Universidad de las Ciencias Informáticas

MODELO LÓGICO Y FÍSICO DE LA BASE DE DATOS CORRESPONDIENTE A LOS MÓDULOS DE INVESTIGACIÓN CRIMINALÍSTICA Y ESTADÍSTICA DEL PROYECTO CICPC

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

Autor

Arnoldo Domínguez Leyva

Tutor

Ing. Rafael Yordanis Rodríguez Montero

Ciudad de La Habana. Junio de 2008

¨Año 50 de la Revolución¨

DECLARACIÓN DE AUTORÍA

Por este medio declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo.
Para que así conste firmo la presente a los ____ días del mes de _____ del 2008.

_____ _____

Firma del Autor Firma del Tutor

“Ninguna ciencia, en cuanto a ciencia, engaña; el engaño está en quien no sabe.”

Miguel de Cervantes

A Fidel Castro y a la Revolución Cubana por darme la oportunidad de estudiar en una universidad de excelencia.

A mis padres queridos por formarme y por guiarme por el camino correcto.

A mi abuelo Pepito por apoyarme en todo.

Le agradezco a mi novia por estar siempre junto a mí, comprenderme y amarme tanto.

Agradezco a mi familia por darme su cariño y apoyarme en todas mis acciones.

A mi tutor por guiarme en la realización de mi tesis.

Agradezco a mis compañeros por su apoyo incondicional durante todo el proceso de formación académica y realización de este trabajo de diploma

A mis padres, mi novia y toda mi familia que siempre confiaron en que yo podría lograr mis sueños.

Además a todos los que contribuyeron a formarme como profesional.

En el presente trabajo se describe la propuesta del modelo lógico y físico de la base de datos correspondiente a los módulos de Investigación Criminalística y Estadística del proyecto para el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC). Para hacer posible los objetivos de este trabajo, previamente se realizó un estudio de las herramientas más utilizadas para el diseño de bases de datos y los principales gestores de bases de datos que existen. Basado en el estudio anterior y en la decisión de la dirección del proyecto fueron seleccionadas las herramientas que más se ajustaban a las características del trabajo que se iba a desarrollar. Luego de realizar la elección se describe la solución propuesta y se procede a validar dicha solución analizando la integridad y la redundancia de la información, de igual manera se hicieron pruebas para validar funcionalmente el diseño. Finalmente se hace una valoración del trabajo realizado.

ÍNDICE

INTRODUCCIÓN 1

CAPÍTULO 1. FUNDAMENTACIÓN DEL TEMA 4

INTRODUCCIÓN 4
1.1 BASE DE DATOS. 4
1.2 PRINCIPALES TENDENCIAS RELACIONADAS CON EL DISEÑO DE BASES DE DATOS 5
1.2.1 TUNING DE BASE DATOS. 7
1.3 TIPOS DE BASES DE DATOS. 7
1.3.1 SEGÚN LA VARIABILIDAD DE LOS DATOS ALMACENADOS 8
1.3.2 SEGÚN EL CONTENIDO 8
1.4 MODELOS DE BASES DE DATOS. 9
1.4.1 BASES DE DATOS JERÁRQUICAS 9
1.4.2 BASES DE DATOS DE RED 9
1.4.3 BASES DE DATOS RELACIONALES 9
1.4.4 BASES DE DATOS ORIENTADAS A OBJETOS 10
1.4.5 BASES DE DATOS DOCUMENTALES 11
1.4.6 BASES DE DATOS DEDUCTIVAS 11
1.4.7 GESTIÓN DE BASES DE DATOS DISTRIBUIDA 11
1.5 ANÁLISIS Y DESCRIPCIÓN DE LOS PRINCIPALES SISTEMAS GESTORES DE BASES DE DATOS. 11
1.5.1 MICROSOFT SQL SERVER 12
1.5.2 ADABAS 13
1.5.3 POSTGRESQL. 13
1.5.4 ORACLE 10G 14
1.6 METODOLOGÍAS DE DESARROLLO 15
1.6.1 XP (EXTREME PROGRAMING) 16
1.6.2 RATIONAL UNIFIED PROCESS (RUP) 16
1.7 ANÁLISIS Y DESCRIPCIÓN DE HERRAMIENTAS DE MODELADO 17
1.7.1 CASE STUDIO 17
1.7.2 ER/STUDIO 7.0 CON REPOSITORIO 18
1.8 CLIENTES PARA ORACLE. 19
1.8.1 EMS SQL MANAGER PARA ORACLE 19
1.8.2 PL/SQL DEVELOPER 19
1.9 HERRAMIENTAS PARA EL MODELADO DE CLASES PERSISTENTE 21
1.9.1 RATIONAL ROSE 21
1.9.2 VISUAL PARADIGM PARA UML 21
CONCLUSIONES 22

CAPÍTULO 2. ANÁLISIS Y DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA. 23

INTRODUCCIÓN 23
2.1 ESTRATEGIA DE INTEGRACIÓN CON OTROS MÓDULOS Y SISTEMAS. 23
2.2 DESCRIPCIÓN DE LA ARQUITECTURA Y FUNDAMENTACIÓN. 23
2.3 DIAGRAMAS DE CLASES PERSISTENTES. 24
2.4 DESCRIPCIÓN DE LAS CLASES. 26
2.5 DISEÑO DE LA BD. DIAGRAMAS ENTIDAD-RELACIÓN 33
2.5.1 SUBMÓDULO SOLICITUD 34
2.5.2 SUBMÓDULO EXPERTICIA. 36
2.6 DESCRIPCIÓN DE LAS TABLAS DE LA BASE DE DATOS. 38
2.7 DESCRIPCIÓN DE LOS PROCEDIMIENTOS ALMACENADOS. 51
CONCLUSIONES 56

CAPÍTULO 3. VALIDACIÓN DEL DISEÑO REALIZADO. 57

INTRODUCCIÓN 57
3.1 VALIDACIÓN TEÓRICA DEL DISEÑO. 57
3.1.1 INTEGRIDAD DE LOS DATOS. 57
3.1.2 NORMALIZACIÓN DE LA BASE DE DATOS. 60
3.1.3 ANÁLISIS DE REDUNDANCIA DE LA INFORMACIÓN. 62
3.2 VALIDACIÓN FUNCIONAL. 62
3.2.1 HERRAMIENTAS PARA UN LLENADO VOLUMINOSO E INTELIGENTE DE LA BASE DE DATOS. 62
3.2.2 DESCRIPCIÓN DE LAS PRUEBAS 62
3.2.3 ANÁLISIS DE OPTIMIZACIÓN DE CONSULTAS. 66
CONCLUSIONES. 69

