Universidad de las Ciencias Informáticas

Facultad 3

Diseño e implementación de la base de datos de los módulos Queja, Peticiones y Denuncias, y Comunes.

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

Autor:

Abel Andres Irsula Tumbarell

Tutor: Ing. Yenier Figueroa Machado

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

Ing. José Carlos Pupo Acosta

La Habana, Curso 2012-2013

El optimismo es la fe que conduce al éxito. Nada puede hacerse sin esperanza y confianza.

Helen Keller

Declaración de autoría

Se declara que Abel Andres Irsula Tumbarell es el único autor del trabajo de diploma Diseño e implementación de la base de datos de los módulos Queja, Peticiones y Denuncias, y Comunes y se le reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Firma del autor Firma del tutor

Abel Andres Irsula Tumbarell Ing. Yenier Figueroa Machado

Datos de contacto

Datos de tutor:

Nombre: Ing. Yenier Figueroa Machado

Ingeniero en Ciencias Informáticas. 6 años de experiencia laboral. Jefe del proyecto Sistema de Informatización de Gestión de las Fiscalías fase II.

Correo electrónico: yfigueroa@uci.cu

Datos de Co-Tutor:

Nombre: Ing. Manuel Álvarez Alonso

Ingeniero en Ciencias Informáticas.

Correo electrónico: malvareza@uci.cu

Datos de Co-Tutor:

Nombre: Ing. José Carlos Pupo Acosta

Ingeniero en Ciencias Informáticas.

Correo electrónico: jcpupo@uci.u

Datos del Autor:

Nombre: Abel Andres Irsula Tumbarell.

Correo electrónico: aairsula@estudiantes.uci.cu

Agradecimientos

A mi hermana Olivia, por toda la ayuda desde el corazón que supo darme durante todo el tiempo de estudio.

A mis hermanos Félix, Luis, Lismer y a toda mi familia, por el apoyo incondicional durante toda la carrera.

A la Revolución, por haberme dado la posibilidad de estudiar esta carrera que tanto deseé.

A la UCI, por albergarme durante todo este tiempo.

A las amistades que quiero como hermanos, Yunior, Ángel Delvis, Joaquín, Riquene, y Darianna.

A Arianna, Pepe, Polanco, Tony, Arlety, Yosvany, Dannelis, Adrián, Luis Manuel, al Tribilín por convertirse en la familia de mi nueva casa.

A mis co-tutores, por volverse como padres.

A Hortencia por traerme la fuerza en los momentos difíciles.

Al Movimiento Código y Letra, y a la FEU de la Facultad 3, por darme las experiencias inolvidables en otras esferas de la vida.

A todos aquellos que conocí en la universidad, que alguna vez hicieron hincapié en hacerme mejorar como estudiante y persona.

Dedicatoria

A mi mamá por impulsarme a lograr todas mis metas y sueños toda mi vida.

A mi papá y mi hermano Aniel, por ser ejemplos de abnegación y lucha contra los desafíos.

Resumen

Las aplicaciones de gestión en la actualidad tienen la necesidad de responder a la gestión de datos, los cuales sin depender del volumen que presenten deben estar almacenados de forma segura y con integridad. Para la Fiscalía General de la República se le desarrolla un sistema de gestión en la Universidad de las Ciencias Informáticas, Sistema de Informatización de la Gestión de las Fiscalías, SIGEF II, el cual necesita para sus módulos, Queja, Peticiones y Denuncias, y Comunes una base de datos para almacenar y gestionar la información. De ahí que el objetivo de este trabajo sea desarrollar el diseño y la implementación de una base de datos para los módulos Queja, Peticiones y Denuncias, y Comunes que contribuya a la seguridad de la información y la integridad de los datos. Para llegar a la solución se toma como guía de desarrollo la metodología Dbplanning Framework, como herramientas el sistema gestor de base de datos PostgreSQL 9.1, Visual Paradigm 8.0 para el modelado y el NetBeans 7.2 para describir el acceso a datos del framework para el mapeo relacional de objetos Doctrine 2.3.2.

PALABRAS CLAVE

almacenar, gestionar, integridad

Índice

Introducción 1

Capítulo 1. Fundamentación teórica. 6

1.1 Introducción. 6

1.2 Base de datos y sistemas gestores de bases de datos. 6

1.2.1 Base de datos (BD). 6

1.2.2 Sistemas gestores de base de datos (SGBD). 7

1.2.3 PostgreSQL 9.1 8

1.2.4 Estrategias de indexado. 8

1.3 Modelo de datos. 9

1.3.1 Modelo relacional. 11

1.3.2 Modelo entidad-relación. 11

1.4 Diseño de base de datos. 12

1.4.1 Fases del diseño de base de datos. 12

1.4.2 Patrones de diseño de bases de datos. 13

1.4.3 Diseño de base de datos a través de la metodología Dbplanning Framework. 18

1.4.4 Características generales de Dbplanning Framenwork. 18

1.5 Normalización de base de datos. 19

1.6 Integridad en las base de datos. 22

1.7 Seguridad en las base de datos. 23

1.8 Herramientas del entorno de desarrollo. 23

1.8.1 Lenguaje de modelado UML 2.0. 24

1.8.2 Visual Paradigm 8.0. 24

1.8.3 Subversion 1.5. 25

1.8.4 RapidSVN 0.12.1 25
1.8.5 PGAdmin III 1.14.0 26
1.8.6 ORM Doctrine 2.3.2. 26
1.8.7 IDE Netbeans 7.2. 26
1.9 Herramientas para las pruebas. 27
1.9.1 EMS Data Generator 2005 for PostgreSQL 2.21. 27
1.9.2 Apache JMeter 2.9. 28
1.10 Conclusiones parciales. 29
Capítulo 2. Análisis de la solución propuesta. 30
2.1 Introducción. 30
2.2 Configuración del entorno de desarrollo. 30
2.2.1 Usuarios y privilegios. 31
2.3 Arquitectura de datos. 31
2.4 Nomenclatura y normas para el modelo de base de datos. 33
2.4.1 Generalidades. 33
2.4.2 Nomenclatura para entidades. 33
2.4.3 Nomenclatura para atributos. 34
2.5 Descripción general de los modelos entidad relación. 34
2.5.1 Descripción de las principales tablas del módulo Comunes. 34
2.5.2 Descripción de las principales tablas del módulo QPD de CLEP. 37
2.5.3 Patrones utilizados en el diseño. 39
2.6 Optimización. 40
2.6.1 Normalización del modelo. 40

2.6.2 Estrategia de indexado utilizada. 41

2.6.3 Mantenimiento de la de la base de datos (VACUUM). 41

2.7 Acceso a datos. 42

2.7.1 Asignación de llaves a través del ORM. 44

2.8 Conclusiones parciales. 45

Capítulo 3. Validación y pruebas. 46

3.1 Introducción. 46

3.2 Validación teórica. 46

3.2.1 Integridad de la base de datos. 46

3.2.2 Transacciones. 47

3.3 Validación Funcional. 49

3.3.1 Pruebas de volumen. 49

3.3.2 Pruebas de rendimiento. 51

3.3.3 Pruebas de carga y estrés. 53

3.4 Conclusiones parciales. 56

Conclusiones Generales 57

Recomendaciones 58

Bibliografía 59

Anexos 63

Índice de figuras.

Fig. 1 Árbol fuertemente codificado.....14

Fig. 2 Árbol simple, notación UML 14

Fig. 3 Árbol estructurado en la notación UML. 15

Fig. 4 Grafo dirigido simple en UML.....15

Fig. 5 Grafo dirigido estructurado en UML. 16

Fig. 6 Máquina de estado para un tipo de entidad representado usando notación UML para clases. 16

Fig. 7 Máquina de estado para un escenario en notación UML.....17

Fig. 8 Patrón Entidad-Atributo-Valor en notación UML.....17

Fig. 9 Relación de fases y actividades de Dbplanning Framework.....19

Fig. 10 Distribución de los datos.....32

Fig. 11 Tablas principales del módulo Comunes.....35

Fig. 12 Tablas principales del módulo QPD de CLEP. 37

Fig. 13 Entidad ndivision del módulo Comunes.....40

Fig. 14 Entidad dfiscalia_ndivision del módulo Comunes.....40

Fig. 15 Entidad dantecedentes del módulo QDP.....40

Fig. 16 Código del ORM Doctrine 2 sobre clases entidades. 43

Fig. 17 Código DQL del ORM Doctrine 2. 43

Fig. 18 Identificador generado para una tupla en la entidad dproceso. 44

Fig. 19 Método de asignación de llaves a través del ORM.....45

Fig. 20 Ejemplo de Enfoque Implícito.....47

Fig. 21 Ejemplo de Enfoque Explícito. 48

Fig. 22 Ejemplo de Enfoque Explícito, implementado en la solución. 48

Fig. 23 Respuesta de prueba de volumen 50

Fig. 24 Consulta a realizar como prueba 52

Fig. 25 Resultado de consulta sin indexado. 53

Fig. 26 Resultado de consulta con indexado. 53

Fig. 27 Respuesta de Jmeter Prueba 1. 55

Fig. 28 Respuesta de Jmeter Prueba 2. 56

Índice de tablas

Tabla 1 Relación de herramientas y funciones en el entorno de desarrollo 30

Tabla 2 Entidad dpersona del módulo Comunes 36

Tabla 3 Entidad dfiscalia del módulo Comunes 36

Tabla 4 Entidad dproceso del módulo Comunes 36

Tabla 5 Entidad ddireccion del módulo Comunes 37

Tabla 6 Entidad dqueja del módulo Queja, peticiones y denuncias 38

Tabla 7 Entidad dprocesarqueja del módulo Queja, peticiones y denuncias 38

Tabla 8 Entidad dregistrarqueja del módulo Queja, peticiones y denuncias 38

Tabla 9 Entidad dplanteamiento del módulo Queja, peticiones y denuncias 39

Tabla 10 Entidad dinforme del módulo Queja, peticiones y denuncias 39

Tabla 11 Algunos campos indexados en la entidad dproceso 41

Tabla 12 Descripción de integridad de dominio 47

Tabla 13 Capacidad de Almacenamiento de PostgreSQL 9.1 51

Tabla 14 Tiempos para consultas 51

Tabla 15 Datos cargados para prueba de rendimiento 52

Tabla 16 Cantidad de trabajadores por tipo de fiscalía 54

Tabla 17 Propiedades de los hilos de las pruebas 55

Introducción

Almacenar información, tener acceso a ella y a su manejo, ha sido una necesidad de todos los tiempos de la humanidad. El desarrollo y evolución de las nuevas Tecnologías de la Información y las Comunicaciones (TIC) ha propiciado cambios que alcanzan estos ámbitos de la actividad humana. Gracias al surgimiento de las base de datos, como fuente de almacenamiento, en la ciencia de la informática, la información ha encontrado una mejor forma de almacenarse y tratarse.

Cuba sin estar ajeno a este desarrollo, enfrenta el difícil reto de informatizar disímiles tareas sociales, políticas y económicas, donde almacenar información es uno de los elementos importantes. Varias instituciones han surgido para desarrollar soluciones de este tipo. Entre estas se encuentra la Universidad de las Ciencias Informáticas (UCI), convertido en uno de los centros que marcha a la vanguardia en este sector en el país, "*…con la misión de formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Servir de soporte a la industria cubana de la informática*" (UCI, 2011).

En la UCI existen varios centros de producción de software, y entre ellos se encuentra el Centro de Gobierno Electrónico (CEGEL) perteneciente a la Facultad 3, el cual desde su creación ha dedicado sus esfuerzos a la investigación y desarrollo de sistemas informáticos para el área del gobierno electrónico, incorporando resultados importantes en el campo de la *informática jurídica*[1]. Instituciones como la Fiscalía General de la República (FGR), que desde 1973, al salir la Ley No.1250, es la encargada de controlar el estricto cumplimiento de la constitución del país; ha encontrado una opción para disminuir esfuerzo y tiempo, en estos momentos donde que se encuentra inmersa en la mejora de sus procesos para un mejor desempeño de su función social y política, por lo que ha solicitado la creación de un sistema que le permita gestionar sus procesos.

El Sistema de Informatización de la Gestión de las Fiscalías 2 (SIGEF II) es la solución en desarrollo donde se recogerá toda la información de las fiscalías; tiene como objetivo optimizar la calidad de la tramitación, supervisión y control en tiempo los procesos fiscales, teniendo como premisa obtener una fuerza fiscal con mayor economía y seguridad en cada uno de sus procesos.

El sistema SIGEF II está compuesto por varios subsistemas o áreas, las cuales comprenden módulos correspondientes a las tareas que en la institución se realizan. Entre estas áreas se

[1] Informática jurídica: Conjunto de aplicaciones de la informática en el ámbito jurídico

encuentra Control de la Legalidad de los Establecimientos Penitenciarios (CLEP), encargada de controlar todo lo relacionado con visitas de inspección a los establecimientos penitenciarios, las quejas, peticiones o denuncias de los familiares de las personas implicadas en algún delito, o de las que son internas o detenidas en la misma, respecto a la documentación que se genera en estas instituciones.

Los módulos derivados en esta área son:

- Quejas, Peticiones y Denuncias (QPD).
- Dictámenes
- Inspección a locales de detección y centros penitenciarios.

El sistema actual de trabajo en este subsistema no satisface las necesidades de la institución, sumado a esto que el personal existente es insuficiente para cubrir estas necesidades, provocando que se presenten afectaciones en el manejo de la información como:

- Recepción de la información de forma manuscrita, dando la posibilidad de errores en los archivos, que pueden ser de escritura, tachaduras y repetición de palabras.
- Tendencia a la acumulación de estos archivos, por la poca capacidad de procesamiento de documentos que se generan.
- Los reportes emitidos sobre el estado de los mismos no ofrecen la información necesaria sobre los datos que se están manejando.
- El deterioro que sufren estos documentos al cabo del tiempo, provoca que esto se vuelvan borrosos y de difícil manejo.
- La pérdida de relevancia que se le asigna a estos documentos, dependiendo en el área que estén, termina imposibilitando su uso en el futuro.

