate 2008)

Fase 4: Diseño físico.

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: (Cabrera 2010)

- Obtener un conjunto de tablas, relaciones entre ellas 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.

II. Patrones de diseño de base de datos.

En el proceso de diseñar y modelar bases de datos es habitual que se encuentren elementos duplicados en diversos modelos, los que correctamente identificados pasan a ser patrones de diseño.

Los patrones de diseños ayudan a conseguir diseños optimizados de manera ágil y con pocos esfuerzos al proponer buenas prácticas y soluciones pensadas y testeadas. Estos aportan ideas reutilizables y adaptables a un amplio campo de problemas. En la actualidad se cuenta con algunos patrones de diseño de base de datos que son de

mucha ayuda a los diseñadores de base de datos para realizar el diseño de una forma óptima. (Blaha 2010)

En esencia, los patrones de diseño de bases de datos constituyen la base para la búsqueda de soluciones a problemas comunes en el proceso de diseño de las mismas. Seguidamente se explican algunos de estos (Blaha 2010):

a) Patrones de Árboles.

Árboles simples: Este patrón es utilizado cuando el árbol es la representación de una estructura de datos. Posee como limitación que los elementos a almacenar tienen que ser del mismo tipo, es decir pueden ser almacenados en la misma entidad. Ejemplo un nombre único global para cada nodo (Ver anexo 1).

b) Patrones para flujos de trabajo.

Máquina de estado para un tipo de entidad: Este patrón se aplica para representar cuales son los estado por los que puede atravesar un tipo de entidad. No refleja el almacenamiento de las ocurrencias sino las propiedades del flujo de trabajo, por lo tanto esto es el diseño del diagrama de flujo (Ver anexo 2).

c) Patrón de llaves subrogadas.

Este patrón consiste en asignar una llave o identificador único para cada entidad, el cual usualmente no tiene ninguna relación con el negocio. Por lo general esta llave es de tipo numérico, preferiblemente entero (auto-incremental), o GUID[6]. Está demostrado que de este último no se repiten claves únicas puesto que su número total es tan grande que se considera que la probabilidad de que se generen dos veces un mismo número puede ser extremadamente baja o nula en la práctica. Es muy utilizado en la actualidad pues goza de una amplia aplicación y aceptación por parte de los diseñadores de bases de datos. A continuación se citan algunas de las ventajas y desventajas de este (González and Díaz 2011):

Ventajas:

[6] GUID: Globally Unique Identifier o Identificador Globalmente Único.

- La lógica de negocio no está en las claves. Protege ante cambios.
- Son pequeñas. Enteros que ocupan muy poco. JOINS más rápidos.
- No hay contención pues los mecanismos de generación secuencial son rapidísimos y los provee el sistema.
- Uniformidad. Se pueden programar tareas de mantenimiento sobre tablas que asumen un esquema de llave primaria común.

Desventajas:

- Siempre se requiere de JOINS para buscar en las tablas hijas/relacionadas.
- Normalización. Se requiere un índice único adicional sobre la clave candidata (natural) para evitar la existencia de llaves candidatas duplicadas en el dominio.

1.6.2. Normalización de la base de datos.

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. (García 2005).

Su propósito es mejorar la integridad de los datos a través de la minimización de la redundancia y la inconsistencia en ellos. Entre las ventajas que proporciona este proceso se encuentran que evita anomalías en inserciones, actualizaciones y eliminaciones en los datos, mejora la independencia en estos y protege la integridad de los mismos. Otra ventaja de gran importancia en relación al consumo de espacio de almacenamiento, ya que una base de datos normalizada ocupa menos espacio en disco que una que no lo está.

Esta involucra una serie de pasos o fases consecutivos, donde cada uno corresponde a una Forma Normal (FN) en la que se establecen determinados parámetros a cumplir. Conforme se va avanzando en la normalización, las relaciones tienen un formato más estricto y, por lo tanto, son menos vulnerables a las anomalías. En la tabla que se muestra a continuación se evidencian cada una de estas FN y sus creadores.

Forma normal	Definida por
Primera Forma Normal (1NF)	Existen dos versiones, la de E. F. Codd de 1970 y la de C. J. Date de 2003.
Segunda Forma Normal (2NF)	E. F. Codd en 1971.
Tercera Forma Normal (3FN)	E. F. Codd en 1971.
Forma Normal de Boyce-Codd (BCNF)	Raymond F. Boyce y E. F. Codd en 1974.
Cuarta Forma Normal (4NF)	Ronald Fagin en 1977.
Quinta Forma Normal (5NF)	Ronald Fagin en 1979.
Forma Normal de Dominio/Clave (DKNF)	Ronald Fagin en 1981.
Sexta Forma Normal (6NF)	C. J. Date, Hugh Darwen y Nikos Lorentzos en 2002.

1. Primera forma normal (1FN).

Una relación está en la primera forma normal si, y solo si, satisface la limitación de que contenga solamente valores atómicos. En otras palabras, una relación está en primera forma normal si, y solo si, no incluye ningún grupo repetitivo (un grupo repetitivo es un atributo que contiene un conjunto de valores y no un único valor). (García 2005)

2. Segunda forma normal (2FN).

Una relación está en 2FN si, y solo si, está en 1FN y cada atributo no llave es totalmente dependiente de la llave primaria. En otras palabras, una relación R se dice que está en 2FN si, y solo si, está en 1FN y los atributos no llaves (ni primarias, ni candidatas) de R son funcional y completamente dependientes de la llave primaria de R. (García 2005)

Las únicas relaciones que pueden violar la 2FN son las que tienen llaves compuestas pues no es posible que en una relación cuya llave primaria sea simple (compuesta por

un solo atributo) haya atributos que dependan de parte de la llave primaria. Una relación que esté en 1FN y que tenga una llave primaria simple ya está en 2FN. (García 2005)

3. Tercera forma normal (3FN).

Una relación R está en 3FN si, y solo si, está en 2FN y los atributos no llaves son independientes de cualquier otro atributo no llave primaria. En otras palabras, para que una relación esté en 3FN se deben eliminar las dependencias transitivas de atributos no llaves respecto a la llave primaria; estando ya la relación en 2FN. (García 2005)

Para aclarar el concepto de dependencia transitiva es necesario decir que: sean A, B y C conjuntos de atributos de una relación R. Si B es dependiente funcionalmente de A y C lo es de B, entonces C depende transitivamente de A. (García 2005)

4. Forma normal Boyce Codd (FNBC).

Una relación R está en FNBC si y solo si cada determinante[7] de la relación es una súper llave de R. Para que una relación no esté en FNBC se deben cumplir la condición necesaria más no suficiente: (Cabrera 2010)

La relación tiene llaves candidatas múltiples, estas llaves candidatas son compuestas y las llaves candidatas se solapan (tienen al menos un atributo en común).

5. Cuarta forma normal (4FN).

La 4FN se asegura de que las dependencias multivaluadas independientes estén correcta y eficientemente representadas en un diseño de base de datos. La 4FN es el siguiente nivel de normalización después de la forma normal de Boyce Codd. (Valero and Pérez 2011)

6. Quinta forma normal (5FN).

La quinta forma normal, también conocida como forma normal de proyección-unión, es un nivel de normalización de bases de datos designado para reducir redundancia en

[7] Es cualquier atributo cuyo valor determina otros valores de atributos.

las bases de datos relacionales que guardan hechos multi-valores aislando semánticamente relaciones múltiples. Una tabla se dice que está en 5FN si y sólo si está en 4FN y cada dependencia de unión (JOIN) en ella es implicada por las claves candidatas. (Valero and Pérez 2011)

1.6.2.1. Desnormalización.

Para los que se guían estrictamente por las reglas de las formas normales, 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. (Marqués 2009)

No se puede establecer una serie de reglas que determinen cuando desnormalizar tablas, pero hay algunas situaciones bastante comunes en donde puede considerarse esta posibilidad, por ejemplo: (Marqués 2009)

1. *Combinar relaciones de uno a uno:* esto puede ser conveniente cuando hay tablas involucradas en relaciones de uno a uno, se accede a ellas de manera conjunta con frecuencia y casi no se accede a ellas por separado.

2. *Duplicar atributos no clave en relaciones de uno a muchos:* para evitar operaciones de unión (JOIN) entre tablas, se puede incluir atributos de la tabla madre en la tabla hija de las relaciones de uno a muchos.

3. *Duplicar atributos en relaciones de muchos a muchos:* durante el diseño lógico se crea una nueva tabla para almacenar las ocurrencias de una relación de muchos a muchos, de modo que si se quiere obtener la información de la relación de muchos a muchos, se tiene que realizar el JOIN de tres tablas. Para evitar algunos de estos JOIN se puede incluir algunos de los atributos de las tablas originales en la tabla intermedia.

4. *Partir tablas:* las tablas se pueden partir horizontalmente (por datos) o verticalmente (por atributos) de modo que a partir de una tabla grande, que tiene datos que no se acceden con frecuencia, se obtengan tablas más pequeñas. Algunas de estas contienen solamente datos a los que sí se accede muy a menudo.

1.6.3. Índices.

El hecho de que una consulta funcione correctamente no quiere decir que lo haga de la forma más rápida. Existen dos maneras básicamente para acelerar las consultas: una es afinando el servidor para que responda lo mejor posible mientras que la otra es haciendo uso de los índices de forma correcta.

En la actualidad una de las prioridades de una aplicación informática es que tengan un mayor rendimiento, prioridad que se logra entre otras cosas mediante la disminución del tiempo de respuesta de la base de datos a dicha aplicación. El empleo correcto de estructuras auxiliares denominadas índices forma parte de los elementos utilizados para lograr cumplir tales prioridades.

En el mundo de las bases de datos los índices funcionan similares al índice de un libro, mediante el cual se puede acceder rápidamente a la información de interés. Por tanto, los índices son una estructura de datos utilizada para mejorar e incrementar la velocidad de búsqueda de valores en una base de datos. Se actualizan automáticamente cuando se realizan operaciones de escrituras en una tabla.

Del modelo relacional se puede obtener la indexación, tanto de claves primarias como foráneas, donde las llaves primarias identifican unívocamente a cada elemento de una tabla y las llaves foráneas establecen las relaciones entre tablas.

A continuación se ofrecen un conjunto de buenas prácticas para realizar un uso adecuado de los índices (González and Díaz 2011):

- *Crear un índice sobre los campos que son utilizados en las búsquedas* (los que aparecen en las cláusulas WHERE o JOIN) para mejorar una consulta SQL de tipo SELECT.
- *Utilizar índices sobre campos con valores únicos.* Los índices funcionan peor si el campo tiene valores duplicados.
- *Tratar de que los índices sean cortos.* Si indexa un campo de texto, evite hacerlo sobre campos de longitud variable, y acorte siempre el tamaño del índice a lo que considere más adecuado.

- *Evitar la creación de índices innecesarios.* Estos se actualizan con cada cambio en la tabla asociada y pueden ralentizar las modificaciones de la misma.
- *Evitar el uso de los índices en las tablas en las que se utilizan frecuentemente operaciones de escritura* como: Insert, Delete, Update, puesto que los índices se actualizan cada vez que se modifica una columna.
- *Evitar el uso de los índices en las tablas demasiado pequeñas* porque no se necesita ganar tiempo en las consultas.

PostgreSQL ofrece otros tipos de índices como son:

- ***Btree (árbol-B):*** Este tipo de índice es estándar, donde la letra B es sinónimo de equilibrio, por lo que este árbol está equilibrado, lo que significa que tiene la misma cantidad de datos tanto en el lado izquierdo como en el derecho. Su estructura tiene la forma de un árbol invertido donde las estructuras superiores se llaman ramas y la estructura inferior constituye las hojas. Habitualmente este tipo de índice tiene uno o más niveles de ramas, las cuales contienen la columna índice (la clave) y la dirección de otro bloque, mientras que los nodos hoja contienen la clave de cada fila de la tabla.

  Puede gestionar consultas usando operadores de igualdad o de rango de datos (<, <=, =, >, =>). Además, se puede utilizar 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 antes mencionados. También se utiliza para evitar las grandes operaciones de ordenación. Se obtiene el mejor resultado cuando se aplica sobre columnas con una alta cardinalidad, es decir, sobre columnas que tengan muchos valores diferentes. Actualmente, sólo el acceso de este tipo brinda soporte para índices multi-columna.