CONCLUSIONES GENERALES 70

RECOMENDACIONES 71

REFERENCIAS BIBLIOGRÁFICAS. 72

BIBLIOGRAFÍA 74

GLOSARIO DE TÉRMINOS 75

ANEXOS 77

Índice de Figuras

Figura 1. Fases o Etapas del diseño de bases de datos 5
Figura 2. Diagrama de Clases Persistentes del SubMódulo Solicitud 24
Figura 3. Diagrama de Clases Persistentes del SubMódulo Experticia. 25
Figura 4. Diagrama Entidad – Relación del SubMódulo Solicitud. 34
Figura 5. Diagrama Entidad – Relación del SubMódulo Experticia. 36
Figura 6. Ejemplo de chequeo de validez. 58
Figura 7. Ejemplo de secuencia de DKitMuestra. 59
Figura 8. Ejemplo de secuencia de DPeculiaridadInd. 59

Introducción

La Revolución Bolivariana de Venezuela, desde su triunfo, ha sido incuestionable por la profundidad de sus cambios. Ningún proceso, en ninguna época y en ninguna parte se realizó con menor costo humano. Sin embargo, a pesar de los innumerables cambios que se han venido produciendo en las diferentes esferas como la salud y la educación aún queda mucho por hacer en cuanto a la seguridad ciudadana.

Los ciudadanos se preocupan como nunca antes por este tema. En las calles, la sensación de inseguridad entre las personas se ha incrementado considerablemente. Pero, más que esta situación, preocupa el aumento dramático de la violencia y de la delincuencia en las dos últimas décadas en Venezuela que se observa en las siguientes cifras dadas a conocer por el Ministerio del Poder Popular: en 1990 se registran 29.621 lesiones a nivel nacional y en 2005 la cifra asciende a 33.605, y para el caso de los homicidios (indicador clave y destacado) el incremento resulta sustancial, conociéndose 2.474 homicidios en 1990 y 9.964 en 2005, lo que se traduce en que el número de homicidios que se conocen en Venezuela para 2005 cuadruplica el registro de 1990. (1).

En aras de eliminar esta situación, se ha establecido como parte de la colaboración entre Cuba y Venezuela la automatización de las actividades llevadas a cabo en el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC) de la República Bolivariana de Venezuela con el fin de garantizar la seguridad ciudadana.

El CICPC es la institución que garantiza la eficiencia en la investigación del delito mediante su determinación científica, asegurando el ejercicio de la acción penal que conduzca a una sana administración de justicia. (2). Sus principales acciones están encaminadas a:

- ✓ Optimizar las acciones de investigación criminal tendentes a lograr el esclarecimiento de los hechos delictivos.
- ✓ Dotar al capital humano del CICPC de herramientas, mecanismos logísticos y de infraestructura que garanticen el óptimo desempeño de sus funciones.
- ✓ Fortalecer organizacionalmente la institución y su sinergia con otros organismos de la Administración Pública Nacional y con instituciones privadas.

Dentro de la gran estructura organizativa de esta Institución se encuentra la Coordinación Nacional de Criminalística, que tiene como función fundamental planificar, coordinar y dirigir todos los procesos

técnico-científicos. Esta Coordinación es la encargada de definir las políticas, normas, procedimientos y planes estratégicos que se implementan para el correcto trabajo investigativo, además de diseñar las estrategias que garanticen la cadena de custodia de evidencias físicas, la promoción de la actualización técnico-científica y definir métodos de estandarización en criminalística. Además se encuentra la División de Estadísticas, responsable de elaborar las estadísticas cuantitativas, comparativas y porcentuales requeridas por el CICPC y otros Entes Públicos como el Despacho del Ministro de Seguridad Ciudadana y la Sala de Prensa del Ministerio del Poder Popular para las Relaciones del Interior y Justicia, con el propósito de orientar las políticas y estrategias necesarias para la toma de decisiones.

El CICPC utiliza para la gestión de toda la información ADABAS como Sistema de Gestión de Bases de Datos (SGBD); este no es un sistema relacional trayendo como consecuencia ciertos problemas de actualización y duplicación de los datos. Es válido afirmar que la base de datos actual, presenta algunas limitaciones siendo ineficiente para realizar todos los procesos de almacenamiento que requiere el sistema. Así mismo no almacena toda la información necesaria; además de no permitir realizar análisis estadísticos de gran complejidad; la escalabilidad que ofrece es pobre haciendo muy difícil la incorporación de nuevas funcionalidades al sistema.

Por tanto el **problema** queda definido de la siguiente forma: ¿Cómo realizar un óptimo diseño lógico y físico de la base de datos de los módulos de Investigación Criminalística y Estadística del proyecto CICPC?

El **objeto de estudio** es el proceso de almacenamiento y diseño físico y lógico de Bases de Datos.

El **campo de acción** es el diseño lógico y físico de la base de datos relacional para los módulos de Investigación Criminalística y Estadística.

El **objetivo general** del presente trabajo es realizar el diseño lógico y la implementación del modelo físico de la base de datos de los módulos de Investigación Criminalística y Estadística del proyecto CICPC.

Por lo que se plantea como **idea a defender** que si se crea un óptimo diseño lógico y físico de la base de datos relacional para los módulos de Investigación Criminalística y Estadística que cumpla con los requerimientos necesarios, se logrará la eficiencia y rapidez en el proceso de almacenamiento y utilización de los datos de los módulos de Investigación Criminalística y Estadística de la base de datos del proyecto CICPC.

Las **tareas** que se proponen para dar solución a los objetivos anteriormente mencionados se encuentran:

- ✓ Elaborar el diseño teórico de la investigación.
- ✓ Elaborar la fundamentación teórica de la investigación.
- ✓ Buscar documentación relacionada a los procesos de normalización y de normalización de bases de datos.
- ✓ Consultar otros diseñadores de bases de datos vinculados a proyectos similares en la Universidad.
- ✓ Realizar el diseño lógico de la base de datos de los módulos de Investigación Criminalística y Estadística.
- ✓ Realizar el diseño físico de la base de datos de los módulos de Investigación Criminalística y Estadística.
- ✓ Analizar la integridad de los datos en la base de datos.
- ✓ Definir y realizar pruebas a la base de datos de los módulos de Investigación Criminalística y Estadística del proyecto CICPC.
- ✓ Probar el rendimiento de los procedimientos almacenados.