Para resolver estos problemas respecto al tratamiento de la información de QPD del subsistema CLEP, se ha creado un módulo de igual nombre. El cual tendrá la capacidad de gestionar gran volumen de información, de forma estructurada y organizada para facilitar su acceso y gestión a partir de los cambios que se realicen así como permitir el registro de nuevos datos.

Además se ha creado un módulo de Comunes, donde se agrupan características similares de todas las áreas o subsistemas, proporcionando la posibilidad de centralizar información con la que todos módulos trabajarían, propiciando ventajas de rapidez de gestión, y reduccion de la ocurencia de redundacia en la informacion.

En estos momentos no se cuenta con una estructura que permita almacenar los datos de los procesos de QPD y centralizar la información común con la que se trabaja. Por lo que las circunstancias derivan al siguiente **problema a resolver:** ¿Cómo garantizar el almacenamiento y la gestión de la información de los módulos Queja, Peticiones y Denuncias, y Comunes de forma que se contribuya a la seguridad de la información y la integridad de los datos?

Como **objeto de estudio**: Desarrollo de base de datos operacionales.

Del problema descrito anteriormente se identifica como **objetivo general,** desarrollar el diseño y la implementación de una base de datos para los módulos Queja, Peticiones y Denuncias, y Comunes que contribuya a la seguridad de la información y la integridad de los datos.

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

Para darle cumplimiento al objetivo general propuesto se han definido los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Diseñar la propuesta de la base de datos para los módulos Queja, Peticiones y Denuncias, y Comunes.
- Implementar la propuesta de la base de datos para los módulos Queja, Peticiones y Denuncias, y Comunes.
- Validar la solución propuesta a través de pruebas.

Para desarrollar el presente trabajo, es necesario definir **tareas de la investigación** que guíen el camino a seguir para dar cumplimiento a los objetivos específicos:

- Estudio del marco teórico que fundamenta el objeto de investigación.
- Selección de la herramienta de modelado.
- Selección del gestor de base de datos.
- Diseño de los Modelos de Datos.
- Implementación de la base de datos.
- Selección de las pruebas a realizar para validar la solución.
- Validación de la solución propuesta.

Para realizar la investigación se utilizaron los siguientes métodos de investigación:

Métodos teóricos:

- Histórico-lógico: se empleó para analizar la trayectoria y evolución de los sistemas de bases de datos jurídicos.
- Analítico-sintético: permitió el procesamiento de la información y arribar a las conclusiones prácticas y teóricas de la investigación, así como precisar las herramientas utilizadas para el diseño y la implementación de la base de datos.
- Modelación: se utilizó para la creación de abstracciones que explican la realidad, este método es un instrumento de la investigación que permite la creación de modelos, descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio.

Método empírico:

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

El contenido del presente trabajo está estructurado de la siguiente forma:

Capítulo 1: Trata todo el basamento en que se fundamenta teóricamente el trabajo, ubica al lector en las tendencias del uso de bases de datos, abordándose las características y el funcionamiento de las mismas. En este capítulo se desarrollan temas relacionados con los modelos de datos y las características del diseño de bases de datos. Además se analizan herramientas relacionadas con el modelado, la administración y la gestión de bases de datos.

Capítulo 2: Se describe la propuesta de solución obtenida, las características de la base de datos de los módulos Quejas, Peticiones y Denuncias, y Comunes del Sistema de Informatización de la Gestión de las Fiscalías 2.Se describe el estándar de nomenclatura utilizado, muestra las características generales del diseño, se ejemplifica los patrones de bases de datos utilizados, se presenta el modelo físico, la utilización de la normalización y los índices. También se describen acciones tomadas para asegurar la integridad y la seguridad de la base de datos.

Capítulo 3: Los aspectos de seguridad implementados sobre la base de datos, los elementos relacionados con la validación de la propuesta de solución está abordado en este capítulo, lo que incluye las pruebas que se realizan de volumen, para analizar el comportamiento de la base de datos al almacenar grandes cantidades de datos, pruebas de rendimiento para conocer el tiempo

de respuesta de la base de datos ante las consultas realizadas y pruebas de estrés donde se realizan consultas simulando varias conexiones de manera concurrente.

Capítulo 1. Fundamentación teórica.

1.1 Introducción.

En el siguiente capítulo se aborda el estudio realizado sobre el desarrollo de base de datos para sistemas de gestión fiscal, plataforma teórica que sirve de base al desarrollo de este trabajo. Se adentra en otros temas, como son los sistemas gestores de base de datos, el modelo de datos, las características del diseño de base de datos, sus fases y patrones, y se incluyen las herramientas que se utilizan para realizar estas tareas y sus especificaciones. Además se abordan los pasos generales para la normalización, las características de la integridad y seguridad de los datos y se describe de manera introductoria el uso del lenguaje UML para el modelado.

1.2 Base de datos y sistemas gestores de bases de datos.

1.2.1 Base de datos (BD).

El surgimiento de nuevos conceptos, técnicas y herramientas para la gestión de datos, denota hacia donde se dirigen las tendencias de almacenar información. Las bases de datos han sido definidas por especialistas como:

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

Un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada (Date, 2003).

Una estructura de computadora integrada, compartida, que aloja un conjunto de: Datos para el usuario final, es decir, hechos en bruto interesantes para este. Metadatos o datos sobre datos, mediantes los cuales se integran los datos (Peter, et al., 2003).

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

- Una BD representa algún aspecto del mundo real, en ocasiones llamado minimundo o universo de discurso. Las modificaciones del minimundo se reflejan en la BD.
- Una BD es un conjunto de datos lógicamente coherente, con cierto significado inherente. Una colección aleatoria de datos no puede considerarse propiamente una BD.

- Toda BD se diseña, construye y puebla con datos para un propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a dichos usuarios.

Estos conceptos tienen como punto de contacto que las bases de datos son estructuras integradas, persistentes y variables en el tiempo, donde se puede almacenar datos sobre cualquier aspecto del mundo real. Para su manejo e interrelación, se utilizan los sistemas gestores de base de datos.

1.2.2 Sistemas gestores de base de datos (SGBD).

Las herramientas o tecnologías para interactuar con una BD representan la automatización de la gestión de la información almacenada. Hablar de la tecnología de bases de datos es prácticamente lo mismo que hablar de la tecnología de los sistemas de gestión de bases de datos (Orallo, 2002).

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

Los SGBD son una capa intermedia entre los programas que el usuario utiliza y el sistema, por tanto son los encargados de establecer la comunicación entre ellos, deben presentar los siguientes servicios (Zambrano Ramírez, 2008):

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

Entre los gestores más utilizados en el mundo se encuentran Oracle, MySQL, PostgreSQL, Microsoft SQL Server; y existen otros gestores menos reconocidos como: SQLite, Microsoft

Access, DBase, Advantage Database, Fox Pro, Paradox, Open Access, NexusDB, Sybase IQ, Window Base, IBM Informix (CAVSI, 2012).

1.2.3 PostgreSQL 9.1.

PostgreSQL es un sistema gestor de base de datos objeto-relacional, bajo licencia BSD[2]. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

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

1.2.4 Estrategias de indexado.

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

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

- Se pueden indexar los campos que son más utilizados en las búsquedas (los que aparecen en las cláusulas WHERE o JOIN) para mejorar una consulta con el operador (SELECT).
- Los índices se deben crear sobre campos con valores únicos pues funcionan de una mejor manera si el campo no tiene valores duplicados.
- Se deben indexar campos con valores de la menor longitud posible, preferiblemente enteros y si se indexa un campo de texto, se debe evitar hacerlo sobre campos de longitud variable.

[2]BSD: Berkeley Software Distribution License. Licencia de Distribución de Software.

- Por último se recomienda evitar crear índices innecesariamente pues estos se actualizan con cada cambio en la tabla asociada y pueden ralentizar las modificaciones de la misma.

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

La opción estándar de índices del gestor PostgreSQL es la estrategia “B-tree”, Árbol-B (árbol balanceado), 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: <, <=, =, >, =>. En este tipo de estrategia la búsqueda de datos es utilizando las llaves primarias y foráneas, además de las llaves subrogadas, poseen índices de este tipo, lo que implica que cualquier búsqueda que se realice utilizando las llaves se optimizará mediante este método (Postgresql Cuba, 2012).

1.3 Modelo de datos.

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

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

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

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

Las diferentes definiciones aportadas por Carlos Batini (Batini, 2004)y Carlos Alberto García Chávez (García Chavez, 2005), permiten agrupar los modelos de datos en tres grupos, donde se precisa un tipo de estructura en la BD.

- Modelos conceptuales: Refieren a conceptos muy cercanos al modo en que la mayoría de los usuarios percibe los datos. Se usan para especificar la estructura lógica global de la BD y para proporcionar una descripción a nivel más alto de la implementación.
    - Modelo entidad-relación.
    - Modelo semántico: dedicados específicamente a representar la realidad sobre la cual versa la BD.
    - Modelo orientado a objetos: Se basa en objetos, los cuales contienen valores y métodos, entendidos como órdenes que actúan sobre los valores, en niveles de anidamiento. Los objetos se agrupan en clases, relacionándose mediante el envío de mensajes.
- Modelos físicos: Para describir como se almacenan los datos en la computadora, representando informaciones tales como: las estructuras de los registros, el ordenamiento de los registros y las rutas de acceso. Los más conocidos son: el modelo unificador y el modelo memoria de elementos.
- Modelos lógicos: Utilizados para describir datos en el nivel conceptual y el externo, permitiendo especificar las restricciones de los datos. Los modelos lógicos son los más entendibles para el usuario final, aunque ocultan algunos detalles de cómo se almacenan los datos.
    - Modelo relacional.
    - Modelo red o reticular: los datos se representan como colecciones de registros y las relaciones entre los datos se representan mediante conjuntos, que son punteros en la implementación física.
    - Modelo jerárquico: es un tipo de modelo de red, donde cada nodo puede tener un solo padre.

De acuerdo a los conceptos estudiados, se puede resumir que un modelo sirve para llevar a un plano teórico una abstracción del mundo real, para lo cual los modelos de datos se basan en especificar las relaciones y restricciones que deben cumplirse en los datos.

1.3.1 Modelo relacional.

Los estudios de este modelo datan del año 1970, cuando el Dr. Edgar F. Codd, de los laboratorios de investigación de IBM, redactó un artículo en el que presentaba a este modelo y las deficiencias de los modelos jerárquicos y el de red. Los primeros sistemas basados en este modelo fueron apareciendo a finales de los setenta y principios de los ochenta. Entre los que se encuentra el System R de IBM, que se desarrolló para probar la funcionalidad del modelo relacional, proporcionando una implementación de sus estructuras de datos y sus operaciones. Esto incitó dos grandes desarrollos (Marqués, 2001):

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

El modelo relacional de datos es un modelo simple, potente y formal para representar la realidad. También ofrece una base firme para enfocar y analizar formalmente muchos problemas relacionados con la gestión de BD, como el diseño de la BD, la redundancia, la distribución, etcétera. El formalismo y una base matemática son las piedras angulares en el desarrollo de la teoría de las bases de datos relacionales (Batini, 2004).

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

1.3.2 Modelo entidad-relación.

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

El modelo entidad–relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas. Este modelo opera con los conceptos de entidad (estas generalmente se extraen de las especificaciones de requisitos de usuario y en estas especificaciones se buscan los nombres o los sintagmas nominales que se mencionan y representan un objeto o concepto del mundo real) y las relaciones que se establecen entre entidades (Mato García, 2005).

Se popularizó rápidamente como herramienta para representar el modelo conceptual de un sistema de información, creando la clásica separación en las etapas del desarrollo de un sistema de información: diseño conceptual, diseño lógico, diseño físico e implantación (Baizán, 1987).

1.4 Diseño de base de datos.

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

- Mantenerse comunicado con los usuarios.
- Seguir una metodología de diseño.
- Emplear una técnica centrada en datos.
- Poner consideraciones de integridad dentro de los modelos.
- Utilizar diagramas para representar los modelos.
- Utilizar un lenguaje de diseño de BD.

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

1.4.1 Fases del diseño de base de datos.

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

Diseño conceptual:

Como resultado de esta fase se obtiene el esquema conceptual de la BD, donde quedan descritas las especificaciones de requisitos de usuario. Se representan los elementos que pertenecen a una

BD, que reciben el nombre de entidades, las cuales se corresponden con el concepto de clase de la Programación Orientada a Objeto (POO) y donde cada tupla de una futura relación representaría un objeto de la POO.

Diseño lógico:

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

Diseño físico:

En esta fase se transforma la estructura obtenida en la etapa del diseño lógico, con el objetivo de conseguir una mayor eficiencia; además, se completa con aspectos de implementación física que dependerán del SGBD utilizado. Por ejemplo, si se trata de una BD relacional, la transformación de la estructura puede consistir en lo siguiente: tener almacenada alguna relación que sea la combinación de varias relaciones que se han obtenido en la etapa del diseño lógico, partir una relación en varias, añadir algún atributo calculable a una relación. Los aspectos de implementación física que hay que completar consisten normalmente en la elección de estructuras físicas de implementación de las relaciones, la selección del tamaño de las memorias intermedias (buffers) o de las páginas.

1.4.2 Patrones de diseño de bases de datos.

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

Patrón: árbol fuertemente codificado. (Hardcoded tree).