- ***Hash:*** 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. Son más útiles cuando se almacenan grandes cantidades de información. Estas almacenan la información en posiciones pseudo-aleatorias, por lo que el acceso ordenado a su contenido es bastante lento. El hash sólo puede ser usado para consultas de equivalencia, o sea, para consultas con el operador: =. No

soporta el comando IS NULL, a diferencia del B-tree que si lo permite. (Castro 2010)

- ***GIST:*** No son solo tipo de índice, sino una infraestructura dentro de la cual muchas de las estrategias de indexación diferentes pueden ser implementadas. GIST es el acrónimo de Generalized Search Tree, (árbol de búsqueda generalizada). Los operadores en que pueden ser utilizados varían dependiendo de la estrategia de indexación (clase de operador). Soporta consultas usando: <<, &<, &>, >>, <<|, &<|, |&>, |>>, @>, <@, ~=, &&. Btree, Rtree y muchos otros esquemas de indexación pueden ser implementados en GIST. Una de las ventajas de su uso es que permite el desarrollo de tipos de datos personalizados con los métodos de acceso adecuados, por un experto en el dominio del tipo de datos, en lugar de un experto en bases de datos.

Los índices GiST pueden ser utilizados como "null safe", significa que pueden indexar columnas que incluyen valores nulos. Soportan el concepto de "lossiness", el cual es muy importante cuando se manejan grandes objetos GIS que el tamaño de página de PostgreSQL. (Smith 2009)

- ***GIN:*** Es el acrónimo de Generalized Inverted Index (Índice Invertido Generalizado). Los GIN son índices invertidos que pueden controlar los valores que contienen más de una clave, por ejemplo las matrices. Al igual que con GiST, puede soportar muchas estrategias de indexación diferentes definidas por el usuario. Los operadores particulares con los que se puede utilizar un índice GIN varían dependiendo de la estrategia de indexación. Por ejemplo, la distribución estándar de PostgreSQL incluye clases de operador GIN para matrices unidimensionales, que admiten consultas indexadas utilizando estos operadores: <@, @>, =, &&. Estos son operadores para datos tipo array y para "Full Text Searching" dentro de documentos a través de lexemas. (Smith 2009)

Un uso inapropiado de los índices traerá consigo que se ralentice el funcionamiento de las consultas, ya que los tiempos de actualización e inserción se ven incrementados ante la presencia de índices. Además estos consumen espacio en el disco. Una vez creado un índice, el sistema tiene que mantenerlo sincronizado con la tabla. Esto implica una sobrecarga de datos y operaciones de manipulación, por

lo que los índices que rara vez o nunca se utilizan en las consultas deben ser eliminados.

El uso de los índices es de vital importancia para aquellas bases de datos de grandes volúmenes. Cuando en una tabla no existen índices, el sistema gestor tendrá que leer todos los registros de la tabla para poder resolver la consulta. A este proceso es lo que se le denomina comúnmente “escaneo completo de una tabla”. Indiscutiblemente son muchos los problemas que se evitan al hacer uso correcto de los índices, a continuación se ofrecen algunos de estos:

- **Sobrecarga de CPU:** La verificación de cada uno de los registros en una tabla tiene poca relevancia cuando se tienen pocos datos, no siendo así cuando aumenta la cantidad de registros en la tabla. Este aspecto provoca que se realice una sobrecarga en el CPU pues el número de registros y el tiempo que emplea el gestor en revisar totalmente una tabla es directamente proporcional.

- **Sobrecarga de disco:** El escaneo completo a una entidad considerablemente grande, trae como consecuencia el consumo de una gran cantidad de entrada/salida en el disco. De ahí proviene que el servidor de base de datos pueda calentarse significativamente, sobre todo si el disco utilizado es un IDE antiguo.

- **Concurrencia:** Mientras un sistema gestor está leyendo los datos de una tabla, éste la bloquea, de tal manera que nadie más puede escribir en ella, aunque si pueden leerla. Ocurre de forma similar cuando se está actualizando o eliminando filas de una tabla, con la diferencia que aquí no se puede leer. (González and Díaz 2011)

1.6.4. Optimización de consultas SQL.

El proceso de optimización de consultas se define como el conjunto de técnicas, algoritmos y reglas que le permiten, al motor de base de datos, elegir una alternativa entre varias, con la cual pueda obtener los datos de una manera óptima.

Este proceso tiene que ser lo bastante rápido como para que sea conveniente para el motor efectuar este cálculo sin afectar el rendimiento en la construcción del resultado de la consulta. A continuación se muestran los pasos en los que se descompone el procesamiento de una consulta (Perdomo and Fernández 2009):

- *Transmisión del texto de la consulta al backend de PostgreSQL:* En este paso no hay mucho que se pueda mejorar, aunque si se tienen consultas demasiado largas embebidas en el código de la aplicación y que no pueden ser preparadas en el servidor con antelación, lo mejor es guardarlas en la base de datos como un procedimiento almacenado para minimizar el tiempo de transmisión.
- *Parseo del texto:* Una vez que el texto de la consulta llega al gestor este la convierte en tokens. Escribir las consultas como procedimientos de almacenado, también minimiza el tiempo invertido en este paso.
- *Planeación de la consulta:* Usualmente este es el paso en que el SGBD decide cómo se ejecutará la consulta. Más adelante se explica cómo trabaja el Query Planner o Planificador de consultas.
- *Ejecución de la consulta:* La velocidad con la que se ejecuta una consulta depende del camino que el planificador de consultas haya decidido para ejecutar la misma, de la configuración de hardware que se disponga y de cómo se haya configurado el servidor de PostgreSQL.
- *Transmisión del resultado:* Aunque este paso no se pueda optimizar mucho hay que tener en cuenta que todos los datos que se transmiten son extraídos desde discos, por lo que en las consultas se debe tratar de evitar solicitar datos innecesarios.

Query Planner:

El planificador de consultas de PostgreSQL es el encargado de decidir cómo ejecutar una consulta determinada y realizar dicha ejecución de la manera más eficiente. Esto sucede debido a que SQL es un lenguaje declarativo donde el programador especifica qué va a devolver la consulta y no cómo hacerlo.

Un plan de consulta se representa en forma de árbol, donde cada nodo es una operación que tiene que realizar el gestor para ejecutar una consulta específica, dígase: JOINS, ordenamiento, búsqueda en disco, etc. Las tuplas que serán mostradas en el resultado de la consulta empiezan a subir desde las hojas del árbol (que casi siempre son las búsquedas en disco) hasta la raíz. Para obtener un resultado, el ejecutor de la consulta solo tiene que pedirle al nodo raíz un resultado y este les pasa la petición a sus hijos y así sucesivamente, siempre y cuando el plan escogido sea bueno, la ejecución de la consulta presentará un buen rendimiento. (Perdomo and Fernández 2009)

Existen parámetros que pueden ser alterados para obligar al planificador de consultas a escoger planes alternativos para determinadas consultas, algunos de estos son (Perdomo and Fernández 2009):

- ***enable_bitmapscan(boolean):*** Este parámetro habilita o deshabilita el uso de planes que utilicen el escaneo de mapas de bit. Por defecto su valor es on.
- ***enable_hashagg(boolean):*** Este parámetro habilita o deshabilita el uso de planes que utilicen agregaciones hash. Por defecto su valor es on.
- ***enable_hashjoin(boolean):*** Este parámetro habilita o deshabilita el uso de planes que utilicen hash-join. Por defecto su valor es on.
- ***enable_indexscan(boolean):*** Este parámetro habilita o deshabilita el uso de planes que utilicen el escaneo de índices. Por defecto su valor es on.
- ***enable_mergejoin(boolean):*** Este parámetro habilita o deshabilita el uso de planes que utilicen merge joins. Por defecto su valor es on.
- ***enable_nestloop(boolean):*** Este parámetro habilita o deshabilita el uso de planes que utilicen ciclos anidados. Por defecto su valor es on. Aunque este valor puede ser cambiado es imposible suprimir completamente el uso los ciclos anidados, ya que existen situaciones donde es imposible evitar su uso, al situar este parámetro en off, simplemente se evita que el planeador de ejecución de consultas de PostgreSQL utilice ciclos anidados para ejecutar una consulta si existe otra vía.

- ***enable_seqscan(boolean):*** Este parámetro habilita o deshabilita el uso de planes que utilicen búsquedas secuenciales. Por defecto su valor es on. Aunque este valor puede ser cambiado es imposible suprimir completamente el uso las búsquedas secuenciales, ya que existen situaciones donde es imposible evitar su uso, al situar este parámetro en off, simplemente se evita que el planeador de ejecución de consultas de PostgreSQL utilice búsquedas secuenciales para ejecutar una consulta si existe otra vía.

- ***enable_sort(boolean):*** Este parámetro habilita o deshabilita el uso de planes que utilicen ordenamientos. Por defecto su valor es on. Aunque este valor puede ser cambiado es imposible suprimir completamente el uso los ordenamientos ya que existen situaciones donde es imposible evitar su uso, al situar este parámetro en off, simplemente se evita que el planeador de ejecución de consultas de PostgreSQL utilice ordenamientos para ejecutar una consulta si existe otra vía.

Es preciso tener presente que la modificación de estas cuantificaciones puede ser una solución para forzar al planificador de consultas de PostgreSQL a modificar su plan de ejecución. Sin embargo, no se aconseja que se modifiquen los valores que traen las mismas por defecto para alguna consulta en particular porque es posible que esa acción cause más problemas que beneficios.

1.6.5. Optimización en PostgreSQL.

Uno de los pasos de vital importancia que se realizan para la optimización actualmente es modificar algunos de los parámetros de configuración en el SGBD PostgreSQL a conveniencia de los administradores y condicionado por los requisitos del sistema. Es preciso tener en cuenta que todos los parámetros tienen un valor por defecto sin embargo, esta configuración está diseñada más bien para lograr compatibilidad con casi cualquier configuración de hardware en la cual pueda ser instalado y no para alcanzar un buen rendimiento.

Uno de los objetivos principales en este aspecto es lograr que PostgreSQL se ejecute de una manera más rápida en la plataforma en la que se encuentre instalado.

Propósito que es posible alcanzar ya sea afinando los parámetros de rendimiento o añadiendo más y/o mejor hardware. Postgresql.conf es el fichero de configuración de dicho gestor en el que se optimizan parámetros como la memoria caché, la memoria compartida y la paginación. A continuación se muestran una breve descripción de algunos de los parámetros que provee este fichero (Perdomo and Fernández 2009):

- ***max_connections:*** Este parámetro define exactamente lo que su nombre indica, establecer el número máximo de conexiones clientes permitidas en el servidor. Este valor es muy importante para algunos de los otros parámetros que se explicarán a continuación, particularmente work_mem, porque hay algunos recursos de memoria que son asignados por conexión, por lo que el máximo número de clientes que pueden estar conectados es un aspecto a tener en cuenta a la hora de asignar dichos recursos. Generalmente, PostgreSQL con un buen hardware puede soportar algunos cientos de conexiones sin afectar el rendimiento. Pero si se necesitan miles de conexiones se debe usar connection pooling software, que en español se puede traducir este término como: software de agrupación de conexiones, para reducir la sobrecarga en el servidor por exceso de peticiones.

- ***shared_buffers:*** Este parámetro es importantísimo y define el tamaño del buffer de memoria utilizado por PostgreSQL. El hecho de aumentar mucho este valor no significa que se tendrá mejor respuesta. En un servidor se puede empezar con un 25% del total de la memoria RAM[8]. Nunca más de 1/3 (33%) del total.

- ***maintenance_work_mem:*** Usada en operaciones del tipo VACUUM, ANALYZE, CREATE INDEX, ALTER TABLE, ADD FOREIGN KEY. Su valor dependerá mucho del tamaño de las bases de datos. Por ejemplo, en un servidor con 4Gbytes de memoria, se puede usar 256 Megabyte (MB) como valor inicial.

[8] RAM: Memoria de Acceso Aleatorio (en inglés Random Access Memory).