CAPÍTULO 1. Fundamentación del Tema

Introducción

En el presente capítulo se introducen conceptos importantes para un mejor entendimiento de este trabajo. Se describen las principales tendencias relacionadas con el diseño de bases de datos y algunas herramientas y tecnologías existentes, fundamentando la selección de las empleadas para desarrollar el diseño de la base de datos de los módulos de Investigación Criminalística y Estadística.

1.1Base de Datos.

Las Bases de Datos (BD) y sus tecnologías tienen un impacto decisivo con el creciente uso de las computadoras. Las BD desempeñan un papel crucial en casi todas las áreas de aplicaciones informáticas, como los negocios, la ingeniería, la medicina, el derecho, la educación, la seguridad ciudadana y la bibliotecología.

Una BD es un conjunto de datos relacionados entre sí, además tiene las siguientes propiedades implícitas:

- ✓ Representa algún aspecto del mundo real, llamado minimundo o universo de discurso. Las modificaciones del minimundo se reflejan en la BD.
- ✓ Es un conjunto de datos lógicamente coherentes, con un cierto significado inherente. Una colección aleatoria de datos no puede considerarse propiamente una BD.
- ✓ Una BD se diseña, construye y puebla con datos para propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a distintos usuarios.

O sea, una BD tiene:

- ✓ Una fuente de la cual se derivan los datos.
- ✓ Cierto grado de interacción con los hechos del mundo real.
- ✓ Un público activamente interesado en el contenido de la BD.
- ✓ Su tamaño es variado.
- ✓ Debe ser posible buscar, obtener y actualizar los datos siempre que sea necesario. [1] (3)

[1] Véase también http://teleformacion.uci.cu/course/view.php?id=45

1.2Principales tendencias relacionadas con el diseño de bases de datos

Las últimas décadas se han caracterizado por un fuerte crecimiento en el número e importancia de las aplicaciones de bases de datos, estas son componentes esenciales de los sistemas de información. El diseño de bases de datos se ha convertido en una actividad popular no solo desarrollada por profesionales. Según ha avanzado la tecnología de bases de datos, se han desarrollado las metodologías y técnicas de diseño. Se ha alcanzado una aceptación sobre la descomposición del proceso de diseño de bases de datos en fases, sobre los principales objetivos de cada fase y sobre las técnicas para conseguir estos objetivos. Dentro de las fases o etapas del diseño de bases de datos se encuentran el diseño conceptual, el diseño lógico y por último el diseño físico.

Figura 1. Fases o Etapas del diseño de bases de datos

En el diseño conceptual se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este se le denomina esquema conceptual. Al construirlo, los diseñadores descubren el significado de los datos de la empresa: encuentran entidades, atributos y relaciones. El objetivo es comprender la perspectiva que cada usuario tiene de los datos, la naturaleza de los datos y el uso de los datos a través de las áreas de aplicación. El esquema conceptual se construye utilizando la información que se encuentra en la especificación de

los requisitos de usuario. El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el SGBD que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física. Durante todo el proceso de desarrollo del esquema conceptual éste se prueba y se valida con los requisitos de los usuarios. Este esquema es una fuente de información para el diseño lógico de la base de datos.

El diseño lógico es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo de base de datos específico, independiente del SGBD concreto que se vaya a utilizar y de cualquier otra consideración física. En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario. El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos. Para dar comienzo a esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico. (4)

Otro aspecto a tener en cuenta para el diseño de una base de datos es el proceso de normalización de bases de datos que no es más que aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional.

Las bases de datos relacionales se normalizan para:

- ✓ Evitar la redundancia de los datos.
- ✓ Evitar problemas de actualización de los datos en las tablas.
- ✓ Proteger la integridad de los datos.

Existen seis Formas Normales: Primera Forma Normal(1FN), Segunda Forma Normal(2FN), Tercera Forma Normal(3FN), Forma Normal de Boyce-Codd (FNBC), Cuarta Forma Normal(4FN) y por último

Quinta Forma Normal(5FN). Siendo las cuatro primeras las más utilizadas. Cada una de estas Formas Normales tiene reglas específicas que las tablas de la base de datos deben cumplir.

1.2.1 Tuning de Base Datos.

Tuning (optimización).

Es importante tener en cuenta esta serie de acciones para lograr optimización en las Bases de Datos:

- ✓ Utilizar índices sobre columnas lo más selectivas posibles (aquellas que reducen al máximo el espacio de búsqueda), y en el caso de los índices compuestos, el orden en el que se declaran estas columnas deberá ser de la más a la menos selectiva (siempre que sea posible). En algunos casos es conveniente sustituir índices compuestos por varios índices simples.
- ✓ En el caso de los JOINS entre múltiples tablas, considerar que hay distintas opciones para obtener el mismo resultado, teniendo en cuenta además el coste de las operaciones, considerar en algunos casos alternativas al JOIN (consultas anidadas, cláusula exists subconsulta, etc.). El orden de las tablas en el JOIN es importante.
- ✓ Gestión de las sentencias SQL (del inglés Structured Query Language o Lenguaje Estructurado de Consultas) que contienen vistas. Si una consulta contiene una vista, el optimizador tiene dos formas de actuar: resolver primero la vista y después la consulta o integrar la vista en el texto de la misma. Si se resuelve primero la vista, el resultado completo de la vista se determina en primer lugar y, el resto de las condiciones de la consulta se aplican como filtro. Dependiendo del tamaño de las tablas involucradas puede resultar conveniente hacerlo de un modo u otro. Para ello se debe tener en cuenta que, si una vista contiene una operación de conjunto (GROUP BY, SUM, COUNT o DISTINCT), no podrá ser integrada en la consulta.
- ✓ Se deben optimizar las subconsultas.
- ✓ Limitar los accesos a tablas remotas.
- ✓ Utilizar la cláusula UNION ALL antes que UNION siempre que sea posible.
- ✓ Revisar las consultas periódicamente, puesto que pueden haber dejado de estar construidas de la forma más óptima debido al constante cambio en el tamaño de las tablas, la distribución de los valores, el esquema etc. (5)

1.3Tipos de Bases de Datos.

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio a elegir.

1.3.1 Según la variabilidad de los datos almacenados

1.3.1.1 Bases de Datos Estáticas

Son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

1.3.1.2 Bases de Datos Dinámicas

Son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.

1.3.2 Según el contenido

1.3.2.1 Bases de datos bibliográficas

Solo contienen un representante de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque si no se está en presencia de una base de datos a texto completo o de fuentes primarias. Como su nombre lo indica, el contenido está compuesto por cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

1.3.2.2 Bases de datos de texto completo

Almacenan las fuentes primarias, por ejemplo, todo el contenido de las ediciones de una colección de revistas científicas.

1.3.2.3 Directorios

Un ejemplo son las guías telefónicas en formato electrónico.