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

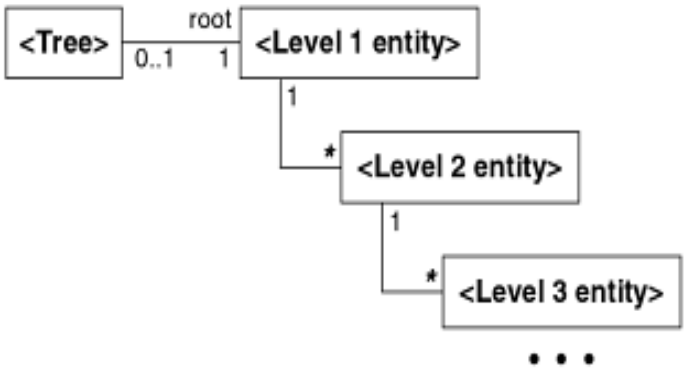


Fig. 1 Árbol fuertemente codificado

Patrón: árbol simple.

Patrón normalmente utilizado cuando el árbol es la representación de una estructura de datos. Los elementos a almacenar son del mismo tipo, es decir, pueden ser almacenados en la misma entidad. No pueden existir ciclos, es decir, un hijo no puede ser su propio padre.

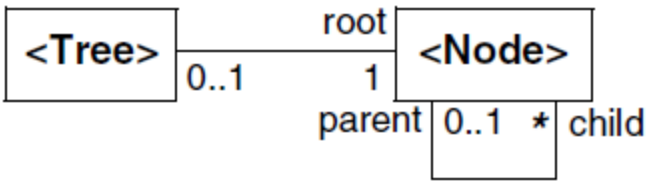


Fig. 2 Árbol simple, notación UML

Patrón: árbol estructurado.

Este modelo es usado cuando se necesitan diferenciar los nodos hojas (leaf), de aquellos que generan una nueva rama (branch), porque ambos tipos de nodos tienen diferentes atributos,

relaciones y/o semántica. No pueden existir ciclos, es decir, un hijo no puede ser su propio padre. La generalización tiene cubrimiento total y exclusivo, cada elemento de la entidad nodo, debe tener su correspondiente elemento en la entidad Leaf o en la entidad Branch.

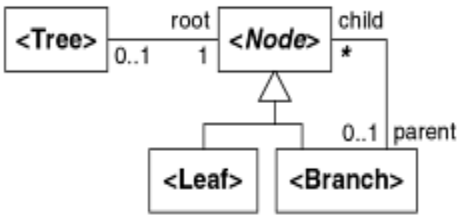


Fig. 3 Árbol estructurado en la notación UML.

Patrón: grafo dirigido simple.

Se utiliza cuando todos los nodos contienen el mismo tipo de datos. Es similar al árbol simple, la diferencia es que en este caso la relación recursiva sobre nodo tiene cardinalidad de muchos a muchos y se por tanto se genera una nueva entidad.

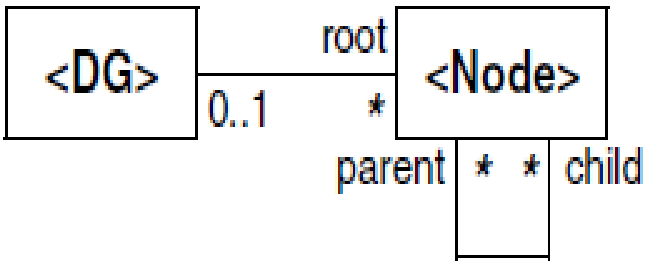


Fig. 4 Grafo dirigido simple en UML

Patrón: grafo dirigido estructurado.

Este modelo es usado cuando se necesita diferenciar los nodos hojas (leaf), de aquellos que generan una nueva rama (branch), porque ambos tipos de nodos tienen diferentes atributos, relaciones y/o semántica. Es similar al árbol estructurado, la diferencia es que en este caso la relación recursiva sobre nodo tiene cardinalidad de muchos a muchos.

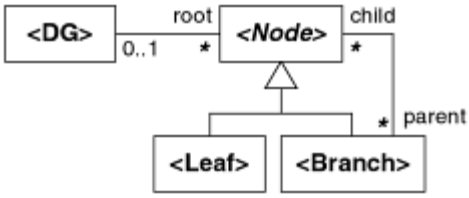


Fig. 5 Grafo dirigido estructurado en UML.

Patrones de flujo de trabajo.

El Lenguaje Unificado de Modelado (UML) tiene una propia forma de representación de flujos de trabajo, de ahí que sea importante tener formas para representar estos flujos en una BD. Existen dos vertientes de este patrón (Blaha, 2010).

- Máquina de estado para un tipo de entidad: Representa los posibles cambios de estado por los que puede atravesar un tipo de entidad.

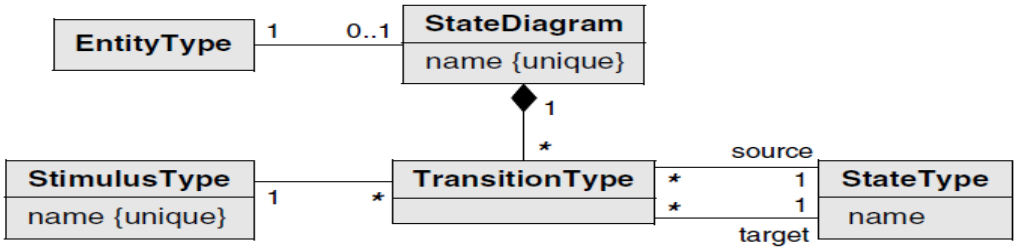


Fig. 6 Máquina de estado para un tipo de entidad representado usando notación UML para clases.

- Máquina de estado para escenarios (control de flujo): Este modelo representa la ocurrencia del cambio de estado en un escenario de una entidad dada, por lo tanto considera el tiempo y la persistencia del mismo en las tablas resultantes. También representa la ocurrencia de un estímulo en una fecha y los estados por los que ha pasado, caracterizados por la fecha de inicio y la fecha fin.

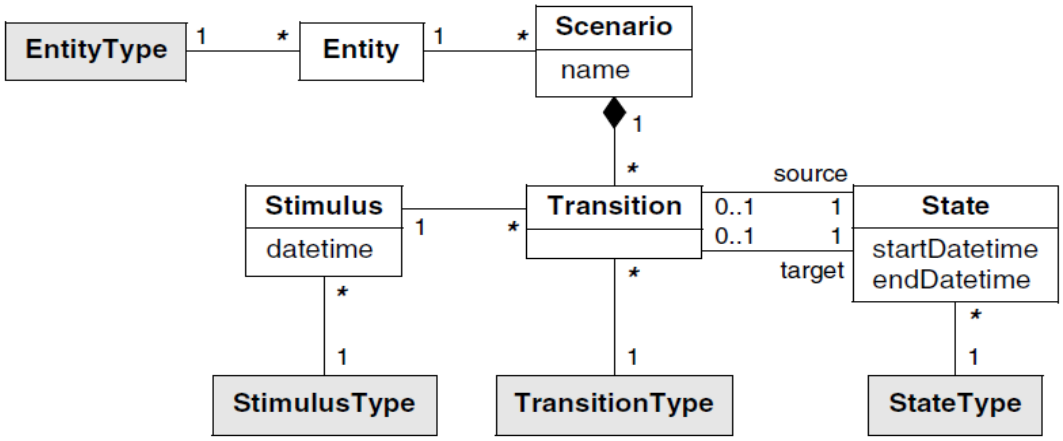


Fig. 7 Máquina de estado para un escenario en notación UML.

Patrón modelo entidad-atributo-valor.

Es la representación de un modelo flexible donde se pueden representar objetos con sus atributos. Es un acercamiento al modelo orientado a objeto representado en el modelo relacional, donde la entidad Class representa las clases, la entidad Attribute representa los atributos de las clases, por su parte la entidad Object representa las instancias de las clases, mientras que la entidad Value representa los valores de cada atributo para cada objeto dado.

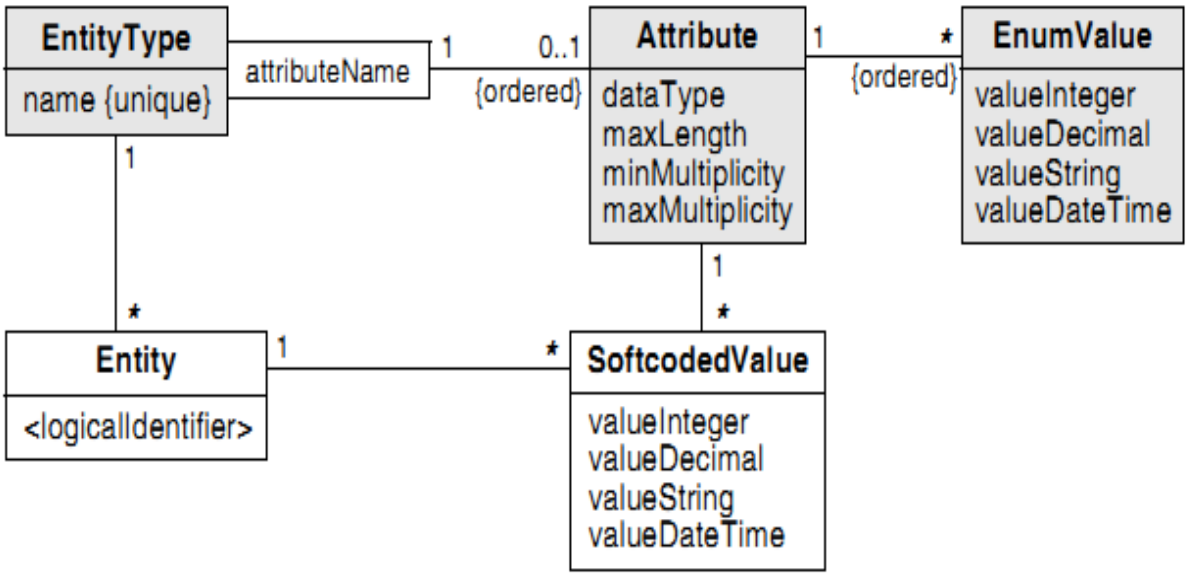


Fig. 8 Patrón Entidad-Atributo-Valor en notación UML.

Patrón de llaves subrogadas.

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

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

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

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

La utilización de este marco de trabajo está fundamentada en cubrir las lagunas de diseño de base de datos que no describen específicamente metodologías, tanto ágiles como tradicionales, a la hora de concebir el desarrollo de software. De ello cuenta que *la metodología Proceso Unificado de Rational (RUP) aborda el desarrollo de base de datos como una tarea de diseño* (UTM, 2012), careciendo de especificaciones necesarias en todo el esfuerzo de desarrollar y mantener una base de datos; por otro lado la Agile Data Method define buenos enfoques prácticos desde el punto de vista ágil, pero el seguimiento sobre cómo se evoluciona no está definido; y la metodología XP trata el tema dejando esta tarea en manos de los programadores, lo cual comúnmente propicia malas decisiones en la actividad.

1.4.4 Características generales de Dbplanning Framenwork.

El framework transita por las fases de Inicio, Desarrollo y Despliegue. Plantea 4 actividades: Modelado de Datos, Configuraciones, Implementación y ADTP (Acceso a Datos, Tuning (optimización por su traducción del inglés) y Prueba), las cuales se desarrollan durante las 3 fases definidas. Las actividades establecen relaciones las cuales no se deben violar.

En la fase de inicio se define la arquitectura del sistema a partir de los requisitos funcionales y no funcionales; en la fase de desarrollo se desarrolla la solución de software y en la fase de despliegue, se realiza la puesta en marcha y soporte del sistema, incluyendo la etapa de pruebas de liberación.

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

La relación entre actividades y fases se muestra en la siguiente imagen.

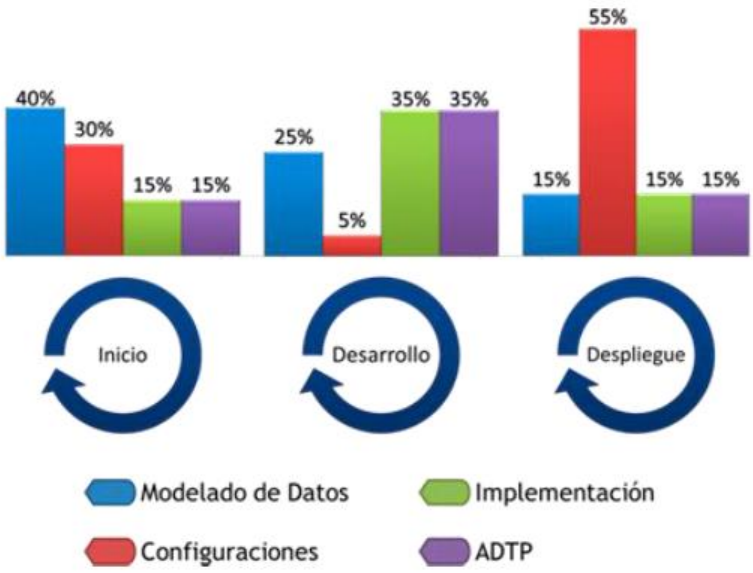


Fig. 9 Relación de fases y actividades de Dbplanning Framework.

De esta forma queda representado el por ciento de esfuerzo de trabajo que se realiza de cada actividad en cada fase, donde cada fase representa un 100 % de esfuerzo de trabajo.

1.5 Normalización de base de datos.

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

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

Las ventajas de aplicar la misma son:

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

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

Primera Forma Normal (1FN):

Describe una relación en la cual (Kroenke, 2003):

- Todos los atributos claves están definidos.
- No existen grupos repetidos en la tabla. En otras palabras, cada intersección de fila/columna puede contener uno y sólo un valor, no un conjunto de valores.
- Todos los atributos son dependientes de la clave primaria.

Segunda Forma Normal (2FN):

Describe una relación en la cual (Peter, et al., 2003):

- Está en 1FN.
- Todos sus atributos no primos tienen dependencia funcional total, respecto a cada una de las claves.
- Aún podemos observar que es posible que una relación que cumpla con 2FN exhiba dependencias transitivas; es decir uno o más atributos pueden ser funcionalmente dependientes de atributos no primos.