- ***checkpoint_segments:*** Este parámetro es muy importante cuando se realizan con frecuencia numerosas operaciones de escritura (INSERT, UPDATE, DELETE), ya que PostgreSQL escribe las nuevas transacciones a la base de datos en archivos llamados segmentos del WAL que son de 16 MB de tamaño. Es aconsejable que se incremente dicho parámetro en sistemas con una alta concurrencia de actualizaciones de los datos.

- ***checkpoint_timeout****:* Si bien el parámetro **checkpoint_segments** servía para indicar la cantidad de memoria del WAL que provoca un punto de chequeo, éste parámetro lo establece en cuanto al tiempo. Su valor por defecto es 5 minutos.

- ***effective_cache_size:*** Este valor debe ser fijado basándose en un estimado de cuánta memoria caché queda disponible en el sistema operativo. Esta estimación se realiza después de tener en cuenta la cantidad que el propio SO utiliza, la que será destinada a usar por el gestor PostgreSQL y la del resto de las aplicaciones que puedan estar corriendo en el sistema.

- ***sort_memory:*** Este parámetro establece la máxima cantidad de memoria que puede usar una conexión a la base de datos para realizar procesos de ordenamiento. Si una consulta tiene sentencias ORDER BY o GROUP BY y requiere ordenar una gran cantidad de datos, incrementar este parámetro ayuda a mejorar el rendimiento. Sin embargo, hay que tener cuidado porque dicho parámetro es para cada ordenamiento de cada conexión, por lo que no se debe aumentar mucho su valor, sobre todo en bases de datos con muchos usuarios.

- **fsync:** Este parámetro establece si se va a escribir o no en disco tan pronto como se haga la transacción, lo cual es hecho a través del Write Ahead Logging (WAL). Si se tiene suficiente confianza en el hardware, en el abastecimiento de energía eléctrica y en la batería de respaldo, entonces se puede establecer este parámetro en ON y se logra aumentar la velocidad de escritura en disco. Sin embargo, esto significa que si sucede cualquier imprevisto y se apaga el servidor, es muy probable que se tenga que restaurar

la base de datos con el último respaldo que se le haya hecho, pues se perdería un gran volumen de datos.

- ***vacumm_mem:*** Este parámetro establece la cantidad de memoria que puede utilizar el proceso de VACUUM. Normalmente VACUUM es un proceso intensivo de lectura/escritura de disco, pero aumentando este parámetro aumentará la velocidad del proceso mencionado porque le permite a PostgreSQL copiar mayores bloques a la memoria. Aunque no se debe poner muy grande tampoco, porque es memoria que se estaría restando a los procesos normales de la base de datos y por tanto afectaría el rendimiento de la misma. Para la mayoría de los sistemas un valor entre 16 MB y 32 MB debe ser suficiente.
- ***max_fsm_relations:*** Este parámetro indica la máxima cantidad de relaciones (tablas) que serán vigiladas por el FSM, algo que es tan sencillo como conocer la cantidad de tablas con las que cuenta el servidor.

Conclusiones Parciales

Con el estudio de temas relacionados con bases de datos relacionales y criterios de optimización empleados para incrementar el rendimiento de las mismas, se obtuvieron los conocimientos teóricos necesarios para continuar con la investigación lo que posibilitará desarrollar la propuesta de solución. Mediante la realización de dicho estudio se cumplió el primer objetivo trazado en el presente trabajo de diploma.

Capítulo 2: Propuesta de solución.

2.1. Introducción.

En el presente capítulo se realizará básicamente la propuesta de solución partiendo del estudio realizado en el capítulo anterior. Es preciso tener en cuenta que la propuesta formulada será aplicada a toda la base de datos del proyecto y que por las características actuales del mismo, se tomarán como muestras de aplicabilidad solamente los subsistemas Penal y Común. Estos subsistemas se caracterizan por ser los de mayor complejidad, desde el punto de vista del negocio. La complejidad del subsistema Penal consiste en que dicha materia se encarga de dar solución a los conflictos que se originan por la comisión de delitos. Además de realizar todos los actos procesales, formas y procedimientos que se hacen en el Tribunal para dictar sentencia y ejecutarla conforme dispone la Ley de Procedimiento Penal[9]. Esta comprende el Expediente en Fase Preparatoria que contiene todos los acusados que poseen las imputaciones de los delitos, sus clasificaciones y las sanciones así como las sanciones accesorias. Además, las situaciones legales de los acusados, el control de vencimientos de los términos y la tramitación del expediente, teniendo informada a cada una de las partes en todo momento. En cambio la complejidad del subsistema Común radica en que contiene todas las estructuras de datos comunes de los subsistemas del SITPC, dígase la estructura genérica de las leyes, lo referente a los expedientes de los acusados. También, las personas tanto naturales como jurídicas involucradas en los procesos judiciales, las estructuras organizacionales (como la Fiscalía, bufetes, PNR), entre otros. Además contiene los nomencladores y las funcionalidades comunes. Dentro de estas últimas se encuentran los servicios de turnado de los jueces a un expedientes y el visor de documentos.

Por lo expresado anteriormente hace de estos dos subsistemas, una muestra representativa y fundamental para llevar a cabo la posterior implantación de la propuesta a la totalidad de la base de datos.

Con el desarrollo de este capítulo se describe el impacto y la repercusión que pueden traer los cambios planteados en el desarrollo del mismo así como un conjunto de

[9] Ley No. 5 de Procedimiento Penal, de fecha 13 de agosto de 1977.

procesos que se deben de tener en cuenta para la optimización de la base de datos. Dentro de estos se encuentra el diseño de los esquemas correspondientes a los subsistemas mencionados anteriormente y la normalización de los mismos. Conjuntamente se expondrán algunos de los índices que se agregaron a la base de datos. Se plantearán los patrones de diseño de base de datos a emplear en la propuesta y algunas de las pautas a seguir para programar consultas SQL con el objetivo de incrementar el rendimiento de manera general.

2.2. Descripción de la base de datos del proyecto SITPC.

La base de datos relacional del proyecto estaba compuesta por siete esquemas conceptuales independientes, los cuales se correspondían a cada materia y a los subsistemas AG y Común. En AG se guardaban los datos relacionados con la administración del sistema, dígase roles, permisos, usuarios, entre otros. Estos estaban estrechamente relacionados con el componente ACAXIA del marco de trabajo SAUXE y todos los datos concernientes a la configuración de las estructuras necesarias para operar dentro de un tribunal. Mientras que en Común se almacenaban las funcionalidades comunes mencionadas anteriormente. En la gráfica, que se expone a continuación, se evidencian las estadísticas generales de cómo estaba conformada la base de datos antes de dar por fracasado el proyecto en Septiembre de 2012 y como se encuentra una vez aplicada la propuesta de solución a los esquemas Penal y Común en Abril de 2013:

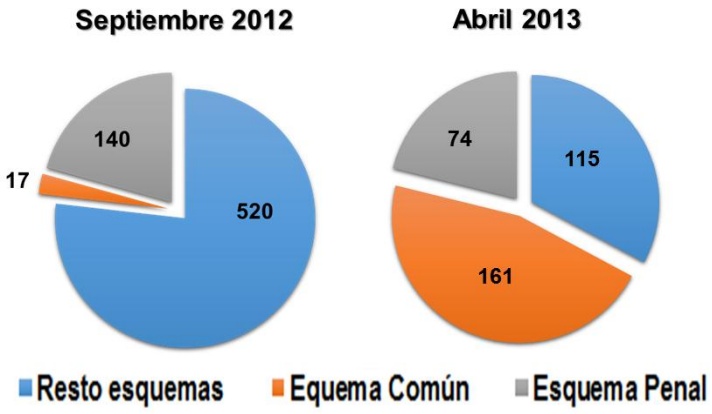


Figura 1: Estadísticas generales de la base de datos del proyecto SITPC.

Una vez realizado el análisis del mismo se infiere que la base de datos relacional del proyecto estaba compuesta por 677 entidades. Es preciso aclara que el proceso de modelación de la base de datos no se había culminado y se estima que una vez concluido dicho proceso el total de entidades de la misma ascendería aproximadamente a 800. Los esquemas Común y Penal contenían 17 y 140 entidades respectivamente.

Partiendo de un estudio preliminar que se realizó en el proyecto se propone la siguiente solución aplicando correctamente los criterios de optimización explicado con anterioridad. La base de datos está estructurada por siete esquemas conceptuales correspondientes a cada una de las materias, así como los esquemas Seguridad y Común. Este último constituye el esquema base, por lo que va a poseer todos los datos, nomencladores y funcionalidades comunes de los demás esquemas. Este cambio circunstancial posibilita que se minimice la redundancia de los datos y la inconsistencia en los mismos en los esquemas Penal y Común, que los datos comunes estén centralizados en un único esquema. Además, que se reduzca el número de tablas de la base de datos, posibilitando que se minimice el espacio de almacenamiento y que los programadores tengan mayor facilidad para la comprensión y desarrollo de su trabajo. Actualmente la base de datos cuenta con 300 entidades, 74 y 161 correspondientes a los esquemas Penal y Común respectivamente, lo que representa un 67.1 % del total de entidades. Es importante aclarar que este último contiene más tablas en la solución que se propone en comparación con la versión del diseño anterior y es aquí, donde se ve la minimización con respecto al número de entidades de la base de datos en general, ya que mientras más tablas posea este, menos entidades tendrán el resto de los esquema.

2.3. Arquitectura de la base de datos.

Cada tribunal contará con una base de datos independiente para gestionar toda la información. Al mismo tiempo existirá un centro de datos que almacenará la información de todos los tribunales y estará sincronizado con cada uno de ellos. Los tribunales no se comunicarán directamente entre sí, sino a través del centro de datos; de esta forma se garantiza la independencia de la tramitación judicial en cada uno de los tribunales y el control centralizado de toda la información.

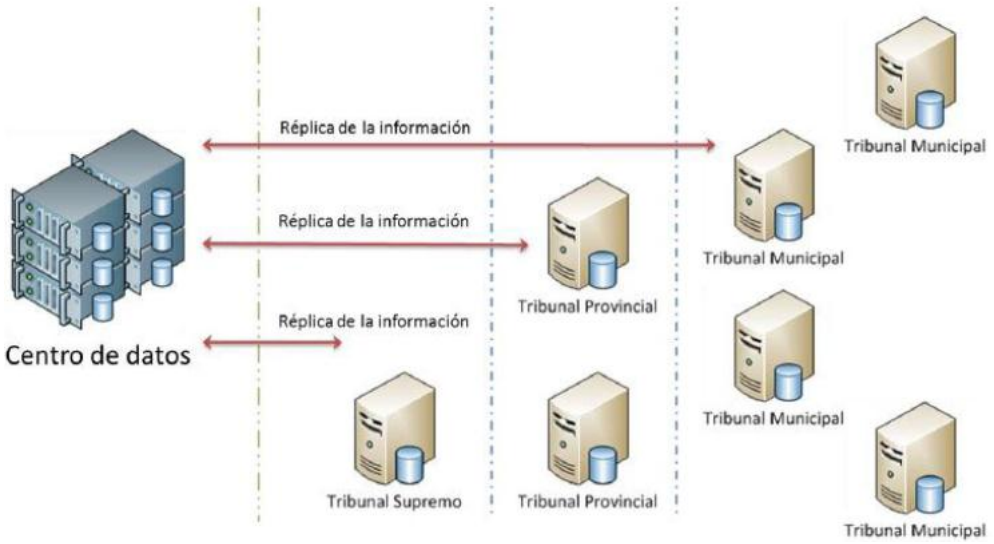


Figura 2. Distribución de la información

La arquitectura empleada se corresponde con una variante de un Sistema de Base de Datos Distribuidos[10].

Estos sistemas representan la estructura geográficamente descentralizada de una organización, aumentan la disponibilidad de los datos y reducen el tráfico de comunicación. Además, permite la reducción de los costos en el equipamiento y la infraestructura de comunicaciones de las redes.