Debido a la necesidad de que en el nuevo sistema a automatizar, determinada información almacenada de los módulos de Investigación Criminalística y Estadística se pueda modificar con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta, el tipo de base de datos seleccionada para el desarrollo de este trabajo ha sido Dinámica.

1.4Modelos de Bases de Datos.

Además de la clasificación por la función de las bases de datos, éstas se pueden clasificar también de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos, así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos, y conceptos matemáticos.

1.4.1 Bases de datos jerárquicas

Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés).

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

1.4.2 Bases de datos de red

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; aún así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

1.4.3 Bases de datos relacionales

Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. No tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas".

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red), lo cual tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas en bases de datos relacionales es SQL, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

1.4.4 Bases de datos orientadas a objetos

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- ✓ Encapsulación: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- ✓ Herencia: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- ✓ Polimorfismo: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes: la interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

1.4.5 Bases de datos documentales

Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes. Tesaurus es un sistema de índices optimizado para este tipo de bases de datos.

1.4.6 Bases de datos deductivas

Un sistema de base de datos deductivo, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógica, a raíz de que se basan en lógica matemática.

1.4.7 Gestión de bases de datos distribuida

La base de datos está almacenada en varias computadoras conectadas en red. Surgen debido a la existencia física de organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a distintas universidades, sucursales de tiendas, etc.

Analizando las características de los modelos expuestos anteriormente y teniendo en cuenta que el modelo relacional es el más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente, se decidió utilizar el mismo para la propuesta de solución.

1.5 Análisis y descripción de los principales Sistemas Gestores de Bases de Datos.

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. (6)

Los SGBD se clasifican en varios tipos:

- ✓ SGBD Centralizados.
- ✓ SGBD Cliente-Servidor.
- ✓ SGBD Paralelos.
- ✓ SGBD Distribuidos.

Ventajas y Desventajas de los SGBD.

Ventajas:

- ✓ Eliminan las inconsistencias en los datos.
- ✓ Permiten compartir los mismos datos entre diferentes aplicaciones con distintas necesidades.

- ✓ Ahorran espacio de almacenamiento al no existir redundancia o ser ésta escasa.
- ✓ Mejoran la seguridad de los datos, pues normalmente incorporan mecanismos de seguridad en el propio SGBD.
- ✓ Permiten la creación de entornos de alta disponibilidad, pues con algunos SGBD es posible llegar a disponer de aplicaciones funcionando ininterrumpidamente.
- ✓ Mejora en los servicios de copias de seguridad y de recuperación ante fallos.

Desventajas:

- ✓ Requieren una gran cantidad de espacio en disco y de memoria para trabajar de forma eficiente.
- ✓ Vulnerable a los fallos. El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse.

1.5.1 Microsoft SQL Server

Es un sistema de gestión de bases de datos relacionales (SGBDR) basado en el lenguaje Transact-SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Ventajas:

- ✓ Soporte de transacciones.
- ✓ Escalabilidad, estabilidad y seguridad.
- ✓ Soporta procedimientos almacenados.
- ✓ Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- ✓ Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- ✓ Además permite administrar información de otros servidores de datos.

Desventajas:

- ✓ Licencias con costos altos.
- ✓ El servidor solo está disponible para Sistemas Operativos Windows.

1.5.2 ADABAS

Adabas (Adaptable Database System), es una base de datos jerárquica de alto rendimiento creada por la empresa alemana Software AG, en el año 1969. Actualmente se sigue comercializando bajo la versión Adabas 2006.

Es considerado por algunos como uno de los primeros productos de base de datos disponibles comercialmente.

Adabas es una Base de datos de lista invertida. Ha sido descrita como “no relacional” aunque “casi relacional” en sus características. He aquí algunas diferencias con SGBDR:

- ✓ Archivos, no tablas, como la mayor unidad organizacional.
- ✓ Registros, no filas, como la unidad contenedora dentro de la unidad organizacional.
- ✓ Campos, no columnas, como componentes de una unidad de contenido.
- ✓ No tiene un motor de SQL embebido, por lo que un mecanismo externo de consulta debe ser provisto.
- ✓ La lectura sucia es el modo estándar de operación.
- ✓ Soporta tablas embebidas, por ejemplo en el caso de archivos anidados.

Se ha probado que es muy exitoso en proveer acceso eficiente a los datos y en mantener la integridad en la base de datos. Adabas es ahora ampliamente usado en aplicaciones que requieren muy altos volúmenes de procesamiento de datos o en ambientes de alto procesamiento de transacciones analíticas en línea.

1.5.3 PostgreSQL.

Está considerado el SGBD de código abierto más avanzado del mundo (6). PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales de alto calibre tales como Oracle.

Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos.

Algunas de sus principales características son:

- ✓ Alta concurrencia.
- ✓ Amplia variedad de tipos nativos.
- ✓ PostgreSQL provee nativamente soporte para:
    - Números de precisión arbitraria.
    - Texto de largo ilimitado.
    - Figuras geométricas (con una variedad de funciones asociadas).
    - Direcciones IP (IPv4 e IPv6).
    - Direcciones MAC.
    - Arrays.
- ✓ Claves ajenas también denominadas Llaves Ajenas o Llaves Foráneas.
- ✓ Disparadores (triggers).

Ventajas.

- ✓ Costo.
- ✓ Estabilidad, Confiabilidad.
- ✓ Funcionalidad (es un motor de bases de datos Avanzado).

Desventajas:

- ✓ Consume bastantes recursos y carga con mucha facilidad el sistema.
- ✓ Velocidad de respuesta un poco deficiente al gestionar BD relativamente pequeñas, aunque esta misma velocidad la mantiene al gestionar BD realmente grandes.

1.5.4 Oracle 10g

Oracle es un sistema de gestión de base de datos relacional, fabricado por Oracle Corporation. Es básicamente una herramienta cliente/servidor.

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- ✓ Soporte de transacciones.
- ✓ Estabilidad.
- ✓ Escalabilidad.
- ✓ Es multiplataforma.

Ventajas:

- ✓ Puede ser usado tanto para el manejo de información personal, como para gigantescas bibliotecas multimedia, y corre en laptops, computadoras personales (PC), microcomputadoras.
- ✓ Soporta la mayoría de los lenguajes de computación al igual que 26 idiomas diferentes.
- ✓ Corre automáticamente en más de 80 arquitecturas de hardware y software distinto sin tener la necesidad de cambiar una sola línea de código.
- ✓ Soporta datos alfanuméricos ubicados en las tradicionales "filas y columnas" de las BD, también soporta textos sin estructura, imágenes, audio y video.
- ✓ Incluye mejoras de rendimiento y de utilización de recursos.
- ✓ Está disponible en múltiples plataformas como Windows, Linux, y todas las versiones de Unix.