Tercera Forma Normal (3FN):

Describe una relación en la cual (Peter, et al., 2003):

- Está en 2FN.
- Ningún atributo no primo depende transitivamente de ninguna clave.

Forma Normal Boyce Codd (FNBC):

Una relación está en FNBC, si todo determinante[3]en ella es una clave candidata (Kroenke, 2003). Evidentemente, si una tabla contiene solamente una clave candidata, la 3FN y la de FNBC son equivalentes. Si se plantea esta proposición de otra manera, la FNBC puede ser violada sólo si la relación contiene más de una clave candidata. La mayoría de los diseñadores consideran la FNBC como un caso especial de la 3FN.

Cuarta Forma Normal (4FN):

La 4NF se asegura de que las dependencias multi-evaluadas independientes estén correcta y eficientemente representadas en un diseño de BD. La 4NF es el siguiente nivel de normalización después de la forma normal de Boyce Codd (Pérez, 2011).

Quinta Forma Normal (5FN):

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

Al realizar el proceso de normalización hasta la 3FN se puede asegurar, en gran medida, la no ocurrencia de los problemas de redundancia en la actualización. La BD propuesta como solución al problema que resuelve este trabajo está normalizada hasta esta fase.

Desnormalización:

Consiste en renunciar a tener la BDR en una forma normal más alta, permitiendo en muchos casos redundancia en la información almacenada, en beneficio de la eficiencia con respecto al tiempo de

[3] Determinante: atributo o conjunto de atributos del cual depende completamente otro atributo (Mato García, 2005).

respuesta de las consultas realizadas sobre dicha información. Si el costo de la obtención de información a partir de los datos relacionados fuera considerablemente más grande que el costo de su mantenimiento entonces surge la necesidad de desnormalizar (Marcelo D. Vinjoy, 2010).

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).

1.6 Integridad en las base de datos.

Cuando los contenidos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes, una puede ser al añadirse datos no válidos a la BD, tales como una secuencia de caracteres no válida para un carnet de identidad, otro caso sería que los cambios se pierdan debido a un error del sistema o fallo en el suministro de energía. De ahí que una de las funciones importantes de un sistema gestor de base de datos relacional es preservar la integridad de sus datos almacenados en la mayor medida posible.

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

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

Datos requeridos:

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

Chequeo de validez:

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

Integridad de entidad:

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

Integridad Referencial:

La integridad referencial es un sistema de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que los registros de tablas relacionadas son válidos y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.

1.7 Seguridad en las base de datos.

Las bases de datos hoy en día son de gran importancia en cada una de las aplicaciones que la requieran, es por esto que la seguridad para su acceso y manejo de la información se restringe en una jerarquía de usuarios. Los SGBD permiten mantener la seguridad mediante el establecimiento de claves para identificar al personal autorizado a utilizar la base de datos. En el caso específico el gestor de bases de datos PostgreSQL 9.1 presenta una serie de características que facilita el trabajo para tener un alto nivel de seguridad.

La seguridad de la base de datos está implementada en varios niveles (Postgresql Cuba, 2012):

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

Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse en base a estos grupos.

1.8 Herramientas del entorno de desarrollo.

Las prestaciones de las herramientas marcan en el desarrollo de un software rapidez, seguridad y estabilidad en los resultados. En el mundo actual las herramientas se miran desde varios puntos de vistas, las privativas presentan en muchos casos, el inconveniente de pago de licencias altamente costosas; las libres, aquellas que no tienen prohibiciones para su ejecución. A estos aspectos se le suma, que su uso sea multiplataforma, capacidad del software de ejecutarse en varios sistemas operativos, y puedan ser de código abierto, las cuales dan la posibilidad de tener acceso al código fuente del software y de este utilizar alguna parte o modificarla para uso propio.

La arquitectura base del proyecto SIGEF II para el desarrollo de la aplicación tiene definido un conjunto de software específicos, estos en su mayoría son software libre y en muchos casos multiplataforma.

1.8.1 Lenguaje de modelado UML 2.0.

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

El lenguaje de modelado unificado (por sus siglas en inglés Unified Modeling Language), permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software. Proporciona una forma estándar de representar los planos de un sistema, y comprende tanto elementos conceptuales, como los procesos de negocio y las funciones del sistema. Permite una comunicación sencilla y rápida entre desarrolladores y clientes del software que se desarrolla y simplifica el proceso complejo de análisis y diseño de software (Pressman, 2005).

El diagrama que mejor describe las relaciones entre clases y la noción de atributos claves que relacionan entre sí las tablas unas con otras, es la extensión de UML, Diagrama de Relación de Entidad (ER diagram), el cual sirve como fiel elemento para seleccionar este lenguaje.

1.8.2 Visual Paradigm 8.0.

Herramienta CASE[4], que permite generación de código y la base de datos a partir de los diagramas UML realizados. Principalmente es utilizado en la modelación de negocio y en el diseño es de gran ayuda a los analistas porque pueden visualizar el flujo central y detallado de

[4] Herramienta Case: (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software (León, 2012).

cada proceso mediante diagramas, además posibilita crear un conjunto amplio de artefactos utilizados con mucha frecuencia durante las disciplinas análisis, diseño, implementación o el despliegue. Esta herramienta permite a múltiples usuarios trabajar sobre el mismo proyecto. Visual Paradigm apoya plenamente la última versión del estándar BPMN y la característica Animación, posibilita a los usuarios ver el flujo de trabajo en acción, dando así una perspectiva más amplia del funcionamiento del negocio (Visual Paradigm, 2013).

1.8.3 Subversion 1.5.

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

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

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

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.).

1.8.4 RapidSVN 0.12.1

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

Características:

- Simple - proporciona una interfaz fácil de usar para las características de Subversión.

- Eficiente - simple para los principiantes pero lo suficientemente flexible como para aumentar la productividad para los usuarios de Subversion con experiencia.
- Portable - se ejecuta en cualquier plataforma en la que Subversion y wxWidgets puede ejecutar: Linux, Windows, Mac OS / X, Solaris, etc.
- Rápido - completamente escrito en C + +.
- Multilingüe - que ha sido traducido a muchos idiomas ya: alemán, francés, italiano, portugués, ruso, ucraniano, chino simplificado, japonés.
- Soporte completo para Unicode.

1.8.5 PGAdmin III 1.14.0

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

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

1.8.6 ORM Doctrine 2.3.2.

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

1.8.7 IDE Netbeans 7.2.

El IDE NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso (Netbeans, 2012).Entre las novedades principales de esta versión destaca mejoras en el soporte para la nueva sintaxis de Java 7, soporte de HTML5, mejoras al editor Java, mejoras en la integración con Web Logic y soporte para PHP 5.3. Otras características que presenta son:

- Tienen soporte para las Base de datos de Oracle.
- El equipo de desarrollo del soporte para PHP de NetBeans 7.0 incluye una interesante característica que permite definir tipos de variables en lo comentarios. Esto es útil, por ejemplo, en aquellas situaciones en los que el código de terminación no reconoce el tipo de objeto que se está tratando.
- Tiene una sección de edición de HTML5.
- Netbeans 7.0 está disponible para usuarios de sistemas operativos Windows, Mac, Linux y Solaris.

1.9 Herramientas para las pruebas.

1.9.1 EMS Data Generator 2005 for PostgreSQL 2.21.

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

- Fácil de usar interfaz de asistente
- Seis idiomas disponibles: Inglés, francés, alemán, italiano, ruso y español.
- Generación de datos a varias tablas de bases de datos diferentes en un host.
- Todos los tipos de datos PostgreSQL, incluyendo Array, direcciones de red y tipos geométricos.
- Los diferentes tipos de generación para cada campo, incluyendo lista, la generación incremental de datos al azar y mucho más.

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

1.9.2 Apache JMeter 2.9.

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

- Se puede cargar y probar diferentes tipos de servidores de rendimiento, (Web-HTTP, HTTPS, JABÓN, Base de datos a través de JDBC, LDAP, JMS, Correo-SMTP (S), POP3 (S) e IMAP (S), Comandos nativos o scripts de Shell)
- Portabilidad completa y pureza Java 100%.
- Permite el muestreo simultáneo de muchas discusiones y toma de muestras simultáneas de diferentes funciones de los grupos de hilos separados.
- Su cuidadoso diseño GUI permite un funcionamiento más rápido y los tiempos más precisos.
- Almacenamiento en caché y el análisis / Reproducción de resultados de las pruebas fuera de línea.
- Samplers conectables permiten capacidades de prueba ilimitadas.
- Varias estadísticas de carga pueden ser elegidos con temporizadores enchufables.
- Análisis de datos y plugins de visualización permiten una gran extensibilidad y personalización.

1.10 Conclusiones parciales.

Al concluir este capítulo se puede evidenciar un estudio de los principales conceptos a conocer sobre el desarrollo de base de datos, diferenciando qué es una base de datos y los sistemas gestores de base de datos. Se estudian los conceptos de los diferentes modelos de datos, de las tareas a realizar en el diseño, de los patrones posibles a usar, así como las características más importantes de la metodología a usar, la normalización y aseguramiento de la integridad y seguridad de los datos. Se realizó una detallada descripción de las herramientas utilizadas en el desarrollo de este trabajo, las cuales fueron definidas por el equipo de trabajo del proyecto.

Capítulo 2. Análisis de la solución propuesta.

2.1 Introducción.

El desarrollo de este capítulo aborda, especificaciones importantes del desarrollo de la base datos, como son las configuraciones de trabajo, normas que se deben aplicar al modelo para hacer más entendible el resultado final. Sobre la solución obtenida se abordan las descripciones sobre los datos, los patrones utilizados, las formas de acceso, la optimización realizada y otras estrategias para mejor manejo de la base de datos en el gestor.

2.2 Configuración del entorno de desarrollo.

Para desarrollar el modelo de datos de los módulos QPD, y Comunes de SIGEF II se utilizó la siguiente configuración.

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

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

Las herramientas de modelado y desarrollo, NetBeans y Visual Paradigm, incluyen opciones de conexión al repositorio de versiones, posibilitando actualizar los cambios desde la misma herramienta cada día. La principal herramienta para guiar este proceso será el RapidSVN. Para mantener un respaldo del trabajo realizado sobre la base de datos, se realiza todos los días una salva de la base de datos a las 4:00pm, de tipo completa.

2.2.1 Usuarios y privilegios.

Para cubrir las necesidades de control de usuario sobre el servidor de BD, se utilizó el patrón RBAC (Control de acceso basado en roles), que implementa el gestor de base de datos. RBAC que funciona como una función de seguridad para controlar el acceso de usuarios a tareas que normalmente están restringidas al superusuario, aplica atributos de seguridad a procesos y usuarios, dividiendo las capacidades de superusuario entre varios administradores. La gestión de derechos de procesos se implementa a través de privilegios (Oracle, 2013).

Para el trabajo con la base de datos del proyecto SIGEF II en el servidor PostgreSQL se crearon diferentes roles. El rol administrador tiene permisos para administrar la base de datos en su totalidad, y el rol usuario que contiene privilegios de actualización, inserción y eliminación de datos.

2.3 Arquitectura de datos.

La Fiscalía General de la República como parte del desarrollo de la aplicación SIGEF II tiene concebida la distribución de datos e información, a lo largo de todo el país. Los servidores de bases de datos estarán situados en cada una de las fiscalías del país. Partirán de un servidor central en la Fiscalía General de la República, el cual se comunicará con las fiscalías provinciales y estas a su vez con las fiscalías municipales en el nivel más bajo. En cada una de las fiscalías se realizará un backup completo de la base de datos mensualmente y se guardaran copias de las bases de datos anteriores en dependencia con los recursos tecnológicos y las políticas de protección de la información de la organización. Estableciendo una política de reemplazo en consonancia con la cantidad de copias establecidas, las cuales van a ser 12 copias (Acosta, 2013).

La distribución de la información se encuentra concebida en la estructura definida por el sistema de réplicas, pensado a aplicarse para la distribución de datos.

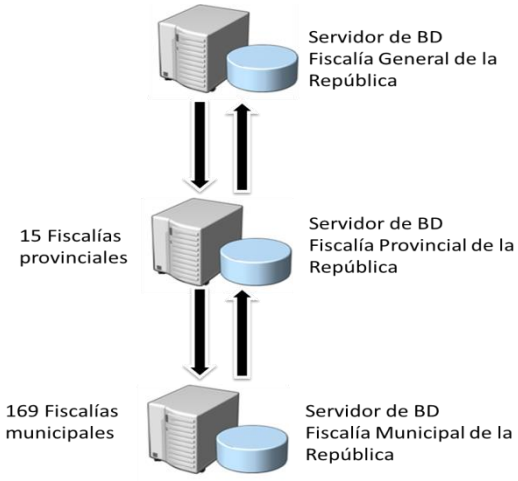


Fig. 10 Distribución de los datos

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

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

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

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

Básicamente existen dos tipos o entornos de réplicas: el de solo lectura o maestro-esclavo (master-slave en inglés) que permite al nodo maestro realizar consultas de lectura/escritura, mientras que

los nodos esclavos solo de lectura; y el otro entorno es el multimaestro (master-master en inglés) donde muchos nodos maestros interactúan entre sí facilitando la sincronización de los cambios.

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

Se utilizará la herramienta Replicador REKO 3.0, desarrollada en la UCI y de código abierto para sistemas distribuidos. Implementada en la plataforma Java Enterpise Edition (JEE), y que provee variadas funcionalidades como (Suarez, 2012):

- Soporte para réplica de datos en ambientes conexión y sin conexión.
- Soporte para réplica entre bases de datos con diferentes estructuras.
- Selección de los datos de réplica ajustada por filtros.
- Soporte para réplica de datos entre gestores diferentes (Postgres-Oracle).
- Monitoreo en tiempo real de los datos de réplica.
- Soporte para Programación del momento de captura y envío de los datos de réplica.