2.3.1. Réplica.

Para lograr la sincronización de los datos de cada una de las instancias del sistema distribuido se utilizará la réplica de datos, que es la operación de transportar idénticamente la información de las tablas deseadas de una instancia a otra a través de la red, posibilitando la corrección, disponibilidad, coordinación y efectividad de los datos en un momento y lugar determinado.

Básicamente existen dos tipos o entornos de réplicas: El de solo lectura o maestro-esclavo (master-slave), que permite al nodo maestro realizar consultas de lectura/escritura, mientras que los nodos esclavos solo de lectura; y el otro entorno es el par-a-par o multi-maestro (multimaster), donde muchos nodos maestros interactúan

[10] Sistema de base de datos distribuido: colección de datos distribuidos en diferentes instancias a través de una red de computadoras.

entre sí, facilitando la sincronización de los cambios. El SITPC contará con un entorno de réplica multi-maestro o maestro-maestro y con una distribución asincrónica de la información.

La arquitectura definida y el establecimiento de sistemas distribuidos proporcionan un valor agregado a la hora de resguardar la información; por ejemplo, si el centro de datos pierde parte de su información, por algún motivo, una vez puesto en funcionamiento cada uno de los nodos restauraría la información contenida. Si esto ocurriera en alguna instancia, esta obtendría la información del centro de datos.

2.3.2. Copias de seguridad.

Las bases de datos de cada una de las instancias contemplarán un plan de copias de seguridad de los datos. Los tipos de copia pueden ser (González Flores 2012):

- *Copia de seguridad de copia:* Copia todos los archivos seleccionados pero no los marca individualmente como copiados (es decir, no desactiva el atributo de modificado). Este método es útil cuando se desea realizar copias de seguridad de archivos entre copias de seguridad normales e incrementales, ya que no afecta a estas otras operaciones.
- *Copia de seguridad diaria:* Copia todos los archivos seleccionados que se hayan modificado el día en que se realiza la copia diaria. Los archivos incluidos en la copia de seguridad no se marcan como copiados (es decir, no se desactiva el atributo de modificado).
- *Copia de seguridad diferencial:* Copia los archivos creados o modificados desde la última copia de seguridad normal o incremental. Los archivos no se marcan como copiados (es decir, no se desactiva el atributo de modificado). Si realiza una combinación de copias de seguridad normal y diferencial, para restaurar los archivos y las carpetas deben disponer de la última copia de seguridad normal y de la última copia de seguridad diferencial.
- *Copia de seguridad incremental:* Sólo copia los archivos creados o modificados desde la última copia de seguridad normal o incremental. Marca los archivos como copiados (es decir, se desactiva el atributo de modificado). Si usa una combinación de copias de seguridad normal e incremental, la restauración de

los datos debe realizarse con el último conjunto copia de seguridad normal y todos los conjuntos de copia de seguridad incremental.

- *Copia de seguridad normal:* Copia todos los archivos seleccionados y los marca como copiados (es decir, se desactiva el atributo de modificado). En las copias de seguridad normales sólo necesita la copia más reciente del archivo o la cinta que contiene la copia de seguridad para restaurar todos los archivos. Las copias de seguridad normales se suelen realizar al crear por primera vez un conjunto de copia de seguridad.

La combinación de copias de seguridad normal e incremental utiliza el mínimo espacio de almacenamiento posible y es el método de copia de seguridad más rápido. En el SITPC cada una de las instancias realizará una copia diferencial (diariamente), una copia incremental (semanalmente) y una copia de seguridad normal (mensualmente).

2.4. Propuesta de nomenclatura de la base de datos del SITPC.

Consideraciones generales de nomenclatura:

» Los nombres se escribirán según la Notación Pascal en el modelo lógico, están limitados a 35 caracteres y siempre comenzarán con una letra y nunca con un número ni caracter especial.

» Se evitará el empleo de abreviaturas puesto que puede malinterpretarse el nombre de acrónimos, ya que algunos de ellos tienen más de un significado.

» No se hará uso de tildes, diéresis ni espacios entre palabras.

» Se sustituirá la letra “ñ” por “nn”.

» Para nombrar se emplearán sustantivos en singular.

» Se utilizará la menor cantidad de los guiones bajos.

» Se hará uso de prefijos y sufijos para diferenciar los tipos de tablas existentes de acuerdo al negocio.

» Para el modelo físico los nombres se escribirán con minúscula para un mejor manejo de los datos en el gestor PostgreSQL.

Tablas:

» Las tablas nomencladoras comenzarán con la letra n.

<<*’n’* + *‘nombre de la tabla’*>>

Ejemplo: nEstado.

» El resto de las tablas comenzarán con la letra d.

<<*'d'* + *'nombre de la tabla'*>>
Ejemplo: dPersona.

» Las especializaciones se denotarán de la siguiente manera:
<<*'nombre de la tabla padre'* + *'nombre de la tabla hija'*>>
Ejemplo:

- Tabla padre: dPrueba.
- Tabla hija: dPruebaTestifical.

» Las tablas de cada esquema serán de colores distintos para diferenciarlas de los demás esquemas.

Columna:

» En los atributos de la tabla no se requerirá mencionar el nombre de la misma, a excepción del campo correspondiente a la llave primaria, el cual estará compuesto por:

<<*'id'* + *'nombre de la tabla'*>>
Ejemplo: De la tabla dPersona, la llave primaria es: IdPersona.

» La restricción de llave primaria se nombrará con el prefijo *Pk_* seguido del nombre de la tabla.

Ejemplo: Pk_dPersona

» Las llaves foráneas tendrán el mismo nombre que aparece en la tabla a la que hace referencia donde son llave primaria, excepto cuando se especifique un rol, y la relación correspondiente entre tablas. Debe nombrarse con los nombres de ambas tablas, primero con el nombre de la tabla padre y luego con el nombre de la tabla hija, con el prefijo *Fk_*.

» Las llaves alternativas se nombrarán con el prefijo *Ak_* seguido del nombre de la tabla.

» La ubicación de los campos nulos en las tablas, será en la última posición según sea el campo. Si es una llave extranjera, será de último entre estos, y si es un campo de la tabla, serán los últimos entre estos.

Índices:

» Los índices se nombrarán de la siguiente manera:

<<*'idx'* + *'nombre de la tabla'*+ *'nombre del campo'*>>

(Puede ser el nombre del campo o de los campos, según la característica de los índices).

Secuencias:

» Las secuencias se generarán automáticamente, especificándole al campo llave primaria que será de tipo serial. La nomenclatura de las mismas es la siguiente:

<<*'nombre de la tabla'* +*'_'*+ *'campo llave primaria'*+*'_'*+ *'seg'*>>

Estándar de codificación – Funciones:

» En las sentencias del tipo *select,* siempre se especificarán los campos a devolver y no con *.

» Las sentencias del tipo *select* en funciones, se ejecutarán sobre vistas previamente implementadas.

» Se hará uso de alias.

» Se manejarán excepciones solo de ser necesario, ya que es más costoso de entrar y salir del bloque que uno que no lo tenga.

» Se implementará la técnica de paginado para las búsquedas, devolviendo las primeras 100 tuplas.

» Se nombrarán según el tipo de funcionalidad a la que corresponden, preferentemente utilizando un verbo en infinitivo para describir el tipo de operación, seguido de un sustantivo, preferentemente el nombre de la tabla, utilizando _ en sustitución del espacio.

Ejemplo: registrar_usuario.

» Los parámetros comenzarán con la letra p.

Ejemplo: pnombre.

Vistas:

» Se nombrarán en plural utilizando _ en sustitución del espacio.

» Comenzarán con la letra v y se nombrarán solo con el tipo de información que devuelven.

Ejemplo: vdatos_usuarios.

» De ser posible se nombrarán igual a la tabla principal a la que hacen referencia.

Ejemplo: vusuarios.

» Las vistas se conformarán de forma genérica y con la mayor cantidad de información posible siempre y cuando sea conveniente.

Funciones:

» Las funciones se nombrarán con el prefijo *tf*.

» Las funciones trigger se nombrarán utilizando el prefijo *tg* seguido del nombre de la tabla.

» Se utilizará un solo trigger para la tabla correspondiente, con todos los eventos que sean necesarios.

2.5. Optimización de la base de datos del SITPC.

La optimización de la base de datos del SITPC depende en gran medida de las modificaciones que se realice en su diseño. Para esto se realizó un estudio severo junto al cliente, los analistas y otras personas involucradas en el tema, donde se descubrió que era necesario realizar algunas transformaciones de gran impacto en aras de eliminar redundancias y ambigüedades, normalizarlo hasta la forma que el negocio lo permita, adaptar el diseño a los nuevos requisitos definidos recientemente, conseguir la máxima seguridad e integridad de los datos y mejorar el consumo de recursos. A continuación se muestran las transformaciones realizadas por cada criterio que se propone en el proceso de optimización:

2.5.1. Empleo de patrones de diseño de base de datos.

Partiendo del modelo conceptual entregado por el equipo de Analistas del proyecto al equipo de Diseño de base de datos se realizó, haciendo uso de la herramienta ER/Studio en su versión 7.5, el diseño de los esquemas Común y Penal. Seguidamente se explican las principales transformaciones realizadas a los esquemas correspondientes a dichos subsistemas.

Diseño del esquema Común:

Dentro de las principales transformaciones que sufrió dicho esquema se encuentran:

Adición de tablas:

Por la nueva concepción que se posee del mismo se añadieron un conjunto de tablas que son comunes para todos los esquemas y que en la versión anterior se encontraban en cada uno de ellos. Entre las entidades que son consideradas más trascendentales se encuentran:

- *nLey:* contiene nomenclados las leyes judiciales vigentes en Cuba.
- *nOACE:* posee los Organismos de la Administración Central del Estado.
- *dPersona, dPersonaNatural y dPersonaJurídica*.
- *dExpediente:* contiene los documentos asociados a la tramitación judicial de cada caso que se inicia.
- *nTramite:* contiene los estados por los que puede pasar un trámite determinado y permite saber cuál es el próximo acto procesal que le continúa al anterior.

Diseño del esquema Penal:

Dentro de las principales transformaciones que se efectuaron en dicho esquema se pueden destacar:

Adición de tablas:

Por causa de que el diseño que existía en la versión anterior del sistema no satisfacía las necesidades que requerían los TPC y por cambios que han surgido en el negocio perteneciente a esta materia se añadieron un conjunto de entidades de trascendental relevancia, las cuales se muestran a continuación:

- *dSolicitudPrueba:* contiene las solicitudes de las pruebas que se proponen por un Fiscal en un caso determinado.

- *dSancionPenal:* contiene las sanciones sobre las imputaciones de los acusados.
- *dSancionAccesoria:* contiene las sanciones que constituyen daño colateral.
- *dFianza:* contiene las fianzas de los acusados.
- *nDisposicion:* contiene las diferentes disposiciones que se pueden realizar sobre las piezas y los bienes ocupados de un EFP.

1.5.1.1. Patrones de diseño de base de datos utilizados.

Dentro de los patrones de diseño de base de datos mencionados en el capítulo anterior, se emplearon para esta propuesta los siguientes:

- **Árbol simple:** Se empleó este patrón en la tabla nDPA para representar la estructura jerárquica de la División Político Administrativa (DPA) de cada país.

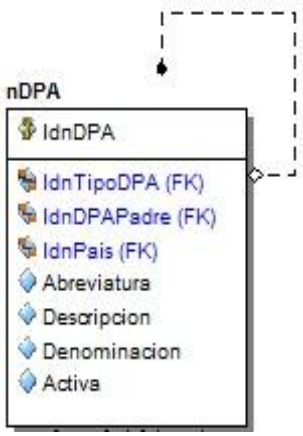


Figura 3: Representación del patrón árbol simple.

- **Máquina de estado para un tipo de entidad:** Este patrón se utiliza para representar los estados por los que puede pasar un trámite determinado y saber cuál es el próximo acto procesal que le continúa al anterior. De esta manera se sabe el flujo de trabajo de los procesos judiciales en los TPC.