Desventajas:

- ✓ Es un software propietario, además su elevado precio (la BD más cara) hace que sólo se vea en empresas muy grandes y multinacionales, por norma general.
- ✓ Elevado costo de soporte técnico.

A pesar de estas limitaciones y teniendo en cuenta las ventajas expuestas anteriormente y el criterio del cliente se decidió por la dirección del proyecto utilizar como gestor de bases de datos Oracle.

1.6 Metodologías de desarrollo

La Metodología es el conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de Sistemas de Información.

Proporcionan guías para estimar costos, políticas y procedimientos para garantizar la calidad del software y las descripciones de los roles y responsabilidades de cada uno. En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo.

En la actualidad existen dos grandes grupos de metodologías. Por una parte se tiene las más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas y notaciones que se usarán. Ejemplo de esta es RUP (Proceso Unificado de Desarrollo).

Por otra parte existen las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige

reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo. Ejemplo de esta metodología es XP(del inglés eXtreme Programing o Programación Extrema.)

1.6.1 XP (eXtreme Programing)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo y corto equipo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Características de XP:

- ✓ Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelante en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- ✓ Re fabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- ✓ Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

1.6.2 Rational Unified Process (RUP)

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- ✓ Inicio: El Objetivo en esta fase es determinar la visión del proyecto.
- ✓ Elaboración: En esta fase el objetivo es determinar la arquitectura óptima.
- ✓ Construcción: En esta fase el objetivo es llevar a obtener la capacidad operacional inicial.
- ✓ Transición: El objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, el cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Es aconsejable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta luego en un entregable al cliente. Trayendo como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

- ✓ Actividades: Son los procesos que se llegan a determinar en cada iteración.
- ✓ Trabajadores: Son las personas o entes involucrados en cada proceso.
- ✓ Artefactos: Puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se exige el uso de artefactos, por lo cual es una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Se selecciona la metodología RUP porque es la que más se ajusta a las características del proyecto en cuestión. El rol que se desarrollará es el de diseñador de BD del proyecto (responsable del diseño de la BD), jugando su papel principal en el flujo de trabajo de análisis y diseño en la fase de elaboración.

1.7 Análisis y Descripción de Herramientas de Modelado

1.7.1 CASE Studio

CASE Studio es una herramienta profesional con la que se pueden diseñar bases de datos, la cual facilita la creación de diagramas de relación, modelado de datos y gestión de estructuras. Tiene soporte para trabajar con una amplia variedad de formatos de bases de datos (Oracle, SQL, MySQL, PostgreSQL, Access, etc.) y permite además generar scripts SQL, aplicar procesos de retroingeniería (reverse engineering) a las bases de datos, usar plantillas de diseño personalizables y crear detallados informes en HTML y RTF. A través de los diagramas de relación se podrá tener una visión más clara del contenido y una estructura de la base de datos, facilitando la gestión y mantenimiento de la misma.

Su principal característica, es su potente sistema de ingeniería inversa, o sea, a partir del modelo de tablas llegar al modelo lógico. Permite identificar y estructurar BD ya existentes para poder trabajar con ellas sin problemas.

Su intuitiva interfaz gráfica hace fácil las tareas más complicadas. Mediante Case Studio se pueden realizar diagramas de flujo con muy poco tiempo y esfuerzo. Permite la generación de disparadores (triggers) para realizar validaciones en las entidades que formarán parte de las tablas de la BD.

1.7.2 ER/Studio 7.0 con Repositorio

Es una herramienta CASE para el modelado y diseño de BD, que brinda productividad en su diseño, generación y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información hasta el modelo físico perfeccionado para las características específicas de la BD diseñada, además ER/Studio permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la BD. Genera automáticamente las tablas y miles de líneas de procedimientos almacenados y disparadores para los principales tipos de bases de datos.

ER/Studio hace fácil el diseño de una BD. Los diseñadores de BD sólo apuntan y pulsan un botón para crear un gráfico del modelo E-R (Entidad-Relación) de todos sus requerimientos de datos y capturar las reglas de negocio en un modelo lógico, mostrando todas las entidades, atributos, relaciones, y llaves importantes.

Beneficios de ER/Studio

- ✓ Asegura consistencia, e integración de los datos del proyecto al proporcionar el bosquejo que las IT necesitan para entender, analizar y comunicar la estructura de la base de datos.
- ✓ Mejora la productividad entre los desarrolladores cuando los diseños de la base de datos son divididos, compartidos, y reutilizados.
- ✓ El ambiente gráfico facilita la visualización de la estructura completa, los elementos claves y el diseño optimizado de la base de datos.
- ✓ Ahorra tiempo al acelerar la creación de bases de datos de alta calidad, transaccionales de alto rendimiento y para data warehouse.
- ✓ Mantiene los recursos y mejora la precisión al sincronizar el modelo y la base de datos.

ER/Studio es una herramienta de base de datos que le ayuda a diseñar, generar y mantener aplicaciones de base de datos de calidad y alto rendimiento. Desde un modelo lógico de sus requerimientos de información y reglas del negocio que definen su base de datos, hasta un modelo físico optimizado por las características específicas de su base de datos de destino, ER/Studio le permite visualizar la estructura adecuada, los elementos claves y un diseño optimizado de la base de datos.

Teniendo en cuenta sus ventajas y por su probada eficacia en el modelado de los datos se utilizará como herramienta de modelado el ER/Studio. Permitiendo realizar de forma sencilla el modelo lógico y físico de la base de datos.