2.4 Nomenclatura y normas para el modelo de base de datos.

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

2.4.1 Generalidades.

- Todos los caracteres para nombrar a atributos y entidades estarán escritos con minúscula.
- Caracteres como tildes, puntos, la ñ no están permitidos.

2.4.2 Nomenclatura para entidades.

- Los nombres concisos y claros con el negocio no excederán los 35 caracteres, además de adaptarse a los nombres definidos en el diagrama de clases.
- Los nomencladores estarán descritos inicialmente por una n y a continuación el nombre identificativo de la entidad especificada en el diagrama de clases.

- Las tablas de datos variados tendrán de prefijo una d y a continuación el nombre identificativo de la entidad, especificada en el diagrama de clases.
- Las relaciones de mucho a mucho donde siempre se genera una nueva tabla estará compuesto el nombre por el de las dos tablas separados por una guion bajo.
- Los nombres compuestos se escribirán juntos.

2.4.3 Nomenclatura para atributos.

- Los atributos tendrán nombres identificativos y claros de lo que representan.
- Los nombres no excederán los 35 caracteres.
- Las llaves primarias tendrán como prefijo id seguido de un guion bajo y el nombre de la tabla.
- Para atributos con nombres compuestos la nomenclatura será (prefijo id, para las llaves primarias) separar las palabras con guion bajo, tratando siempre de no superar los 35 caracteres.
- Los nombres de los atributos compuestos estarán separados por un guion bajo.

2.5 Descripción general de los modelos entidad relación.

Los modelos resultantes ofrecieron un total de 65 tablas para el módulo de Comunes, Anexo 1, de ellas 38 son tablas de datos y 27 son nomencladores. Mientras que para modelo del módulo QPD, Anexo 2, se identificaron un total de 45 tablas, de las cuales 33 son tablas de datos y 12 son nomencladoras.

Resultado de modelo para módulo Queja, Peticiones y Denuncias Anexo 2.

2.5.1 Descripción de las principales tablas del módulo Comunes.

En el módulo Comunes las entidades dpersona, ddireccion, dfiscalia y dproceso, son las más relevantes, pues están asociadas a gran parte de la información que se busca almacenar. En las fiscalías se generarían los procesos de los diferentes subsistemas, los cuales a su vez contienen información de alguna persona asociada, que entre sus características estará definida una dirección. A continuación aparecen tablas para describir los atributos, que contienen estas entidades.

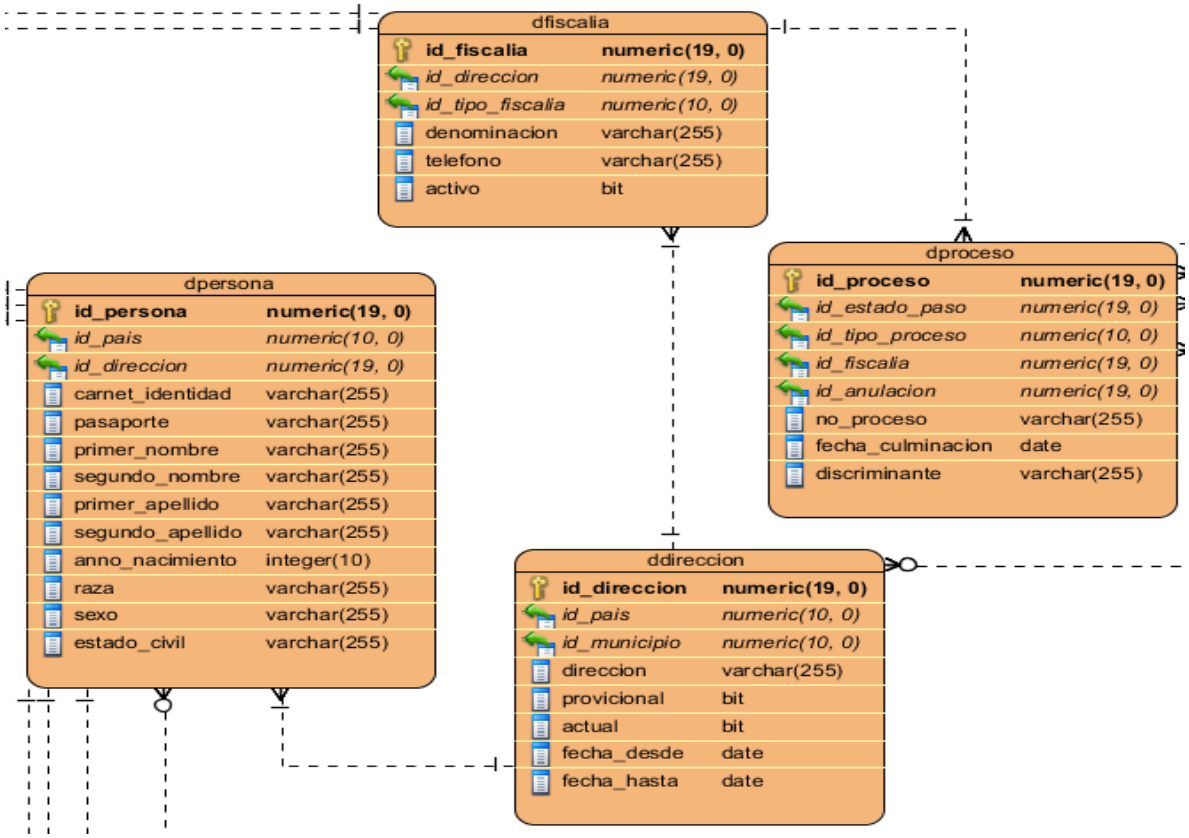


Fig. 11 Tablas principales del módulo Comunes.

Entidad	dpersona	
Descripción	Datos generales de la persona	
Atributos	**Tipo**	**Descripción de los atributos**
id_persona	numeric (19)	Identificador de la persona.
id_pais	numeric (10)	Identificador del país con el que está relacionada la persona.
id_direccion	numeric (19)	Identificador de la dirección que tiene asociada la persona.
carnet_identidad	varchar (11)	Número de carnet de identidad personal.
pasaporte	varchar (50)	Número de pasaporte.
primer_nombre	varchar (255)	Primer nombre de la persona.
segundo_nombre	varchar (255)	Segundo nombre si tiene.
primer_apellido	varchar (255)	Primer apellido de la persona.
segundo_apellido	varchar (255)	Segundo apellido de la persona.
anno_nacimiento	integer (10)	Año de nacimiento de la persona.
raza	varchar (50)	Raza que posee la persona.
sexo	varchar (1)	Sexo que posee la persona.
estado_civil	varchar (50)	Estado civil de la persona.

Tabla 2 Entidad dpersona del módulo Comunes

Entidad		dfiscalia
Descripción		Datos que contiene una fiscalía.
Atributos	**Tipo**	**Descripción de los atributos**
id_fiscalia	numeric(19)	Identificador de la fiscalía.
id_tipo_fiscalia	numeric (10)	Identificador para asociar el tipo de fiscalía.
id_direccion	numeric (19)	Identificador de la dirección que tiene asociada la fiscalía.
denominación	varchar(255)	Denominación completa de la fiscalía.
teléfono	varchar (255)	Teléfono (s) que tiene la fiscalía.
activo	boolean	Define el estado de la fiscalía.

Tabla 3 Entidad dfiscalia del módulo Comunes

Entidad		dproceso
Descripción		Datos que importan en cualquier proceso.
Atributos	**Tipo**	**Descripción de los atributos**
id_proceso	numeric(19)	Identificador del proceso.
id_estado_paso	numeric(19)	Identificador que asocia el estado y paso de un proceso.
id_tipo_proceso	numeric(10)	Identificador asociado al tipo de proceso.
id_fiscalia	numeric(19)	Identificador de la fiscalía asociada al proceso.
id_anulacion	numeric(19)	Identificador para asociar la anulación al proceso.
no_proceso	varchar(255)	Número que tiene el proceso en la fiscalía.
fecha_culminacion	date	Fecha de culminación del proceso
discriminante	varchar(255)	Para uso del ORM Doctrine 2, en el mapeo de la herencia.

Tabla 4 Entidad dproceso del módulo Comunes

Entidad		ddireccion
Descripción		Datos específicos a recoger en una dirección.
Atributos	**Tipo**	**Descripción de los atributos**
id_direccion	numeric(19)	Identificador de la dirección.
id_pais	numeric(10)	Identificador del país asociado a la dirección.
id_municipio	numeric(10)	Identificador del municipio asociado a la dirección.

dirección	varchar(255)	Dirección completa a guardar
provisional	boolean	Verificar si es provisional la dirección.
actual	boolean	Verificar si es actual la dirección.
fecha_desde	date	Fecha desde que tiene la dirección.
fecha_hasta	date	Fecha hasta que estuvo vigente la dirección.

Tabla 5 Entidad ddireccion del módulo Comunes

2.5.2 Descripción de las principales tablas del módulo QPD de CLEP.

En módulo QPD, el proceso principal es la recepción de quejas, de la cual se derivan otras informaciones importantes, como el registro de la queja, el informe emitido sobre esta, el planteamiento que se realiza para argumentar la queja y las restantes informaciones que se derivan o emiten. A continuación se describen las entidades y sus atributos relacionados en el proceso, dqueja, dregistrarqueja, dinforme, dplanteamiento y dprocesarqueja.

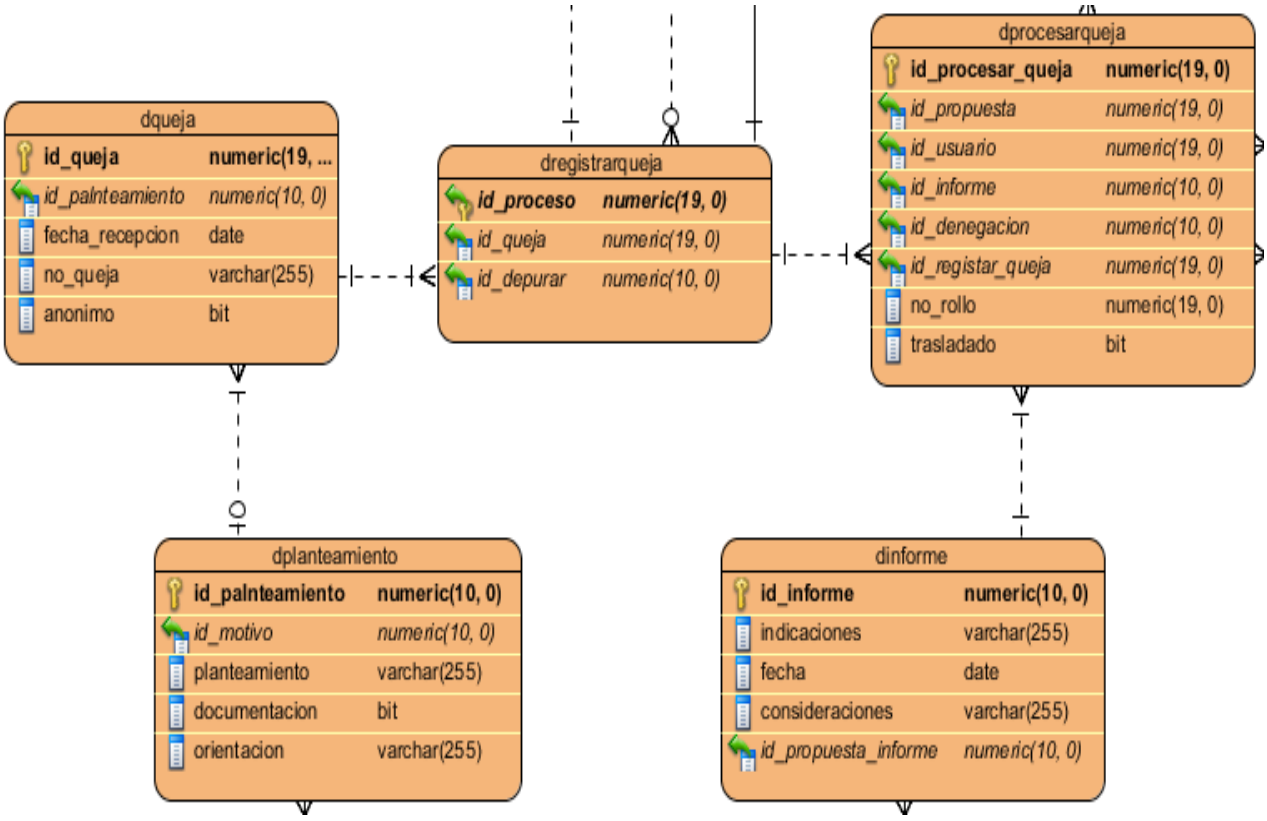


Fig. 12 Tablas principales del módulo QPD de CLEP.

Entidad		dqueja
Descripción		Datos a registrar sobre una queja.
Atributos	**Tipo**	**Descripción de los atributos**
id_queja	numeric(19)	Identificador de la queja a registrar.
id_planteamiento	numeric(10)	Identificador del planteamiento asociado a la queja.
fecha_recepcion	date	Fecha de recepción de la queja.
no_queja	varchar(19)	Número de la queja a registrar.
anonimo	boolean	Afirmación de anonimato de la queja.

Tabla 6 Entidad dqueja del módulo Queja, peticiones y denuncias

Entidad		dprocesarqueja
Descripción		Datos para procesar una queja.
Atributos	**Tipo**	**Descripción de los atributos**
id_procesar_queja	numeric(19)	Identificador del procesamiento de queja que se guarda.
id_propuesta	numeric(19)	Identificador de la propuesta asociada.
id_usuario	numeric(19)	Identificador del usuario asociado al procesamiento.
id_informe	numeric(10)	Identificador del informe asociado al procesamiento.
id_denegacion	numeric(10)	Identificador de la denegación asociada al procesamiento.
id_registrar_queja	numeric(19)	Identificador del registro de la queja.
no_rollo	numeric(19)	Número del rollo al que se asocia el procesamiento.
traslado	boolean	Afirmación de si es un traslado o no.