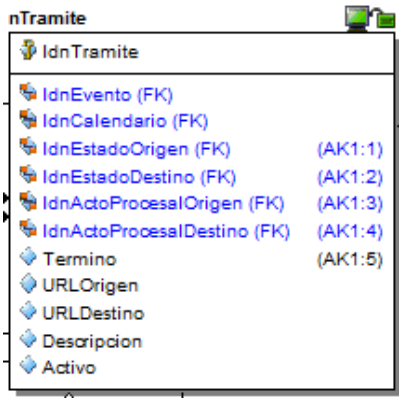


Figura 4: Núcleo de la máquina de estado.

- **Patrón de llaves subrogadas:** Este es uno de los patrones más utilizados en la base de datos. Por ejemplo, en las tablas dTramiteExpediente, dExpediente, dExpedienteEstado se evidencia la utilización del mismo, donde el atributo IdTramiteExpediente, IdExpediente, IdExpedienteEstado constituyen las llaves subrogadas de las entidades mencionadas respectivamente.

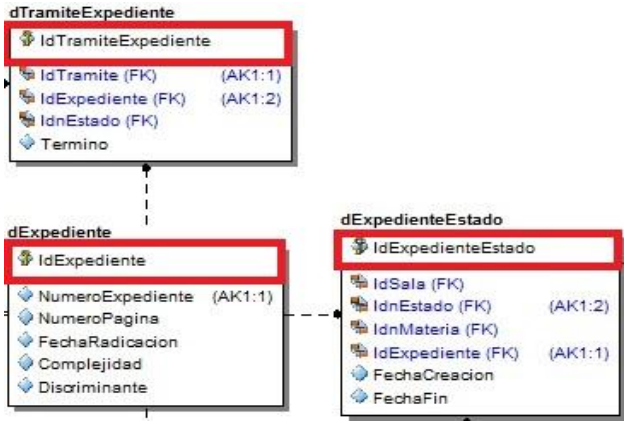


Figura 5: Representación del patrón de llaves subrogadas.

2.5.2. Normalización de los esquemas Penal y Común.

Los esquemas Común y Penal se normalizaron en primera forma normal ya que cada tupla contiene exactamente un valor para cada atributo de las tablas de la base de datos, o sea, no existen campos multivaluados, compuestos ni sus combinaciones. También se normalizaron en 2FN, pues se encuentran en 1FN y todos los atributos que no son claves en las tablas, dependen totalmente de la clave primaria que le

corresponde. Finalmente, se llevó a la tercera forma normal, por encontrarse en segunda forma normal y no tener dependencias transitivas en los atributos no primos de las entidades.

2.5.2.1. Desnormalización.

Con el propósito de obtener un mayor rendimiento en el acceso a los datos en las operaciones de selección y evitar el uso JOINs entre tablas, se desnormalizaron algunas entidades como dEscrito. En esta se evidencia que los atributos *Asunto* y *Definitivo* dependen del atributo no primo discriminante y este a su vez de la clave primaria *IdEscrito*. Esta modificación se debe a que cuando dicha entidad estaba normalizada tenía asociada 50 entidades correspondientes a los tipos de escrito. En la figura 6 y 7 se evidencia una muestra de lo expresado anteriormente. Además se desnormalizaron las entidades *dResolucionJudicial*, *dActa*, *dOficio*, *dDocumentoGenerado*.

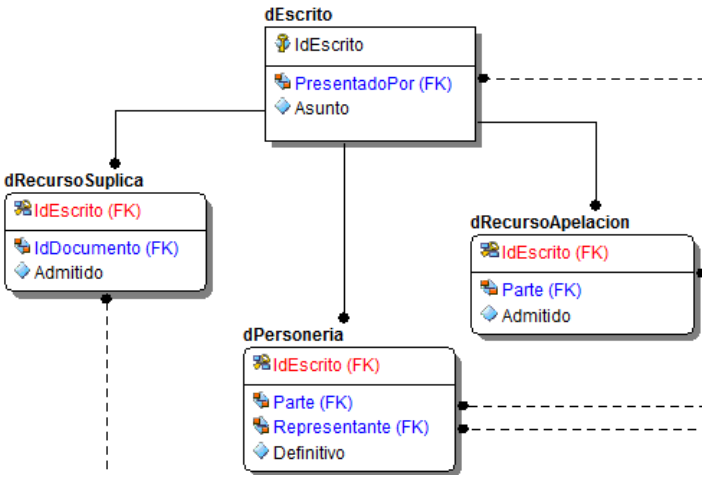


Figura 6: Muestra de la entidad dEscrito normalizada.

Figura 7: Muestra de la entidad dEscrito desnormalizada.

2.5.3. Índices.

En el desarrollo de la base de datos se utilizaron índices de tipo B-Tree, no porque constituye el índice por defecto que provee el sistema gestor PostgreSQL, sino por las potencialidades que brinda, las cuales están dadas precisamente por su estructura. Por ejemplo, cuando se van a realizar búsquedas de filas en las que un valor determinado aparezca no implica recorrer toda la tabla, sino que se utiliza la estructura arbórea del índice definido. Este proceso se realiza internamente bajando desde la raíz del árbol por una de las ramas hasta encontrar en las hojas las referencias a la fila en el fichero. Con esto se consume menos tiempo en hallar el resultado y se reduce el acceso al disco para leer.

Se indexaron atributos de tablas que contendrán grandes volúmenes de datos donde las búsquedas utilizando estos atributos serán muy frecuentes. En la tabla que se presenta seguidamente se evidencian algunos de estos atributos indexados y la entidad y el esquema al que pertenece cada uno de ellos:

Nombre de la Tabla	Atributo Indexado	Esquema
dPersonaNatural	NumeroIdentificacion	Común
dExpediente	NumeroExpediente	Común
dTribunal	NumeroTribunal	Común
dAcusado	SobreNombre	Penal
dEFP	NumeroEFP	Penal

Tabla 3: Ejemplos de atributos indexados.

2.5.4. Optimización de consultas SQL.

Es innegable que en términos de consumos de recursos algunas consultas consumen más que otras debido al cúmulo de información que devuelven. Esto puede incidir negativamente en que el gestor utilizado demore en devolver el resultado afectando así, el tiempo de respuesta de las peticiones que se realizan a la base de datos. En aras de minimizar ambos factores se propone el empleo de un conjunto de buenas prácticas para programar consultas SQL, las cuales se mencionan a continuación:

- Hacer uso de índices correctamente para la búsqueda de información ya que un uso indebido puede ralentizar dicho proceso.
- Utilizar lo menos posible la sentencia JOIN, ya que cuando se emplean bastante provocan que los tiempos de respuestas para cada petición que se le realice a la base de datos no sean los mejores.
- Evitar las consultas de tipo: *Select * From Tabla* ya que hace que el motor de base de datos emplee más tiempo en hacer un *DESCRIBE TABLE* y luego hacer un *Select campo1, campo2,..., campoN From Tabla* para devolver la información.
- Evitar las subconsultas dentro de los campos, o sea de la siguiente manera: *Select a, b, (Select…) From Tabla* ya que para que se ejecute la consulta principal se tiene que ejecutar la subconsulta primeramente. Esto provoca que la consulta se demore en devolver el resultado.
- Seleccionar solo aquellos campos que se necesiten ya que cada campo innecesario genera tiempo extra.
- Evitar los Orden by y los Distinct innecesarios ya que realizan un consumo innecesario de memoria.

2.5.5. Optimización en PostgreSQL.

Existen diversas estrategias para optimizar bases de datos en pos de incrementar su rendimiento. A continuación se mencionan algunas de ellas:

- **Mantenimiento a la base de datos.**

Existe un conjunto de actividades que el administrador de un sistema gestor de bases de datos debe tener presente constantemente, y que deberá realizar periódicamente. En el caso de PostgreSQL, cuando se ejecutan acciones como actualizaciones y/o eliminaciones en una base de datos, este internamente lo que hace es clonar esa tupla con la modificación realizada y marcar como no visible la anterior, la cual recibe el nombre de tupla muerta. Esto trae consigo que en bases de datos muy grandes, donde se realizan muchas de las acciones mencionadas anteriormente, se ocupe más especio de almacenamiento. Un ejemplo de esto lo constituye la base de datos que se utiliza en la investigación.

Con el propósito de reducir el espacio de almacenamiento se propone realizar cada un mes el mantenimiento a la base de datos en aras de limpiar o eliminar las tuplas muertas y actualizar las estadísticas del planificador de consultas. Esto se realiza mediante la sentencia VACUUM y ANALYZE.

Sintaxis:

» VACUUM VERBOSE Tabla

» VACUUM VERBOSE ANALYZE Tabla

VERBOSE: Imprime un reporte detallado de la actividad de limpieza para cada tabla.

ANALYZE: Actualiza las estadísticas usadas por el planificador para determinar la forma más eficiente de ejecutar una consulta determinada.

Tabla: Se coloca el nombre de la tabla a la que se le quiere realizar la limpieza.

Tomando como referencia que los servidores que se instalarán en cada una de las instancias de los TPC tendrán una memoria RAM de 4Gigabytes (GB), se propone que se incremente inicialmente los siguientes parámetros:

- **shared_buffers = 1024 MB**.
- **maintenance_work_mem = 256 MB.**

- **max_connections:** Para cada una de las instancias el máximo de conexiones será de 40**,** excepto en las de la Provincia de La Habana que tendrán de 80 a 100 conexiones a lo sumo.

Conclusiones Parciales.

Con el correcto empleo de los criterios de optimización en la propuesta de solución desarrollada se garantiza:

» La minimización de la redundancia de los datos e inconsistencia en los mismos en los esquemas Penal y Común a través de la normalización de ambos.

» La disminución de los tiempos de respuesta para cada petición y del consumo de recurso mediante la desnormalización de tablas específicas, el indexado de algunos atributos pertenecientes a las entidades que más se utilizan en los TPC de los esquemas mencionados anteriormente y la optimización de consultas SQL.

» Se redujo el espacio de almacenamiento mediante la normalización de los esquemas Penal y Común y el mantenimiento rutinario realizado a la base de datos.

» Por último, se puede afirmar que se dio cumplimiento al segundo objetivo específico trazado en la investigación.

Capítulo 3: Validación de la propuesta.

3.1. Introducción.

En el presente capítulo se plasmarán los resultados generados una vez realizadas las pruebas de rendimientos a la propuesta de solución planteada anteriormente. Dentro de las cuales se encuentran las pruebas de volumen, pruebas de carga y estrés y el comando EXPLAIN ANALYZE el cual se empleó para visualizar el comportamiento de las consultas SQL realizadas a la base de datos.

3.2. Pruebas de rendimiento.

Las pruebas de rendimiento se realizan desde la perspectiva de encontrar errores en los sistemas que incidan negativamente en el rendimiento del mismo. Estas pruebas pueden servir para demostrar que el sistema cumple los criterios de rendimiento, para comparar dos sistemas y encontrar cuál de ellos funciona mejor o para medir qué partes del sistema provocan que el rendimiento del mismo sea el no deseado.

Dentro de las pruebas que existen para evaluar el rendimiento de sistemas se encuentran las pruebas de Volumen, las pruebas de Carga y estrés, entre otras. Estas son las que se emplearon para evaluar el rendimiento de los esquemas Penal y Común de la base de datos del proyecto SITPC. A continuación se citan los resultados que arrojaron una vez realizadas dichas pruebas:

3.2.1 Pruebas de Volumen.

Esta prueba se realiza para analizar el comportamiento de la base de datos comprobando si la misma alcanza su límite de almacenamiento y puede causar fallas.

Es preciso tener presente la magnitud de datos que van a ser almacenados en la base de datos del proyecto SITPC. El país cuenta actualmente con 166 Tribunales Municipales, 15 Tribunales Provinciales y el Tribunal Supremo. Por cada instancia municipal y provincial se digitalizarán archivos judiciales que contienen expedientes que tienen entre 10 y 100 folios y en la instancia superior a las dos mencionadas anteriormente, los expedientes poseen entre 10 y 50 folios.