1.8 Clientes para Oracle.

1.8.1 EMS SQL Manager para Oracle

EMS SQL Manager para Oracle es una herramienta de alto rendimiento para el desarrollo y la administración de servidor de bases de datos Oracle. Este funciona con cualquier versión de Oracle, desde la 8.1.7 a la 10g, y soporta todos los tipos de datos Oracle incluyendo los tipos nuevos BINARY_FLOAT y BINARY_DOUBLE, y otros. Ofrece una gran variedad de herramientas poderosas para usuarios avanzados, tales como Visual Database Designer (diseñador visual de base de datos), Visual Query Builder (constructor visual de consultas), y un poderoso editor de objetos binarios (BLOB) para satisfacer todas las necesidades. SQL Manager para Oracle cuenta con una nueva y avanzada interfaz gráfica de usuario con un sistema asistente bastante descriptivo, tan claro en su uso que ni un principiante se podrá confundir. (7)

Características principales:

- ✓ Conectarse por medio de un puerto local redirigido a través de un túnel SSH.
- ✓ De fácil administración para todos los objetos Oracle (incluyendo las operaciones: crear/editar/eliminar).
- ✓ Navegación y administración rápida de la base de datos.
- ✓ Herramientas avanzadas de manejo de datos.
- ✓ Efectiva administración de seguridad.
- ✓ Excelentes herramientas visuales y de texto para elaboración de consultas.
- ✓ Impresionantes funcionalidades de exportación e importación de datos.
- ✓ Diseñador de reportes con un claro asistente de elaboración de reportes.
- ✓ Poderoso diseñador visual de bases de datos. (7)

1.8.2 PL/SQL Developer

PL/SQL Developer es un ambiente integrado para el desarrollo, prueba, depuración de errores y optimización de PL/SQL de Oracle almacenado en unidades de programa como paquetes y triggers, entre otros. Este cliente contiene ayuda sensitiva al contexto, descripciones de bases de datos de

objetos, sintaxis resaltada, edición y búsqueda de datos, browser gráfico y muchas otras características que le hacen la vida más fácil al usuario.

PL/SQL Developer es un entorno de desarrollo integrado para desarrollar, probar y optimizar unidades de programa almacenadas de Oracle Databases. Este cliente de Oracle es fácil de usar y pretende ofrecer código y rendimiento de muy buena calidad, puntos claves para el desarrollo de aplicaciones en Oracle.

PL/SQL Developer se enfoca a la facilidad de uso, a la productividad y a la calidad del código, ventajas claves para el desarrollo de aplicaciones en Oracle. Incorpora varias herramientas para hacer el desarrollo del día a día más sencillo. Pueden recompilarse todos los objetos no válidos, buscar textos en el código de las unidades de programa de la base de datos, importar y exportar tablas. (8)

Las principales características del PL/SQL Developer son:

- ✓ Potente editor de PL/SQL: El resaltado de sintaxis, la ayuda de SQL y PL/SQL, la descripción de objetos, el asistente de código, los consejos de compilador, el embellecedor de código PL/SQL, los contenidos de código, los botones de navegación, la navegación por hipervínculos, la librería de macros y otras características sofisticadas hacen que el editor sea el más usado por los diseñadores, incluso por los más exigentes.
- ✓ Depurador integrado: Ofrece todas las características deseables: paso a paso por instrucciones, paso a paso por procedimientos, salir de procedimiento actual, ejecutar hasta que se produzca una excepción, establecer puntos de ruptura, ver y establecer valores de variables.
- ✓ Constructor de consultas: El constructor de consultas gráfico facilita crear nuevas sentencias de selección o modificar las existentes. Es tan sencillo como arrastrar y soltar las tablas y vistas, seleccionar las columnas para las listas de campos, las cláusulas WHERE u ORDER BY, unir las tablas basándose en definiciones de claves externas.
- ✓ Embellecedor de código PL/SQL: El embellecedor de código PL/SQL le permite dar formato a su código SQL y PL/SQL a través de un conjunto de reglas definido por el usuario. El código se puede embellecer automáticamente al compilar, guardar o abrir el archivo. Esta característica incrementará su productividad y mejorará la legibilidad de su código PL/SQL si trabaja en proyectos grandes de desarrollo en equipo.
- ✓ Ventana SQL: La ventana SQL permite introducir cualquier sentencia SQL y ver o editar los resultados.

- ✓ Ventana de comandos: Esta ventana tiene el mismo aspecto y funcionamiento de SQL*Plus, y además tiene un editor integrado de scripts con el correspondiente resaltado de sintaxis.
- ✓ Optimización de rendimiento: Para optimizar el rendimiento de su código SQL y PL/SQL puede utilizar el analizador para ver información de tiempo de la ejecución de cada línea de código PL/SQL. Además puede obtener automáticamente las estadísticas de las sentencias SQL ejecutadas y de los programas PL/SQL.
- ✓ Objetos no PL/SQL: Puede ver, crear y modificar tablas, secuencias, sinónimos, librerías, directorios, usuarios y roles sin escribir ningún código SQL. Simplemente se introduce la información de manera sencilla, y PL/SQL Developer generará el código SQL adecuado para crear o alterar los objetos. (8)

Teniendo en cuenta sus características la herramienta seleccionada fue PL/SQL Developer.

1.9 Herramientas para el Modelado de Clases Persistente

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero.

El uso de estas herramientas brinda varios beneficios, por ejemplo, contribuyen a un intercambio de ideas más efectivo con el cliente y entre los propios integrantes del equipo de desarrollo.

1.9.1 Rational Rose

Es una herramienta para el modelado visual para el análisis y diseño de sistemas basados en objetos.

Rational Rose es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic con funciones configurables de sincronización entre los modelos y el código. Ofrece un lenguaje de modelado común que agiliza la creación del software.

1.9.2 Visual Paradigm para UML

Visual Paradigm para UML (VP-UML) Enterprise Edition apoya las últimas notaciones de UML, ingeniería reversa, generación de UML del código, importa desde Rational Rose. El VP permite su integración con el Eclipse, una de las principales razones por la que fue seleccionada por la Dirección del proyecto para llevar a cabo el desarrollo del mismo. Además puede generar código fuente de programación del diagrama de clases a un proyecto Eclipse. Asimismo, puede utilizar técnicas de

ingeniería inversa para llevar del código fuente a sus diagramas de clases. De esta forma mantiene un alto nivel de sincronización entre el modelo y el código. Corre sobre las plataformas: Windows (98, 2000, XP, or Vista), Linux, Mac OS X y Solaris. Permite a los desarrolladores de un mismo equipo trabajar sobre un mismo proyecto.

Conclusiones

En este capítulo se hizo un estudio de las principales tendencias relacionadas con las bases de datos, de las herramientas a utilizar en el desarrollo de la propuesta de solución así como algunos conceptos que se deben tener en cuenta. Del estudio realizado y en algunos casos por peticiones del cliente, para realizar la propuesta de solución se escogió Oracle como gestor de bases de datos, Rational Unified Process (RUP) como metodología de desarrollo, Visual Paradigm para UML para realizar los diagramas de clases persistentes, ER/Studio 7.0 con Repositorio como herramienta de modelado y por último PL/SQL Developer como cliente de Oracle.

CAPÍTULO 2. Análisis y Descripción de la Solución Propuesta.

Introducción.