Tabla 7 Entidad dprocesarqueja del módulo Queja, peticiones y denuncias

Entidad		dregistrarqueja
Descripción		Datos para registrar una queja.
Atributos	**Tipo**	**Descripción de los atributos**
id_proceso	numeric(19)	Identificador del proceso asignado para registrar una queja.
id_queja	numeric(19)	Identificador de la queja a registrar.
id_depurar	numeric(10)	Identificador de depuración de la queja.

Tabla 8 Entidad dregistrarqueja del módulo Queja, peticiones y denuncias

Entidad		dplanteamiento
Descripción		Datos a plantear sobre una queja.
Atributos	**Tipo**	**Descripción de los atributos**
id_planteamiento	numeric(10)	Identificador de cada planteamiento
id_motivo	numeric(10)	Identificador del motivo asociado al planteamiento.
planteamiento	varchar(255)	Planteamiento a recoger.
documentacion	boolean	Afirmación de archivo en el planteamiento.
orientacion	varchar(255)	Orientación que tiene el planteamiento.

Tabla 9 Entidad dplanteamiento del módulo Queja, peticiones y denuncias

Entidad		dinforme
Descripción		Datos que contienen los informes.
Atributos	**Tipo**	**Descripción de los atributos**
id_informe	numeric(10)	Identificador de los informes.
id_propuesta_informe	numeric(10)	Identificador de la propuesta asociada a los informes.
indicaciones	varchar(255)	Indicaciones sobre el informe.
fecha	date	Fecha en que se emite el informe.
consideraciones	varchar(255)	Consideraciones emitidas sobre el informe.

Tabla 10 Entidad dinforme del módulo Queja, peticiones y denuncias

2.5.3 Patrones utilizados en el diseño.

Algunos de los patrones utilizados en el modelo, complementan el objetivo de encontrar formas comunes para solventar problemas de diseño.

Árbol simple.

Para no repetir una entidad con los mismos atributos, y que solo la diferencia sería el identificador asociado de su mismo tipo, fue utilizado el patrón árbol simple sobre la tabla “ndivision” del módulo Comunes.

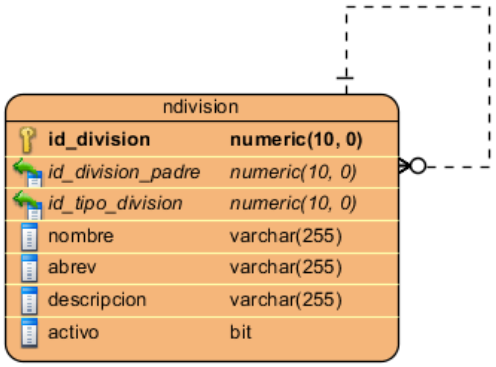


Fig. 13 Entidad ndivision del módulo Comunes.

Llaves subrogadas.

Uno de los patrones más usados en el diseño de base datos, está presente en los modelos resultantes para generar una llave primaria única en cada una de las tablas. La misma que se crea de forma numérica (numeric) de tamaño 19, ejemplo dfiscalia_division del módulo Comunes y dantecedentes en el módulo QPD.

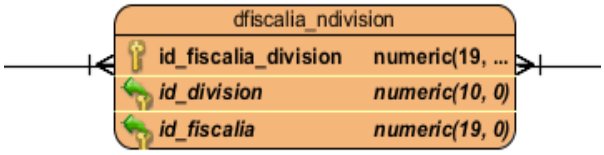


Fig. 14 Entidad dfiscalia_ndivision del módulo Comunes

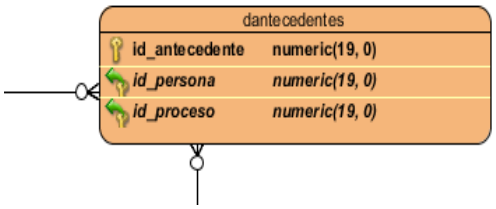


Fig. 15 Entidad dantecedentes del módulo QDP.

2.6 Optimización.

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

2.6.1 Normalización del modelo.

Los modelos obtenidos para los módulos QPD y Comunes se encuentran en primera forma normal pues se puede asegurar que cada tupla contiene exactamente un valor para cada atributo de las

tablas de la base de datos, o sea, no existen campos multi-evaluados. Además se puede decir que están también en segunda forma normal, pues se encuentran en primera forma normal y todos los atributos que no son claves en las tablas, dependen totalmente de la clave primaria. Además está en tercera forma normal, al encontrarse en segunda forma normal y no tener dependencias transitivas los atributos no primos.

Algunas tablas se desnormalizaron para lograr flexibilidad en el diseño, la optimización de consultas y mejorar el acceso a los datos. Ejemplo son: dentidafonetica y dpersonafonetica en el módulo Comunes.

2.6.2 Estrategia de indexado utilizada.

Para la optimización de las operaciones de búsqueda sobre la base de datos se definió utilizar la estrategia de indexado estándar del gestor, “B-tree”. Por ofrecer gran variedad de operadores, además de poder ser utilizada sobre los campos más recurridos en las consultas. Como son las llaves primarias y las llaves foráneas.

CREATE INDEX idx_dproceso_id_fiscalia **ON** base.dproceso **USING** btree (id_fiscalia);	**CREATE INDEX** idx_dproceso_id_trabajador **ON** base.dproceso **USING** btree (id_trabajador);	**CREATE INDEX** idx_dproceso_id_anulacion **ON** base.dproceso **USING** btree (id_anulacion);

Tabla 11 Algunos campos indexados en la entidad dproceso

2.6.3 Mantenimiento de la de la base de datos (VACUUM).

Las operaciones realizadas sobre las tablas de una base de datos, tales como eliminar o modificar una fila, derivan a que su contenido previo no será reemplazado, pero quedará marcado como no válido y no se podrá usar, pero así el nuevo dato será insertado en la BD, y este si será accesible. Este tipo de acciones al cabo del tiempo acumulan datos inservibles en el gestor provocando su mal funcionamiento y por lo que es necesario regularmente realizar una recolección de basura, para asegurarse que la BD no contenga demasiados datos sin uso y se afecte el rendimiento (Ulacia., 2011).

VACUUM es un comando de gestor PostgreSQL utilizado con este propósito, aunque no propio del estándar SQL92. Sirve para dos propósitos, como medio para reclamar almacenamiento, y también para recolectar información para el optimizador. Al ponerlo en función abre cada clase en la base de datos, limpia los registros de transacciones ya pasadas y actualiza las estadísticas en los catálogos del sistema. Las estadísticas mantenidas incluyen el número de tuplas y el número de páginas almacenadas en todas las clases. La ejecución periódicamente del comando aumentará la velocidad de la base de datos al procesar las consultas del usuario (Ulacia., 2011).

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

2.7 Acceso a datos.

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

El framework ORM Doctrine 2.3.2, para el mapeo de la base datos, utiliza anotaciones en las clases php que identifican con igual nombre y atributos a sus similares en la base de datos. La utilización de esta vía en la solución deriva, en que *para las aplicaciones desarrolladas con Symfony2, se recomienda utilizar anotaciones en sus aplicaciones siempre que sea posible* (Eguiluz, 2012), pues permite entender con claridad el porqué de los atributos existentes, y reconocer con facilidad su existencia en la base de datos. En la figura 18 se muestra un ejemplo de un fragmento de código en una clase PHP con sus respectivas anotaciones.

Ejemplo:

```php
/**
 * @ORM\Table(name="base.ddireccion")
 * @ORM\Entity(repositoryClass="Adm\BaseBundle\Entity\DireccionRepository")
 */
class Direccion {

    /**
     * @ORM\Column(name="id_direccion", type="decimal", nullable=false, precision=19)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="NONE")
     * @ORM\SequenceGenerator(sequenceName="base.ddireccion_iddireccion_seq", allocationSize=1, initialValue=1)
     */
    private $id_direccion;
```

Fig. 16 Código del ORM Doctrine 2 sobre clases entidades.

Para la manipulación y acceso a los datos, el framework utiliza su propio lenguaje de consulta, Doctrine Query Language, lenguaje de consulta de Doctrine, por sus siglas DQL. La utilización de esta vía, media de una clase repositorio creada a partir de identificar la entidad con más fuerza en algún proceso. El ejemplo de la figura 19 muestra una consulta desde el repositorio de la entidad Antecedentes (AntecedentesRepository.php) para listar todos los antecedentes contenidos en la base de datos.

```php
use Doctrine\ORM\EntityRepository;
use Adm\BaseBundle\Entity\PersonaProceso;
use Adm\BaseBundle\Entity\Proceso;
use Doctrine\ORM\Query\Expr\Join;

class AntecedentesRepository extends EntityRepository {

    public function listarDivisiones($tipo = null) {

        $qb = $this->getEntityManager()->createQueryBuilder();
        $qb->select('d')
                ->from('BaseBundle:Division', 'd')
                ->innerJoin('d.tipo_division', 'td', Join::WITH, 'td.id_tipo_division = :prmTipo')
                ->setParameter('prmTipo', $tipo);
        return $qb->getQuery()->getResult();
    }
```

Fig. 17 Código DQL del ORM Doctrine 2.

2.7.1 Asignación de llaves a través del ORM.

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

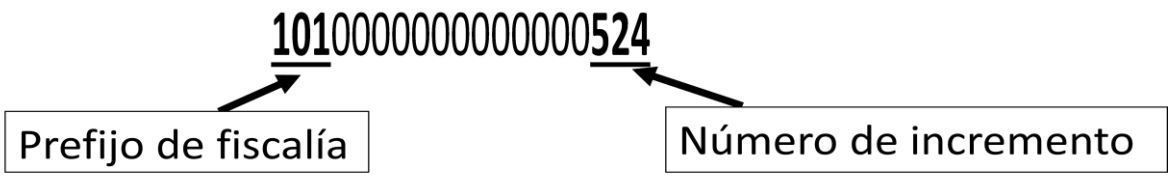


Fig. 18 Identificador generado para una tupla en la entidad dproceso.

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

Método: **generate (EntityManager $em, $entity).**

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

    return $str;
}
```

Fig. 19 Método de asignación de llaves a través del ORM

2.8 Conclusiones parciales.

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

Capítulo 3. Validación y pruebas.

3.1 Introducción.

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

3.2 Validación teórica.

Obtener un modelo encaminado a las necesidades del usuario final, por sí solo no asegura un correcto funcionamiento de la base de datos, además es necesario velar por aspectos importantes como consistencia, integridad y seguridad. El uso de técnicas y reglas desde la aplicación hacia la base de datos para asegurar la información, se convierte en otro de los procesos importante a tener en cuenta en el desarrollo de la base de datos.

3.2.1 Integridad de la base de datos.

Las reglas establecidas para velar por la consistencia de los datos requeridos en las tablas, chequear la validez y unicidad de determinados atributos, incluyendo las restricciones impuestas para los tipos de llaves primarias y foráneas, fueron determinadas de la siguiente forma:

Datos requeridos: Se definieron atributos no nulos, con la cláusula (not null), en las tablas por la importancia que contienen a la hora de agregar datos, aunque resulta poco necesario declarar columnas nulas cuando se requiere guardar siempre este tipo de dato. Se puede tomar como ejemplo el campo primer_nombre de la tabla dpersona.

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

Dominio	Tipo de Datos	Campos donde puede aparece
descripción	varchar(255)	primer_nombre, descripcion, primer_apellido
comentario	texto	comentario, fundamentacion, dictamen. recomendaciones

id_tabla	numeric(19)	Para identificadores de tablas de datos.
id_nomenclador	numeric(10)	Para identificadores de tablas nomencladoras.
fecha	date	fecha, fecha_notificacion, fecha_desde, fecha_hasta
bool	boolean	provisional, activo, actual

Tabla 12 Descripción de integridad de dominio

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

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

3.2.2 Transacciones.

El control de transacciones realizado desde la aplicación hacia la base de datos queda delimitado por el framework ORM Doctrine 2.3.2, encargado de velar por las operaciones de escritura (INSERT/UPDATE/DELETE). El mismo ofrece dos formas de realizar esta acción: La primera es ubicándolas en una cola, hasta que se invoca EntityManager->flush () el cual envuelve todos estos cambios en una sola transacción (enfoque implícito); y la otra opción es realizar este control desde un punto de vista propio (enfoque explícito) (Doctrine, 2013). Ejemplos:

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

Fig. 20 Ejemplo de Enfoque Implícito

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

Fig. 21 Ejemplo de Enfoque Explícito.

El tipo de enfoque utilizado para controlar las transacciones en SIGEF II, es el segundo; porque permite definir parámetros específicos sobre los datos a insertar, actualizar o eliminar. Es necesario porque con la desnormalización de algunas tablas en el modelo, pueden existir atributos nulos en algunas entidades, además que permite controlar la conexión de la aplicación hacia la base de datos. El siguiente ejemplo muestra la implementación realizada de este tipo de enfoque en la clase TransaccionIntercetor.php.