Con el empleo de la herramienta EMS Data Generator 2005 para PostgreSQL se poblaron los esquemas Común y Penal de la base de datos. La generación de datos se realizó para un estimado de datos de un período de 10 años. Por ejemplo, en la materia Penal por lo general se archivan más expedientes judiciales en comparación con las demás materias, anualmente dicha cifra oscila entre 2000 y 2700. De acuerdo a una aproximación por exceso que se realizó se estima que se archiven aproximadamente 3000 expedientes anuales. Cifra que multiplicada por 10, valor que representa la cantidad de años que se tomó como muestra para realizar la prueba, dio como resultado 30 000 expedientes aproximadamente. Por lo tanto, se deduce que cada 10 años se van a archivar aproximadamente treinta mil expedientes judiciales, cantidad que fue insertada en la entidad *dexpediente* cuando se realizó el llenado voluminoso de la misma. En todas las demás entidades fueron insertados 500 000 registros exceptuando a las tablas nomencladoras en las que se almacenaron 800 tuplas.

Sin embargo, en el año 2011 se realizó dicha prueba en la cual se insertó un máximo de 100 000 tuplas en la entidad dPersona. En el resto de las entidades se añadieron entre 1000 y 5000 registros.

La realización de esta prueba arrojó los siguientes resultados: no se presentaron problemas de límites de capacidad ni desbordamiento de columnas. Además, el diseño realizado de la base de datos y el SGBD utilizado soportan el cúmulo de información que será insertada una vez que el sistema esté en explotación en los Tribunales Populares Cubanos.

3.2.2 Pruebas de Carga y Estrés.

Este tipo de prueba se realiza básicamente para observar el comportamiento de la base de datos bajo una cantidad de peticiones esperada permitiendo determinar la solidez de la misma en los momentos de carga extrema y comprobar si rendirá lo suficiente en caso de que la carga real supere a la carga esperada.

Para la realización de la misma se empleó la herramienta JMeter para la cual se tuvieron en cuenta los siguientes valores:

- ✓ ***Número de hilos:*** es el 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 la prueba.

Grupo de hilos	Período de subida (en segundos)	Contador del bucle
100	1	10

Tabla 4: Configuración de los hilos para la prueba.

A continuación se muestran los resultados por indicadores del informe que arrojó dicha prueba:

- # Muestras: número de muestras de peticiones JDBC[11].
- Media: tiempo medio (en milisegundos) transcurrido para un conjunto de resultados.
- Mediana: mediana aritmética (elemento de una serie ordenada de valores crecientes de forma que la divide en dos partes iguales, superiores e inferiores a él).
- Mín: mínimo tiempo transcurrido para las muestras de peticiones JDBC.
- Máx: máximo tiempo transcurrido para las muestras de peticiones JDBC.
- % Error: porcentaje de las peticiones con errores.

[11] **JDBC:** Java Database Connectivity por sus siglas en inglés, es una especificación de un conjunto de clases y métodos que permiten a programas desarrollados en Java acceder a sistemas de bases de datos.

- Rendimiento: rendimiento medido en base a peticiones por segundo.

Label	# Muestras	Media	Mediana	Mín	Máx	% Error	Rendimiento
Petición JDBC	1000	1532	77	1	14461	0,00%	26,6/sec
TOTAL	1000	1532	77	1	14461	0,00%	26,6/sec

Figura 8: Informe de los resultados de la prueba.

La prueba se realizó simulando 100 usuarios con 10 iteraciones por cada uno, realizando una consulta a la base de datos para un total de 1000 peticiones JDBC lo que demoró 1.532 segundos en atender las solicitudes, cifra que se obtuvo al dividir la media/1000.

Sin embargo, cuando se realizó esta prueba en el año 2011 se simularon en inicialmente 300 conectados concurrentemente a la base de datos realizando 5 iteraciones cada uno de ellos para un total de 1500 peticiones JDBC demorando 1.2 segundos en atender las solicitudes. Luego se simularon 500 usuarios realizando 10 iteraciones cada uno para un total de 5000 peticiones JDBC y demoró en 2.1 segundo en atender las solicitudes efectuándose 5.22% de errores, lo que significa que aproximadamente 300 peticiones presentaron errores. En las figuras 9, 10 y 11 se evidencia lo expresado anteriormente:

Propiedades de los hilos	Prueba 1	Prueba 2
Número de hilos	300	500
Período de subida(en segundos)	1	1
Contador del bucle	5	10

Figura 9: Configuración de los hilos para las pruebas realizadas en el año 2011.

Label	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Petición JDBC	1500	1242	1109	3078	0	7593	0,00%	160,5/sec	29,2
TOTAL	1500	1242	1109	3078	0	7593	0,00%	160,5/sec	29,2

Figura 10: Informe de los resultados de la Prueba 1.

Label	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
Petición JDBC	5000	2189	234	7297	0	11031	5,22%	160,5/sec	27,6
TOTAL	5000	2189	234	7297	0	11031	5,22%	160,5/sec	27,6

Figura 11: Informe de los resultados de la Prueba 2.

Es preciso tener en cuenta que la prueba fue realizada en una estación de trabajo que no funciona como servidor dedicado, la cual tiene como propiedades un microprocesador Intel Core 2 Duo de 2.20 Gigahertz y una memoria RAM de 1 GB. Estas características serán superadas en el servidor que estará ubicado en cada una de las instancias de los TPC por lo que se estima que los resultados sean superiores cuando el sistema se esté en despliegue.

Teniendo en cuenta los argumentos expuestos anteriormente se evidencia que la prueba realizada en la presente investigación arrojó mejores resultados que las pruebas realizadas en el año 2011. Por lo tanto, se concluye que la base de datos soporta gran cantidad de peticiones o consultas realizadas simultáneamente y el tiempo de respuesta en atender las solicitudes será menor que los 3 segundos establecidos en los requisitos no funcionales del sistema.

3.2.3 EXPLAIN ANALYZE.

Este comando permite conocer el plan de ejecución de las consultas incluyendo los tiempos asociados a la misma y mejorarlas en caso de que los resultados no respondan a los requisitos establecidos.

Se realizaron pruebas de rendimiento a las consultas SQL de forma general en los esquemas Penal y Común. Por ejemplo, se realizó una consulta empleando la herramienta PGAdmin III con el propósito de obtener el nombre, los apellidos, el número de identificación, el estado civil, el nivel escolaridad, el sobre nombre, el sexo y el lugar de nacimiento de un determinado acusado, la cual quedó estructurada de la siguiente manera:

```
SELECT

  comun.dpersonanatural.primernombre,

  comun.dpersonanatural.primerapellido,

  comun.dpersonanatural.segundoapellido,
```

```
  comun.dpersonanatural.numeroidentificacion,
  comun.dpersonanatural.idnestadocivil,
  penal.dacusado.idnnivelescolaridad,
  penal.dacusado.sobrenombre,
  comun.dpersonanatural.sexo,
  comun.dpersonanatural.naturalde
FROM
  comun.dpersonanatural,
  comun.dlitigante,
  penal.dacusado
WHERE
  comun.dpersonanatural.idpersona = comun.dlitigante.idpersona AND
  comun.dlitigante.idlitigante = penal.dacusado.idlitigante
```

En la figura que se muestra a continuación se evidencia que el tiempo de respuesta que demora la misma es de 1.168 segundos menor que los 3 segundos establecidos en los requisitos no funcionales.

Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Hash Join (cost=130.00..330.05 rows=2000 width=119) (actual time=3.549..4.836 rows=1 loops=1)
2	Hash Cond: (dlitigante.idlitigante = dacusado.idlitigante)
3	-> Hash Join (cost=60.00..230.05 rows=2000 width=86) (actual time=1.720..3.007 rows=1 loops=1)
4	Hash Cond: (dpersonanatural.idpersona = dlitigante.idpersona)
5	-> Seq Scan on dpersonanatural (cost=0.00..137.70 rows=2470 width=86) (actual time=0.013..0.624 rows=2470 loops=1)
6	-> Hash (cost=35.00..35.00 rows=2000 width=16) (actual time=1.518..1.518 rows=2000 loops=1)
7	Buckets: 1024 Batches: 1 Memory Usage: 79kB
8	-> Seq Scan on dlitigante (cost=0.00..35.00 rows=2000 width=16) (actual time=0.011..0.558 rows=2000 loops=1)
9	-> Hash (cost=45.00..45.00 rows=2000 width=41) (actual time=1.799..1.799 rows=2000 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 136kB
11	-> Seq Scan on dacusado (cost=0.00..45.00 rows=2000 width=41) (actual time=0.028..0.821 rows=2000 loops=1)
12	Total runtime: 1.168 ms

Figura 12: Resultado de la consulta SQL realizada.

Además se probaron todos los índices de los esquemas Penal y Común realizando las mismas consultas con los atributos indexados y sin indexar obteniendo como resultado en todos los casos, que el tiempo de respuesta utilizando índices era inferior a los casos donde no eran utilizados. Por ejemplo, en la entidad *dacusado* del esquema Penal se realizó la prueba con el atributo *sobrenombre*. Seguidamente se mostrará la consulta empleada y los resultados de dicha prueba en las figuras 10 y 11:

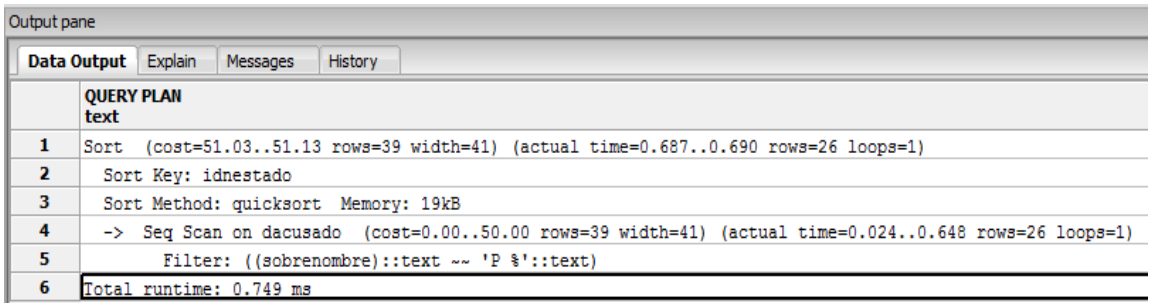
```
SELECT
  penal.dacusado.sobrenombre,
  penal.dacusado.idacusado
FROM
  penal.dacusado
WHERE
  penal.dacusado.sobrenombre LIKE 'P_%'
ORDER BY
  penal.dacusado.idnestado
```

Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Sort (cost=125.49..125.49 rows=1 width=42) (actual time=2.255..2.262 rows=26 loops=1)
2	Sort Key: idnestado
3	Sort Method: quicksort Memory: 20kB
4	-> Seq Scan on dacusado (cost=0.00..125.47 rows=1 width=42) (actual time=0.043..2.185 rows=37 loops=1)
5	Filter: ((sobrenombre)::text ~~ 'P %'::text)
6	Total runtime: 2.342 ms

Figura 13: Prueba con el atributo sobrenombre sin indexar.


Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Sort (cost=51.03..51.13 rows=39 width=41) (actual time=0.687..0.690 rows=26 loops=1)
2	Sort Key: idnestado
3	Sort Method: quicksort Memory: 19kB
4	-> Seq Scan on dacusado (cost=0.00..50.00 rows=39 width=41) (actual time=0.024..0.648 rows=26 loops=1)
5	Filter: ((sobrenombre)::text ~~ 'P %'::text)
6	Total runtime: 0.749 ms

Figura 14: Prueba con el atributo sobrenombre indexado.

En las entidades *dtribunal* y *dexpediente* perteneciente al esquema Común se realizaron de igual manera dichas pruebas con los atributos *numerotribunal* y *numeroexpediente*, en ambos casos se obtuvieron los mismos resultados (Ver anexo 4, 5 6 y 7 respectivamente).

Como conclusión se puede atestiguar que una vez realizada esta prueba el tiempo de respuesta de las consultas a la base de datos es relativamente rápido y cumple con los requisitos no funcionales establecidos para el desarrollo del Sistema de Informatización para la gestión de los Tribunales Populares Cubanos.

Conclusiones parciales.

A través de un conjunto de pruebas de rendimiento realizadas a la propuesta de solución planteada, se demostró que los resultados de estas pruebas son mejores en comparación con los resultados obtenidos en pruebas realizadas anteriormente. Por tales razones es posible asegurar que dicha propuesta tributa al rendimiento eficiente de la base de datos del proyecto SITPC. De esta manera, se dio cumplimiento al tercer y último objetivo específico de la investigación.