En el presente capítulo se describe la solución propuesta, se explica cómo se lleva a cabo la estrategia de integración con otros módulos y sistemas, se realiza una breve descripción de la arquitectura. Además en el mismo se puede ver los diagramas de clases persistentes, la descripción de las clases, los diagramas Entidad – Relación, así como la descripción de sus tablas, también se expondrán los principales procedimientos almacenados utilizados.

2.1Estrategia de Integración con otros módulos y sistemas.

El proyecto CICPC consta de varios módulos que se interrelacionan, dentro de estos están: Investigación Penal, Investigación Forense, Aprehensión, Análisis de Información, Estadísticas, Investigación Criminalística etc.

Investigación Criminalística está compuesto por dos Submódulos: Solicitud y Experticia. Estos Submódulos están estrechamente relacionados ya que en Solicitud están todos los tipos de solicitudes y en Experticia están los informes que le dan respuesta a estas solicitudes.

Sobre la estrategia de integración y desarrollo de la solución podemos decir que los programadores presentan un modelo de clases persistentes por cada caso de uso de donde se parte para diseñar el modelo de datos, para lo cual se crea su modelo lógico y se valida mediante las reglas de la normalización. Este modelo se presenta al equipo de administradores de la base de datos general del proyecto para ser incluido en el mismo.

2.2Descripción de la arquitectura y fundamentación.

La base de datos propuesta contiene cincuenta y ocho tablas en modelo físico, las cuales serán utilizadas por los dos submódulos Solicitud y Experticia del módulo Investigación Criminalística, las principales tablas son: solicitud, solicitud de experticia, solicitud de identificación de individuo, solicitud de experticia criminalística, solicitud de experticia criminalística robo de banco, informe, informe pericial, informe pericial siniestro, informe de experticia de trayectoria balística, informe pericial de reconstrucción de hechos, informe pericial de experticia contable ; las cuales recogen la información de mayor peso en el módulo. Dentro de las cincuenta y ocho tablas hay catorce nomencladores.

El sistema se encuentra estructurado en tres capas siguiendo el patrón de arquitectura layers (capas), por lo que existe una capa de acceso a datos encargada de gestionar la información. Esta capa es generada a partir del Framework Hibernate.

2.3Diagramas de Clases Persistentes.

Las clases persistentes por lo general tienen como origen las clases clasificadas como entidad porque ellas modelan la información y el comportamiento asociado de algún fenómeno o concepto. El diagrama de clase se utiliza para modelar la estructura lógica de la base de datos, con clases representando tablas, y atributos de clase representando columnas. A continuación se muestran los diagramas.