```php
class TransaccionInterceptor implements MethodInterceptorInterface {

    private $em;

    function __construct(EntityManager $em) {
        $this->em = $em;
    }

    public function intercept(MethodInvocation $invocation) {
        $this->em->getConnection()->beginTransaction();
        try {
            $rs = $invocation->proceed();
            $this->em->flush();
            $this->em->getConnection()->commit();
            return $rs;
        } catch (\Exception $ex) {
            $this->em->close();
            $this->em->getConnection()->rollBack();
            throw $ex;
        }
    }

}
```

Fig. 22 Ejemplo de Enfoque Explícito, implementado en la solución.

3.3 Validación Funcional.

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

3.3.1 Pruebas de volumen.

Estas pruebas buscan dar una estimación de hasta dónde se puede llegar cargando el sistema antes de que sea inutilizable (GlobeTesting, 2013). Esta se convierte en una de las pruebas más comunes para sistemas que utilizan base de datos, las cuales estarán dirigidas a la cantidad máxima de datos que se pueda almacenar.

Para tener un medidor de la cantidad de datos que se generan en las fiscalías, se utiliza la tabla dproceso del modelo del módulo Comunes, pues la gran mayoría de las actividades de inserción de datos en algún proceso realiza alguna referencia a esta entidad. Se estima un total aproximado de 200000 procesos anuales entre todas las fiscalías del país, valor que se puede tomar como probable a existir durante los próximos 5 años. Teniendo en cuenta el posible crecimiento descrito anteriormente el total de tuplas a insertar en prueba fue de un 1000000.

El ambiente de este tipo de prueba se restringe a condiciones inferiores a la existente en la Fiscalía General de la República. Utilizándose una máquina con microprocesador Intel(R) Core(TM) 2 Duo, a 2.20 GHz, que utiliza a su vez una Memoria de Acceso Aleatorio (RAM por sus siglas en inglés) de 1 gigabyte. La herramienta de generación de datos aleatorios EMS Data Generator 2005, permite generar la cantidad de datos aleatorios necesarios para la prueba, prestando la posibilidad de configurar los tipos de datos, para que los resultados se asemejen a lo esperado por el equipo de desarrollo.

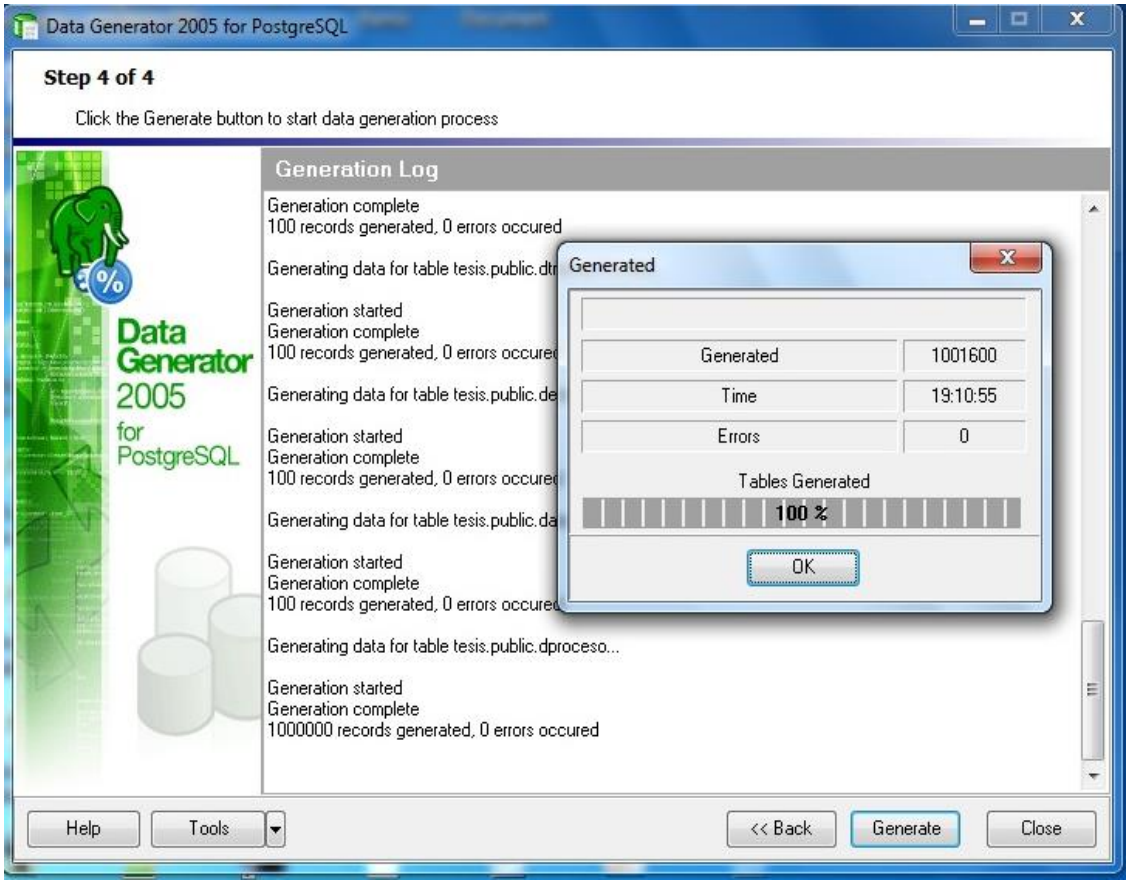


Fig. 23 Respuesta de prueba de volumen

El resultado de la prueba arroja, que la base datos puede soportar la carga de datos que se genera en las fiscalías del país, sin presentar límites de capacidad, desbordamiento de columnas, atributos o tipos de datos. Sirviendo de apoyo las característica de almacenamiento del gestor que se muestran en la siguiente tabla (Gordon, 2012).

Límites	Valor
Tamaño máximo de la base datos	Ilimitado
Tamaño máximo de la tabla	32 Terabyte
Tamaño máximo de la fila	1.6 Terabyte
Tamaño máximo del campo	1 Gigabyte
Número máximo de filas por tabla	Ilimitado

Número máximo de columnas por tabla	250-1600 de acuerdo al tipo de dato de la columna
Número máximo de índices por tabla	Ilimitado

Tabla 13 Capacidad de Almacenamiento de PostgreSQL 9.1

3.3.2 Pruebas de rendimiento.

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

Es preciso definir un rango de tiempos aceptables para las respuestas de las consultas, el cual queda definido de la siguiente forma.

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

Tabla 14 Tiempos para consultas

Para la realización de las consultas se tomó como carga inicial parte de la información generada en la prueba de volumen, quedando de la siguiente forma:

Entidad	Cantidad de tuplas
dfiscalia	170
dproceso	25000
dpersona	10000

ddireccion	10000
dtrabajadorfiscalia	500

Tabla 15 Datos cargados para prueba de rendimiento

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

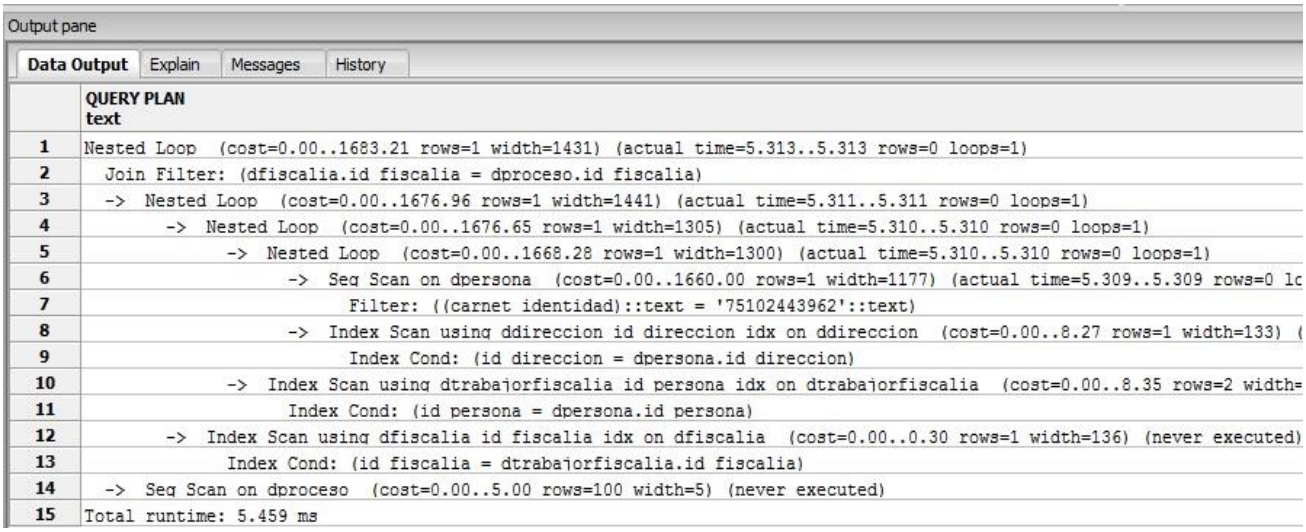
Consulta:

```
EXPLAIN ANALYZE
        SELECT dpersona.id_persona, ddireccion.direccion, dpersona.primer_nombre,
               dpersona.pasaporte, dpersona.carnet_identidad, dpersona.segundo_nombre,
               dpersona.primer_apellido, dpersona.segundo_apellido,
               dpersona.anno_nacimiento, dpersona.raza, dpersona.sexo, dpersona.estado_civil,
               dfiscalia.denominacion
        FROM   public.ddireccion, public.dpersona, public.dproceso, public.dfiscalia,
               public.dtrabajorfiscalia
        WHERE
               ddireccion.id_direccion = dpersona.id_direccion AND
               dfiscalia.id_fiscalia = dproceso.id_fiscalia AND
               dfiscalia.id_fiscalia = dtrabajorfiscalia.id_fiscalia AND
               dtrabajorfiscalia.id_persona = dpersona.id_persona AND
               dpersona.carnet_identidad= '75102443962';
```

Fig. 24 Consulta a realizar como prueba

Resultados:

Ambiente sin indexado. Tiempo: 5.459 ms.



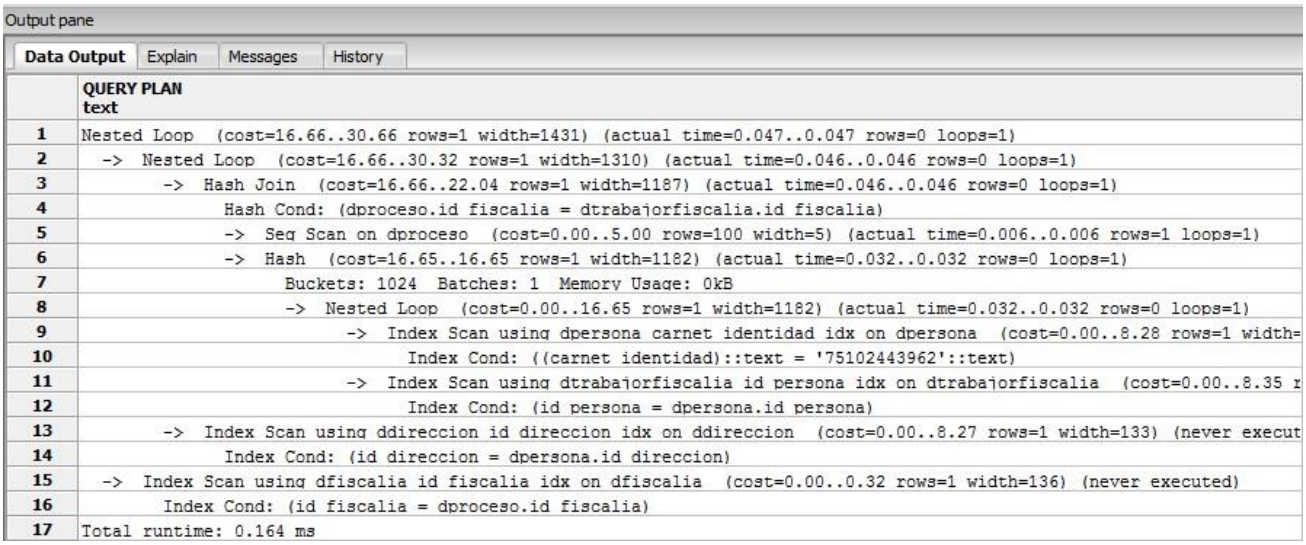
Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Nested Loop (cost=0.00..1683.21 rows=1 width=1431) (actual time=5.313..5.313 rows=0 loops=1)
2	Join Filter: (dfiscalia.id fiscalia = dproceso.id fiscalia)
3	-> Nested Loop (cost=0.00..1676.96 rows=1 width=1441) (actual time=5.311..5.311 rows=0 loops=1)
4	-> Nested Loop (cost=0.00..1676.65 rows=1 width=1305) (actual time=5.310..5.310 rows=0 loops=1)
5	-> Nested Loop (cost=0.00..1668.28 rows=1 width=1300) (actual time=5.310..5.310 rows=0 loops=1)
6	-> Seq Scan on dpersona (cost=0.00..1660.00 rows=1 width=1177) (actual time=5.309..5.309 rows=0 lc
7	Filter: ((carnet identidad)::text = '75102443962'::text)
8	-> Index Scan using ddireccion id direccion idx on ddireccion (cost=0.00..8.27 rows=1 width=133) (
9	Index Cond: (id direccion = dpersona.id direccion)
10	-> Index Scan using dtrabajorfiscalia id persona idx on dtrabajorfiscalia (cost=0.00..8.35 rows=2 width=
11	Index Cond: (id persona = dpersona.id persona)
12	-> Index Scan using dfiscalia id fiscalia idx on dfiscalia (cost=0.00..0.30 rows=1 width=136) (never executed)
13	Index Cond: (id fiscalia = dtrabajorfiscalia.id fiscalia)
14	-> Seq Scan on dproceso (cost=0.00..5.00 rows=100 width=5) (never executed)
15	Total runtime: 5.459 ms

Fig. 25 Resultado de consulta sin indexado.

Ambiente con campos de llaves primarias, llaves foráneas, y la columna carnet_identidad, indexada. Tiempo: 0.164 ms.



Output pane

Data Output | Explain | Messages | History

	QUERY PLAN text
1	Nested Loop (cost=16.66..30.66 rows=1 width=1431) (actual time=0.047..0.047 rows=0 loops=1)
2	-> Nested Loop (cost=16.66..30.32 rows=1 width=1310) (actual time=0.046..0.046 rows=0 loops=1)
3	-> Hash Join (cost=16.66..22.04 rows=1 width=1187) (actual time=0.046..0.046 rows=0 loops=1)
4	Hash Cond: (dproceso.id fiscalia = dtrabajorfiscalia.id fiscalia)
5	-> Seq Scan on dproceso (cost=0.00..5.00 rows=100 width=5) (actual time=0.006..0.006 rows=1 loops=1)
6	-> Hash (cost=16.65..16.65 rows=1 width=1182) (actual time=0.032..0.032 rows=0 loops=1)
7	Buckets: 1024 Batches: 1 Memory Usage: 0kB
8	-> Nested Loop (cost=0.00..16.65 rows=1 width=1182) (actual time=0.032..0.032 rows=0 loops=1)
9	-> Index Scan using dpersona carnet identidad idx on dpersona (cost=0.00..8.28 rows=1 width=
10	Index Cond: ((carnet identidad)::text = '75102443962'::text)
11	-> Index Scan using dtrabajorfiscalia id persona idx on dtrabajorfiscalia (cost=0.00..8.35 r
12	Index Cond: (id persona = dpersona.id persona)
13	-> Index Scan using ddireccion id direccion idx on ddireccion (cost=0.00..8.27 rows=1 width=133) (never execut
14	Index Cond: (id direccion = dpersona.id direccion)
15	-> Index Scan using dfiscalia id fiscalia idx on dfiscalia (cost=0.00..0.32 rows=1 width=136) (never executed)
16	Index Cond: (id fiscalia = dproceso.id fiscalia)
17	Total runtime: 0.164 ms

Fig. 26 Resultado de consulta con indexado.

3.3.3 Pruebas de carga y estrés.

El objetivo de estas pruebas es obtener datos, sobre la carga del sistema, que ayuden a realizar el dimensionamiento del sistema, generando carga en el sistema mediante la simulación de

concurrencia que se acerque con fiabilidad al esperado en la explotación real. Las pruebas de estrés son uno de los últimos tipos de pruebas que se deben ejecutar, pues por su carácter poco realista, podría darse el caso de que la situación de carga simulada nunca se diera en la vida real. (GlobeTesting, 2013).

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

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

Tabla 16 Cantidad de trabajadores por tipo de fiscalía

De acuerdo a estos datos, las pruebas estarán centradas en cubrir el máximo de trabajadores que concurrentemente puedan estar realizando peticiones a la base de datos desde la aplicación. Tomándose entonces la cantidad posible en las fiscalías provinciales y en la Fiscalía General de la República como casos más críticos, por ser los tipos de entidades donde puede acumularse gran carga de trabajo.

De las dos pruebas definidas a continuación se muestran las propiedades de los hilos de cada una, así como el significado y valor definido para cada uno de ellos.

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

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

Tabla 17 Propiedades de los hilos de las pruebas

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

Etiqueta: Nombre de la etiqueta de muestra.

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

Media: El tiempo promedio de un conjunto de resultados.

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

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

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

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

Error%: Porcentaje de solicitudes con errores.

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

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

Resultados:

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
JDBC Request	125	256	9	1245	1	1290	0.00%	86.1/sec	11.5
Total	125	256	9	1245	1	1290	0.00%	86.1/sec	11.5

Fig. 27 Respuesta de Jmeter Prueba 1.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
JDBC Request	500	327	116	1284	1	1670	0.00%	255.4/sec	34.2
Total	500	327	116	1284	1	1670	0.00%	255.4/sec	34.2

Fig. 28 Respuesta de Jmeter Prueba 2.

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

3.4 Conclusiones parciales.

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

Conclusiones Generales

A partir de los estudios y tareas realizadas en este trabajo se pudo cumplir el objetivo de diseñar e implementar una base de datos para almacenar y gestionar la información de las entidades fiscales de forma segura y con integridad desde la aplicación en desarrollo SIGEF II, para los módulos QPD, y Comunes. Al concluir lo siguiente:

- El estudio realizado de los elementos teóricos, como conceptos, metodologías y herramientas, permitió sentar las bases para llegar a un correcto diseño de base de datos.
- El diseño de la base de datos que se realizó, dio paso a obtener el modelo físico de la base datos como artefacto fundamental.
- Los problemas en el diseño que se presentaron fueron solucionados con la aplicación de diferentes reglas y estrategias.
- Las pruebas realizadas, corroboraran la factibilidad del resultado obtenido, demostrando que la base datos responderá, cuando esté puesta en función, de manera satisfactoria.

Recomendaciones

Los resultados obtenidos en el presente trabajo contribuyen un aporte, por lo que se recomienda:

- Continuar utilizando la estrategia de trabajo utilizada en los siguientes modelos a obtener de los restantes módulos del proyecto SIGEF II.
- Realizar un trabajo regular de soporte y mantenimiento de la solución obtenida, para mejorar su funcionamiento.
- Recurrir al modelo de trabajo utilizado para proyectos similares.

Bibliografía

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

Baizán, María Covadonga Fernández. 1987. *El modelo racional de datos: De los fundamentos a los modelos deductivos.* s.l. : Ediciones Díaz de Santos, 1987.

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

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

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

CAVSI. 2012. Computer Audio video Systems Integrator. [En línea] 2012. [Citado el: 04 de 12 de 2012.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

Dataprix. 2013. Dataprix. *Dataprix.* [En línea] 2013. [Citado el: 15 de enero de 2012.] http://www.dataprix.com/11-etapas-diseno-bases-datos.

Date, Dr Christopher J. 2003. *Sistemas de Bases de Datos.* La Habana : Felix Varela, 2003.

Doctrine. 2013. Doctrine Project. [En línea] 2013. [Citado el: 9 de 4 de 2013.] http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/transactions-and-concurrency.html.

Eguiluz, Javier. 2012. Comunidad PHP UCI. [En línea] 2012. [Citado el: 15 de 02 de 2013.] https://php.uci.cu/downloads.php?cat_id=21&file_id=343.

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

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

García Chavez, Carlos Alberto. 2005. mailxmail.com. [En línea] 2005. [Citado el: 22 de noviembre de 2012.] http://www.emagister.com/curso-diseno-base-datos-relacionales/sistemas-bases-datos.

GlobeTesting. 2013. GlobeTesting. [En línea] 2013. [Citado el: 23 de 04 de 2013.] http://www.globetesting.com/pruebas-de-rendimiento/.

Gordon, Salcedo Diego Feernando. 2012. UNIVERSIDAD CENTRAL DEL ECUADOR. [En línea] 08 de 2012. [Citado el: 03 de 05 de 2013.] http://www.dspace.uce.edu.ec/bitstream/25000/318/1/T-UCE-0011-16.pdf.

ICT. 2012. Interactive and Coorporative Technologies Lab. [En línea] 2012. [Citado el: 20 de noviembre de 2012.] http://ict.udlap.mx/people/carlos/is341/bases02.html.

Jmeter, Apache. 2013. The Apache Software Fundation. [En línea] 22 de 01 de 2013. www.apache.org.

Kroenke, David M. 2003. *Procesamiento de bases de datos: fundamentos, diseño e implementación Octava Edición.* 2003.

León, Eduardo. 2012. Scridb.com. [En línea] 2012. [Citado el: 5 de 12 de 2012.] http://es.scribd.com/doc/36636137/Tutorial-Visual-Paradigm.

Linux Six Blog. 2012. Linux Six Blog. [En línea] 2012. [Citado el: 5 de diciembre de 2012.] http://linuxsix.blogspot.com/2011/10/rapidsvn-svn-en-linux.html.

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

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

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

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

Marqués, Mercedes. 2001. *Apuntes de Ficheros y Bases de Datos.* 2001.

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

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

Netbeans. 2012. Netbeans. [En línea] 2012. [Citado el: 12 de enero de 2013.] http://netbeans.org..

Oracle. 2013. Oracle. *Guía de administración del sistema: servicios de seguridad .* [En línea] 22 de 01 de 2013. http://www.oracle.com/technology/documentation/index.html.

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

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

Pérez, Ing. Lennin Caro. 2013. Tu base de datos libre. [En línea] 2013. [Citado el: 28 de 04 de 2013.] http://tubasededatoslibre.org/site/wp-content/uploads/2012/05/PonenciaLCaro.pdf.

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

Peter, Rob y Carlos, Coronel. 2003. *Sistemas de base de datos.* 2003.

Postgresql Cuba. 2012. Postgresql Cuba, Comunidad Tecnica. [En línea] 2012. [Citado el: 2 de diciembre de 2012.] http://postgresql.uci.cu/node/63.

—. 2012. SVN Book. [En línea] 2012. [Citado el: 15 de enero de 2013.] http://svnbook.spears.at/nightly/es/svn-ch-1-sect-3.html.

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

Quiroz, Javier. 2003. El modelo relacional de bases de datos. *Boletín de Política Informática.* 2003, 6.

Suarez, Eivys Hernández. 2012. Bilblioteca de la Universidad de las Ciencias Informáticas. [En línea] 2012. [Citado el: 02 de 05 de 2013.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/4303/1/UCIENCIA-2012-T50-P564-Ponencia-2395.pdf.

Testhouse. 2013. Testhouse. [En línea] 2013. [Citado el: 22 de 04 de 2013.] http://www.es.testhouse.net/pruebas-de-rendimiento/.

Ubuntu, Guía. 2008. Portal-Guía Ubuntu. [En línea] 2008. [Citado el: 9 de diciembre de 2012.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

UCI. 2011. www.uci.cu. *Portal de la Universidad de las Ciencias Informáticas.* [En línea] 2011. [Citado el: 12 de noviembre de 2012.] www.uci.cu.

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

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

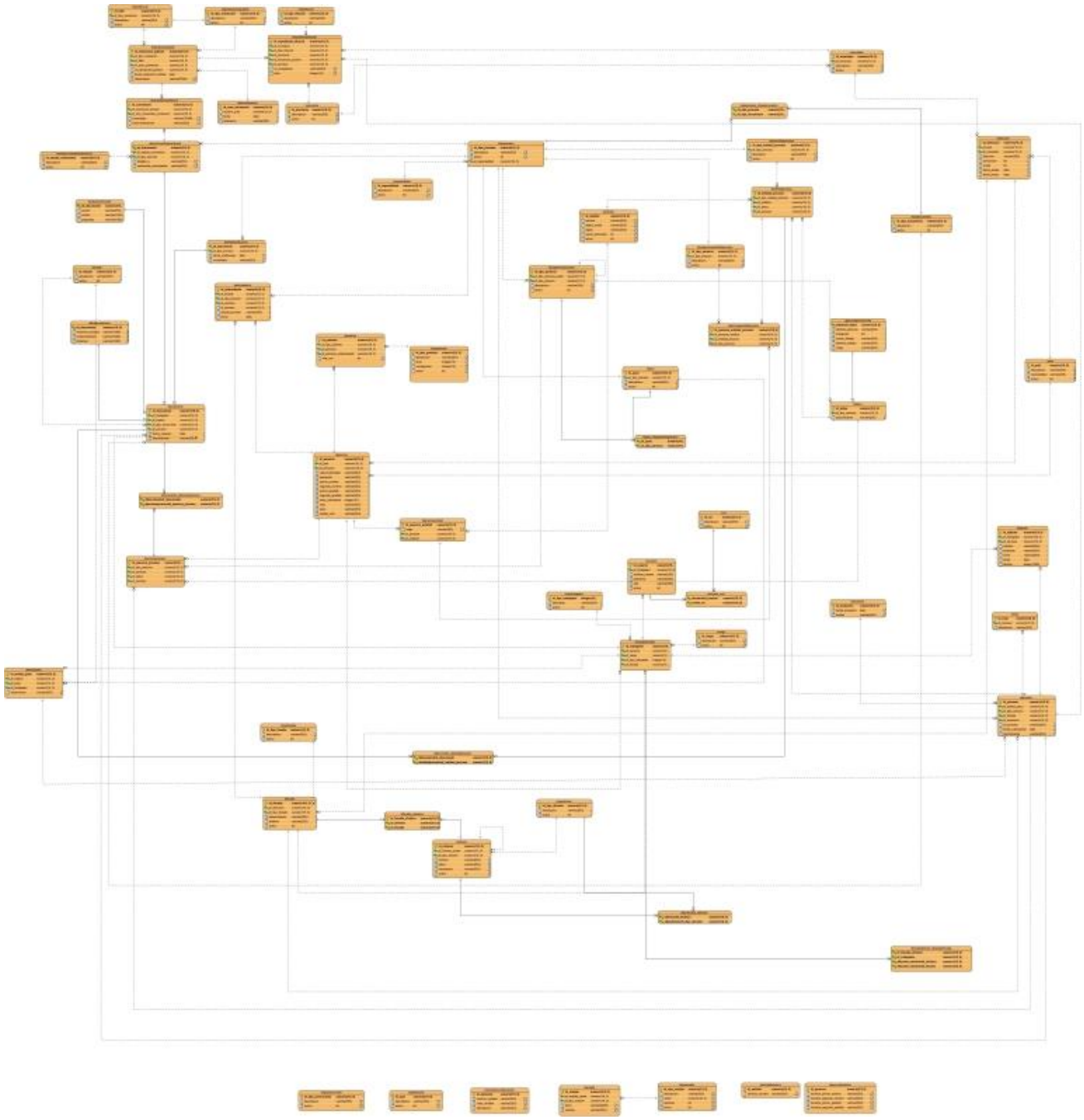
UTM. 2012. Universidad Tecnologica de la Mixteca. [En línea] 2012. [Citado el: 8 de diciembre de 2012.] http://www.utm.mx/~caff/doc/El%20Proceso%20Unificado%20Rational.pdf.

Visual Paradigm. 2013. Visual Paradigm. [En línea] 2013. [Citado el: 14 de enero de 2013.] http://www.visual-paradigm.com/aboutus/10reasons.jsp.

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

Anexos

Anexo 1. Modelo físico de base de datos del módulo Comunes.



Anexo 2. Modelo físico de base de datos del módulo Queja, peticiones y denuncias.