Conclusiones.

- Durante la investigación se estudiaron un conjunto de elementos teóricos permitiéndole al autor apropiarse de conocimientos necesarios para conformar la propuesta de solución.

- La propuesta de solución conformada se validó mediante un conjunto de pruebas de rendimiento como son las de Volumen y Carga y estrés, demostrando que los resultados obtenidos tributarán al rendimiento eficiente de la base de datos del proyecto SITPC.

- Se erradicaron los problemas planteados en la introducción del trabajo lo que significa que con esta propuesta se garantiza que se minimizó el espacio de almacenamiento y la redundancia de los datos, se disminuyeron los tiempos de respuesta y el consumo de recurso.

- Se puede afirmar que se le dio cumplimiento al objetivo general de la investigación.

Recomendaciones.

Se recomienda, luego de haber terminado el desarrollo de la presente investigación y teniendo en cuenta las ideas que surgieron en el transcurso de la misma:

- Aplicar la propuesta realizada y demostrada en el desarrollo del SITPC en otros proyectos con características similares pertenecientes al Centro de Gobierno Electrónico.
- Proporcionar mantenimiento y soporte a la base de datos del SITPC en aras de mejorar cada vez más su funcionamiento.

Referencias Bibliográficas.

1. Álvarez, S. (2010). "Desarrollo Web." Retrieved Septiembre, 2012, from http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.
2. Andrés, M. M. M. (2001). from http://www.uji.es/bin/publ/edicions/bdatos.pdf.
3. Bertino, E. A. and L. A. Martino (1995). Sistemas de bases de datos orientados a objetos, Días Santos.
4. Blaha, M. (2010). Patterns of Data Modeling.
5. Cabrera, M. N. G. (2010). from http://www.mailxmail.com/curso-diseno-creacion-bases-datos/sistemas-gestion-bases-datos-concepto-diseno-mer-normalizacion-3.
6. Castro, R. S. S. (2010). Elementos de análisis de algoritmos. Cuba
7. Diego, F. C. R. (2006). El álgebra relacional como soporte teórico de la verificación. La Paz. Bolivia.
8. Foundation, A. S. (1999-2013). "Apache JMeter." from http://jmeter.apache.org/.
9. G.W. Hansen, J. H. (1997). Diseño y Administración de Bases de Datos, Prentice Hall.
10. García, L. R. M. M. (1999). DISEÑO de BASES DE DATOS.
11. García, R. M. M. (2005). Sistemas de Bases de Datos.
12. González Flores, I., Gómez Perdomo, Y., & Fuentes Aguila, M. (2012). EL EXPEDIENTE JUDICIAL DIGITAL EN EL SISTEMA DE INFORMATIZACIÓN DE TRIBUNALES. . Universidad de las Ciencias Informáticas, La Habana, Cuba.
13. González, I. P. and M. S. Díaz (2011). Propuesta de optimización de la base de datos para el proyecto Sistema de Informatización de la Gestión de las Fiscalías. Facultad 3. La Habana, Universidad de las Ciencias Informáticas. **Ingeniero:** 83.
14. Group, P. G. D. (2009). "Portal en español sobre PostgreSQL." Retrieved Noviembre, 2012, from http://www.postgresql.org.es/node/655.
15. Hernández, M. J. (1997). Database Design for Mere Mortals, Addison-Wesley Developers Press.
16. Marqués, M. (2009). Bases de Datos. Valencia, España.
17. Martín, L. Y. E. (2009). Sistema para la Integración del Proceso de Normalización de Bases de Datos Relacionales con Gestores de Bases de

Datos. Facultad 9. La Habana, Universidad de las Ciencias Informáticas. **Máster**.

18. Navarro, L. P. (2010) "Pasos hacia la informatización de los tribunales." Granma **Volume**, DOI:
19. Navate, B. C. (2008). "Diseño conceptual de las bases de datos. Un enfoque de entidad-relación." from http://books.google.com/books?hl=es&lr=&id=-DP-0puz338C&oi=fnd&pg=PR18&dq=clasificaci%C3%B3n+de+las+base+de+datos&ots=_P.
20. Perdomo, M. H. and E. J. G. Fernández (2009). Guía para la optimización de servidores de bases de datos de PostgreSQL. Facultad 3. La Habana, Universidad de las Ciencias Informáticas. **Ingeniero**.
21. Pérez, N. A. R. (2012). Diseño e implementación de la base de datos del módulo Laboral para el Sistema de Informatización de Tribunales. Facultad 3. La Habana, Universidad de las Ciencias Informáticas. **Ingeniero:** 78.
22. Pérez, T. G. (2010). "Sistemas gestores de bases de datos." Innovación y Experencias Educativas.
23. Quesada, R. A. d. (1997). Tribunal Supremo Popular. Ley 82. De los Tribunales Populares.
24. Ramírez, R. Z. (2008). Sistemas Gestores de Bases de Datos. Córdoba.
25. Riquelme, M. S. L. and M. M. C. Morales (2010). Sistemas Gestores de Base de Datos Relacionales.
26. Sánchez, F. S. B. (2009). "Taller de Base de Datos." Retrieved Octubre, 2012, from http://www.slideshare.net/Mayra00/taller-de-base-de-datos.
27. Smith, G. (2009). "PostgreSQL 9.0." from www.wowebook.com.
28. Solutions, E. D. M. (1999 - 2013). "SQLManager.net." from http://www.sqlmanager.net/products/postgresql/datagenerator.
29. Technologies, E. (2009). "Embarcadero Technologies." from http://www.soluciones-ag.com/pdf_productos/Spanish_ER-Studio_Datasheet_2009.pdf.
30. Valero, E. M. M. and Z. M. Pérez (2011). Base de Datos Ingeniería de Sistemas. Universidad politécnica de la fuerza Armada Bolivariana, Zulia, Venezuela., Universidad politécnica de la fuerza Armada Bolivariana

Bibliografías.

1. Álvarez, S. (2010). "Desarrollo Web." Retrieved Septiembre, 2012, from http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.
2. Andrés, M. M. M. (2001). from http://www.uji.es/bin/publ/edicions/bdatos.pdf.
3. Bertino, E. A. and L. A. Martino (1995). Sistemas de bases de datos orientados a objetos, Días Santos.
4. Blaha, M. (2010). Patterns of Data Modeling.
5. Cabrera, M. N. G. (2010). from http://www.mailxmail.com/curso-diseno-creacion-bases-datos/sistemas-gestion-bases-datos-concepto-diseno-mer-normalizacion-3.
6. Castro, R. S. S. (2010). Elementos de análisis de algoritmos. Cuba
7. Diego, F. C. R. (2006). El álgebra relacional como soporte teórico de la verificación. La Paz. Bolivia.
8. Foundation, A. S. (1999-2013). "Apache JMeter." from http://jmeter.apache.org/.
9. G.W. Hansen, J. H. (1997). Diseño y Administración de Bases de Datos, Prentice Hall.
10. García, L. R. M. M. (1999). DISEÑO de BASES DE DATOS.
11. García, R. M. M. (2005). Sistemas de Bases de Datos.
12. González Flores, I., Gómez Perdomo, Y., & Fuentes Aguila, M. (2012). EL EXPEDIENTE JUDICIAL DIGITAL EN EL SISTEMA DE INFORMATIZACIÓN DE TRIBUNALES. . Universidad de las Ciencias Informáticas, La Habana, Cuba.
13. González, I. P. and M. S. Díaz (2011). Propuesta de optimización de la base de datos para el proyecto Sistema de Informatización de la Gestión de las Fiscalías. Facultad 3. La Habana, Universidad de las Ciencias Informáticas. **Ingeniero:** 83.
14. Group, P. G. D. (2009). "Portal en español sobre PostgreSQL." Retrieved Noviembre, 2012, from http://www.postgresql.org.es/node/655.
15. Hernández, M. J. (1997). Database Design for Mere Mortals, Addison-Wesley Developers Press.
16. Marqués, M. (2009). Bases de Datos. Valencia, España.
17. Martín, L. Y. E. (2009). Sistema para la Integración del Proceso de Normalización de Bases de Datos Relacionales con Gestores de Bases de

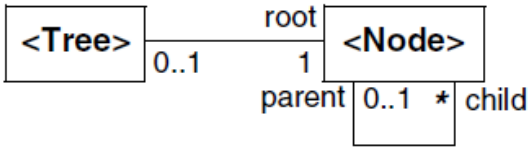
Datos. Facultad 9. La Habana, Universidad de las Ciencias Informáticas. **Máster**.

18. Navarro, L. P. (2010) "Pasos hacia la informatización de los tribunales." Granma **Volume**, DOI:
19. Navate, B. C. (2008). "Diseño conceptual de las bases de datos. Un enfoque de entidad-relación." from http://books.google.com/books?hl=es&lr=&id=-DP-0puz338C&oi=fnd&pg=PR18&dq=clasificaci%C3%B3n+de+las+base+de+datos&ots=_P.
20. Perdomo, M. H. and E. J. G. Fernández (2009). Guía para la optimización de servidores de bases de datos de PostgreSQL. Facultad 3. La Habana, Universidad de las Ciencias Informáticas. **Ingeniero**.
21. Pérez, N. A. R. (2012). Diseño e implementación de la base de datos del módulo Laboral para el Sistema de Informatización de Tribunales. Facultad 3. La Habana, Universidad de las Ciencias Informáticas. **Ingeniero:** 78.
22. Pérez, T. G. (2010). "Sistemas gestores de bases de datos." Innovación y Experencias Educativas.
23. Quesada, R. A. d. (1997). Tribunal Supremo Popular. Ley 82. De los Tribunales Populares.
24. Ramírez, R. Z. (2008). Sistemas Gestores de Bases de Datos. Córdoba.
25. Riquelme, M. S. L. and M. M. C. Morales (2010). Sistemas Gestores de Base de Datos Relacionales.
26. Sánchez, F. S. B. (2009). "Taller de Base de Datos." Retrieved Octubre, 2012, from http://www.slideshare.net/Mayra00/taller-de-base-de-datos.
27. Smith, G. (2009). "PostgreSQL 9.0." from www.wowebook.com.
28. Solutions, E. D. M. (1999 - 2013). "SQLManager.net." from http://www.sqlmanager.net/products/postgresql/datagenerator.
29. Technologies, E. (2009). "Embarcadero Technologies." from http://www.soluciones-ag.com/pdf_productos/Spanish_ER-Studio_Datasheet_2009.pdf.
30. Valero, E. M. M. and Z. M. Pérez (2011). Base de Datos Ingeniería de Sistemas. Universidad politécnica de la fuerza Armada Bolivariana, Zulia, Venezuela., Universidad politécnica de la fuerza Armada Bolivariana
31. Elmasri, R. and S. B. Navathe (2007). Fundamentos de Sistemas de Bases de Datos.

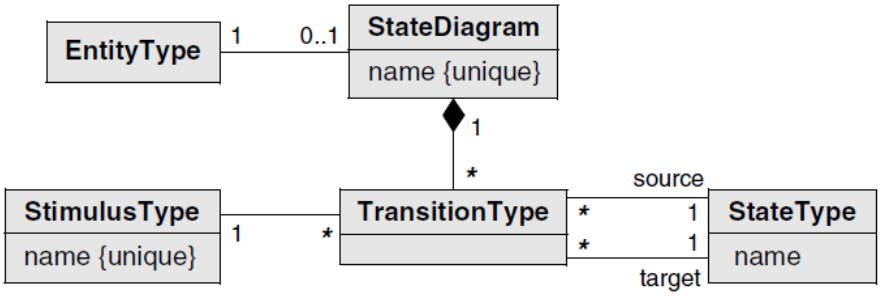
32. Group, P. G. D. (2009). "PostgreSQL 8.3.0 Documentation."
33. Marqués, M. (2011). Base de datos, Publicacions de la Universitat Jaume I. Servei de Comunicació i Publicacions Campus del Riu Sec.
34. Pressman, R. (2005). Ingeniería de Software Un Enfoque Práctico. La Habana, Félix Varela.
35. S, J. O. E. C. E. L. K. O. (2005). SQL FOR SMARTIES: ADVANCED SQL PROGRAMMING.
36. Tardieu, C. (2010). "Diseño de Base de datos relacional." from http://usuarios.multimania.es/cursosgbd/UD4.htm.

Anexos.

Anexo 1: Representación del patrón Árbol simple usando notación UML.



Anexo 2: Representación del patrón Máquina de estado para un tipo de entidad usando notación UML.



Anexo 3: Entrevistas realizadas en el transcurso de la investigación.

Esta entrevista se les realizó a los profesores Anthony Rafael Sotolongo León el cual se desempeña como Jefe del departamento PostgreSQL del Centro de Tecnologías de Gestión de Datos y a Yoan Carlos Machado Espinosa, Arquitecto de datos del departamento de Gestión Universitaria del Centro de Informatización Universitaria:

Pregunta # 1: Conoce usted alguna herramienta libre para el modelado de datos.

Pregunta # 2: ¿Qué consideraciones usted cree que se deben de tener para programar consultas SQL optimizadas?

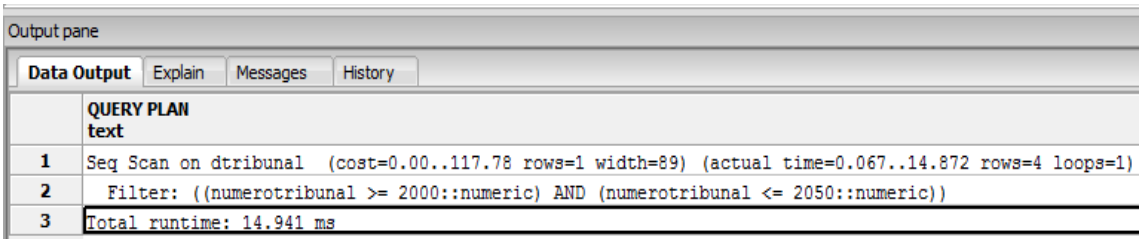
Pregunta # 3: De los parámetros que posee el gestor PostgreSQL en el archivo Postgresql.conf, para usted, ¿cuáles de ellos constituyen una prioridad para incrementar el rendimiento de una base de datos?

Anexo 4: Prueba realizada con el atributo numerotribunal sin indexar.

```
SELECT

  comun.dtribunal.idninstancia,
```

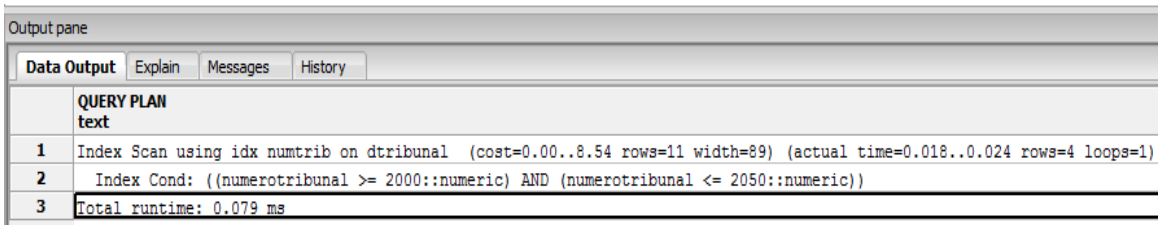
```
  comun.dtribunal.nombre,
  comun.dtribunal.iddireccion,
  comun.dtribunal.idhorariohabil,
  comun.dtribunal.annofiscalactual,
  comun.dtribunal.idtribunalpadre
FROM
  comun.dtribunal
WHERE
  comun.dtribunal.numerotribunal BETWEEN 2000 AND 2050
```



Anexo 5: Prueba realizada con el atributo numerotribunal indexado.

```
SELECT
  comun.dtribunal.idninstancia,
  comun.dtribunal.nombre,
  comun.dtribunal.iddireccion,
  comun.dtribunal.idhorariohabil,
  comun.dtribunal.annofiscalactual,
  comun.dtribunal.idtribunalpadre
FROM
  comun.dtribunal
```

```
WHERE
  comun.dtribunal.numerotribunal BETWEEN 2000 AND 2050
```



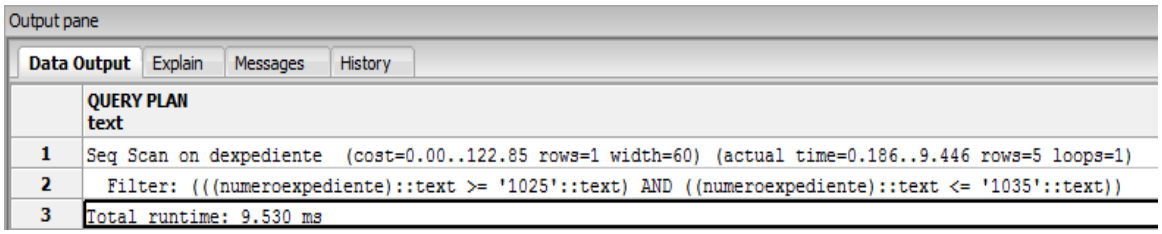
Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Index Scan using idx numtrib on dtribunal (cost=0.00..8.54 rows=11 width=89) (actual time=0.018..0.024 rows=4 loops=1)
2	Index Cond: ((numerotribunal >= 2000::numeric) AND (numerotribunal <= 2050::numeric))
3	Total runtime: 0.079 ms

Anexo 6: Prueba realizada con el atributo numeroexpediente sin indexar.

```
SELECT
  comun.dexpediente.idexpediente,
  comun.dexpediente.numeropagina,
  comun.dexpediente.fecharadicacion,
  comun.dexpediente.discriminante,
  comun.dexpediente.complejidad
FROM
  comun.dexpediente
WHERE
  comun.dexpediente.numeroexpediente BETWEEN '1025' AND '1035'
```



Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Seq Scan on dexpediente (cost=0.00..122.85 rows=1 width=60) (actual time=0.186..9.446 rows=5 loops=1)
2	Filter: (((numeroexpediente)::text >= '1025'::text) AND ((numeroexpediente)::text <= '1035'::text))
3	Total runtime: 9.530 ms

Anexo 7: Prueba realizada con el atributo numeroexpediente indexado.

```
SELECT
```

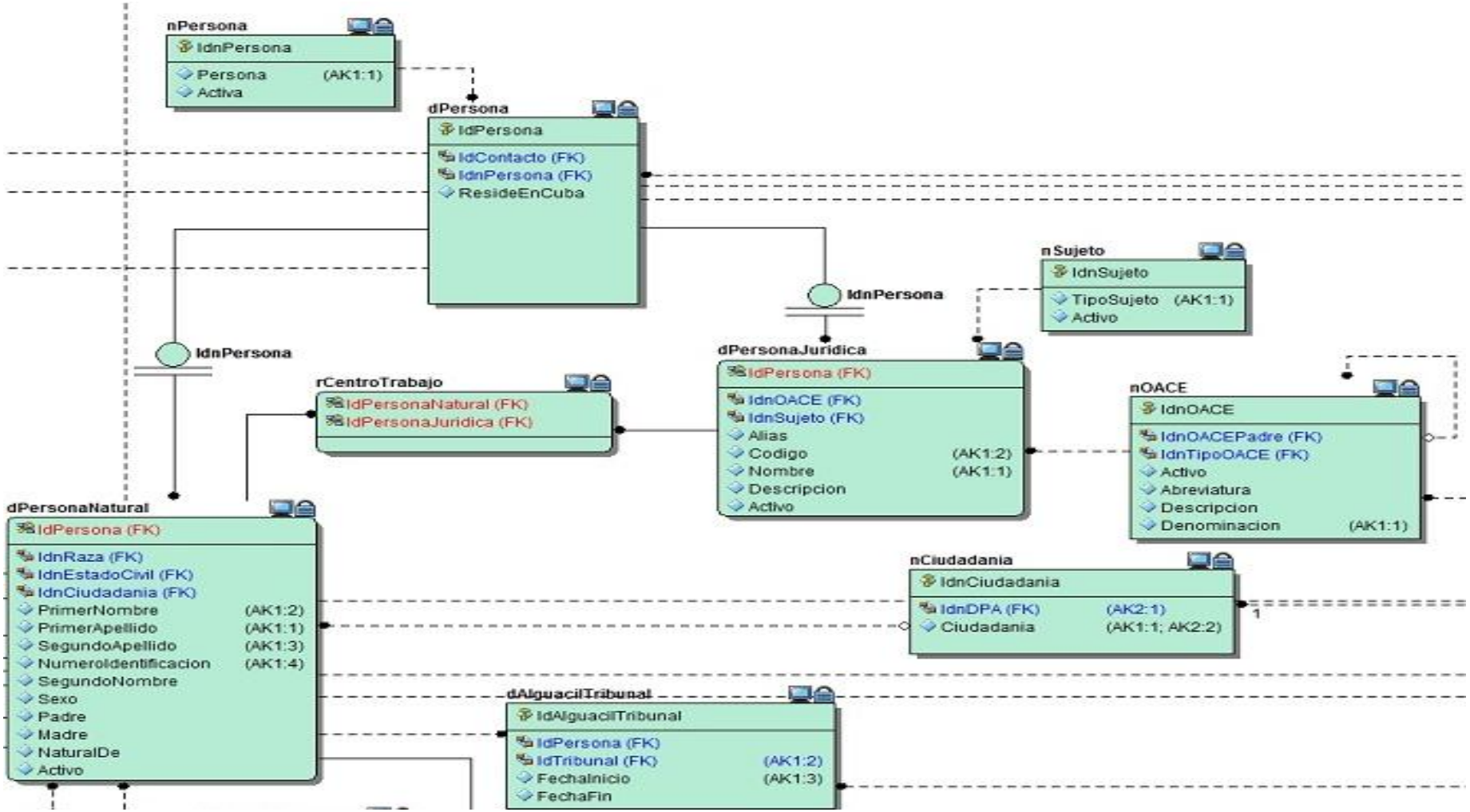
```
  comun.dexpediente.idexpediente,
  comun.dexpediente.numeropagina,
  comun.dexpediente.fecharadicacion,
  comun.dexpediente.discriminante,
  comun.dexpediente.complejidad
FROM
  comun.dexpediente
WHERE
  comun.dexpediente.numeroexpediente BETWEEN '1025' AND '1035'
```

Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Index Scan using idx numexp on dexpediente (cost=0.00..8.27 rows=1 width=60) (actual time=0.048..0.054 rows=5 loops=1)
2	Index Cond: (((numeroexpediente)::text >= '1025'::text) AND ((numeroexpediente)::text <= '1035'::text))
3	Total runtime: 0.108 ms

Anexo 8: Muestra del diseño del esquema Común.


nPersona
IdnPersona
Persona (AK1:1)
Activa
dPersona
IdPersona
IdContacto (FK)
IdnPersona (FK)
ResideEnCuba
IdnPersona
IdnPersona
nSujeto
IdnSujeto
TipoSujeto (AK1:1)
Activo
dPersonaJuridica
IdPersona (FK)
IdnOACE (FK)
IdnSujeto (FK)
Alias
Codigo (AK1:2)
Nombre (AK1:1)
Descripcion
Activo
rCentroTrabajo
IdPersonaNatural (FK)
IdPersonaJuridica (FK)
nOACE
IdnOACE
IdnOACEPadre (FK)
IdnTipoOACE (FK)
Activo
Abreviatura
Descripcion
Denominacion (AK1:1)
dPersonaNatural
IdPersona (FK)
IdnRaza (FK)
IdnEstadoCivil (FK)
IdnCiudadania (FK)
PrimerNombre (AK1:2)
PrimerApellido (AK1:1)
SegundoApellido (AK1:3)
NumeroIdentificacion (AK1:4)
SegundoNombre
Sexo
Padre
Madre
NaturalDe
Activo
nCiudadania
IdnCiudadania
IdnDPA (FK) (AK2:1)
Ciudadania (AK1:1; AK2:2)
1
dAlguacilTribunal
IdAlguacilTribunal
IdPersona (FK)
IdTribunal (FK) (AK1:2)
FechaInicio (AK1:3)
FechaFin

Anexo 9: Muestra del diseño del esquema Penal.