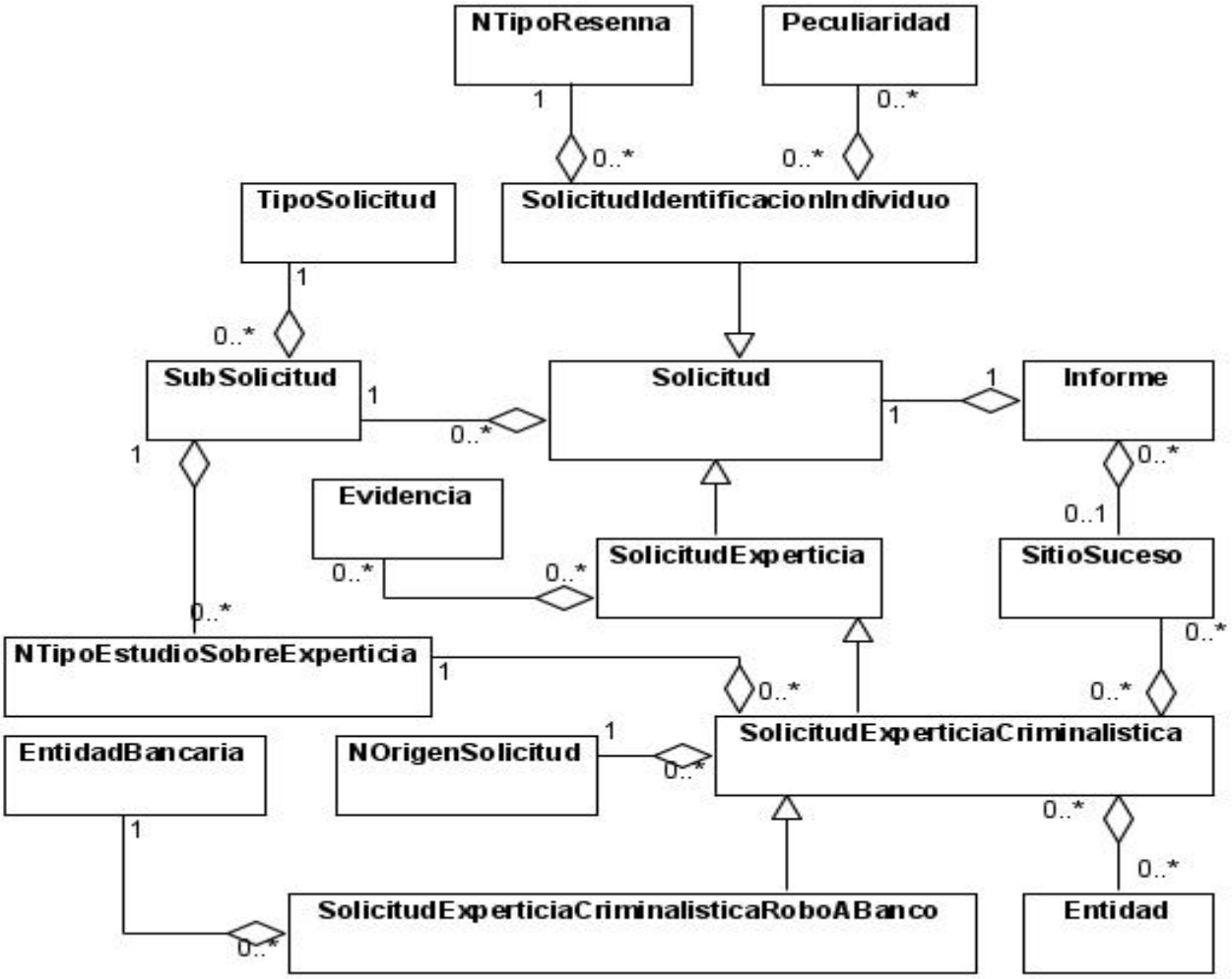

Figura 2. Diagrama de Clases Persistentes del SubMódulo Solicitud

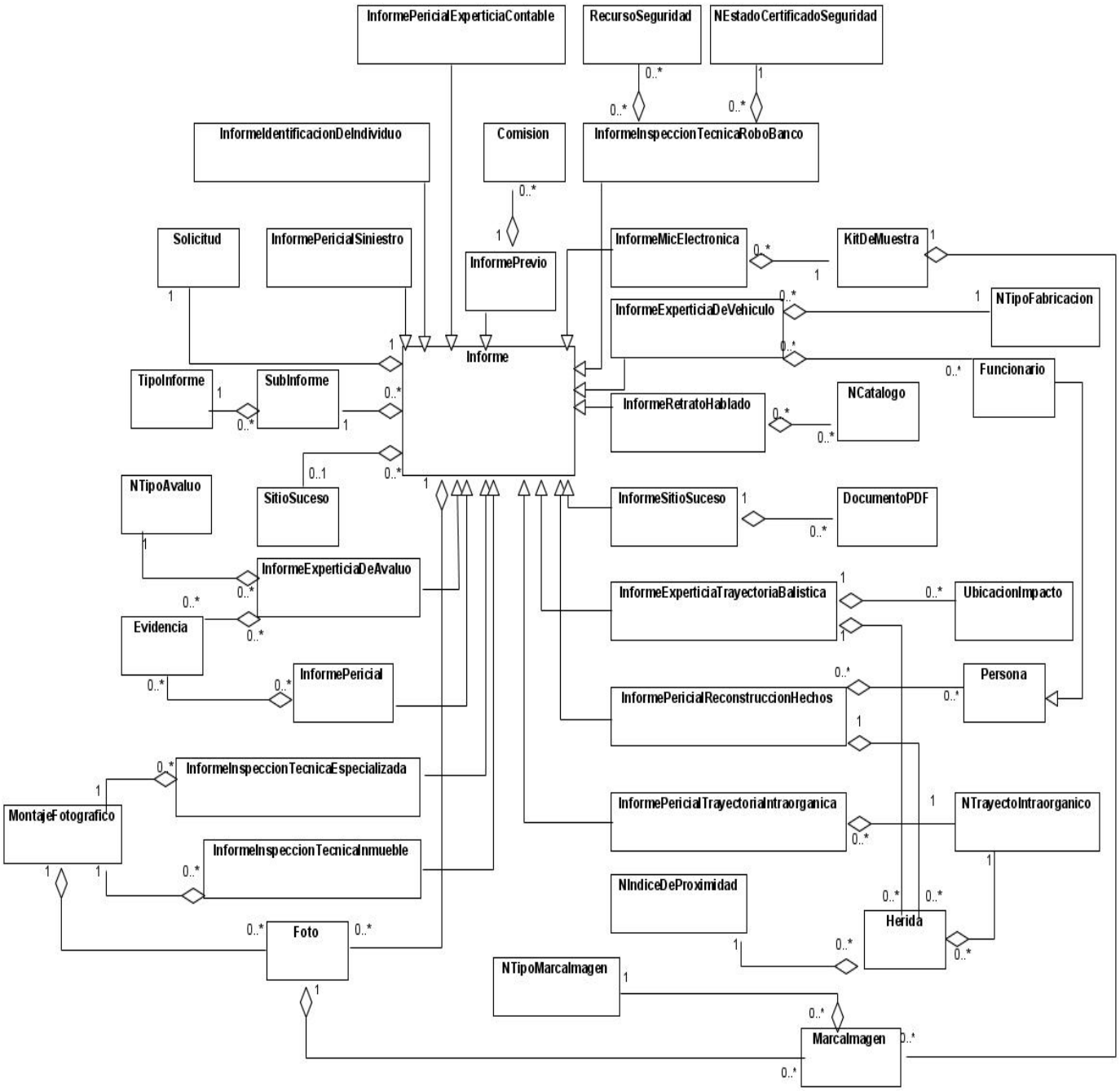

Figura 3. Diagrama de Clases Persistentes del SubMódulo Experticia.

2.4Descripción de las Clases.

La descripción de las clases persistentes muestran los atributos con el tipo de dato que representan cada uno para un mejor entendimiento. A continuación se muestran las principales clases que se utilizan las demás se encuentran en el Anexo 1.

Clase 1. Informe

Nombre: Informe		
	Atributo	**Tipo**
1	noinforme	String

Clase 2. InformeExperticiaDeAvaluo

Nombre: InformeExperticiaDeAvaluo		
	Atributo	**Tipo**
1	peritaje	String
2	conclusiones	String
3	resultado	String
4	descripcion	String
5	referenciales	String
6	calculos	String

Clase 3. InfomreExperticiaDeVehiculo

Nombre: InformeExperticiaDeVehiculo		
	Atributo	**Tipo**
1	peritaje	String
2	conclusiones	String
3	observaciones	String

Clase 4. InformeExperticiaTrayectoriaBalistica

Nombre: InformeExperticiaTrayectoriaBalistica		
	Atributo	**Tipo**
1	conclusiones	String
2	descripcionMedicoLegal	String

Clase 5. InformeIdentificacionDeIndividuo

Nombre: InformeIdentificacionDeIndividuo		
	Atributo	**Tipo**
1	observacion	String
2	conclusiones	String
3	fechaIdentificacion	Date
4	correspondenciaHuellas	Boolean
5	individuoIdentificado	Boolean
6	primerNombre	String
7	segundoNombre	String
8	primerApellido	String
9	segundoApellido	String
10	letraCedula	Character
11	numeroCedula	String

Clase 6. InformeInspeccionTecnicaEspecializada

Nombre: InformeInspeccionTecnicaEspecializada		
	Atributo	**Tipo**
1	motivoNoEviaMontalefotografico	String
2	fechaInspeccionRealizada	Date
3	descripcion	String

Clase 7. InformeInspeccionTecnicaInmueble

Nombre: InformeInspeccionTecnicaInmueble		
	Atributo	**Tipo**
1	descripcionEfectuado	String
2	fechaHoraEfectuadaInspeccion	Date

Clase 8. InformeMicElectronica

Nombre: InformeMicElectronica		
	Atributo	**Tipo**
1	peritaje	String
2	conclusiones	String
3	textoExposicion	String

Clase 9. InformeInspeccionTecnicaRoboBanco

Nombre: InformeInspeccionTecnicaRoboBanco		
	Atributo	**Tipo**
1	fechaOcurrenciaRobo	Date
2	fechaInspeccion	Date
3	tiempoCaducidad	Int
4	observaciones	String
5	conclusiones	String
6	modusOperandis	String
7	noAsaltantes	Int

Clase 10. InformePericial

Nombre: InformePericial		
	Atributo	**Tipo**
1	peritaje	String
2	resumen	String
3	conclusiones	String
4	nombreEspecificoInfPericial	String

Clase 11. InformePericialExperticiaContable

Nombre: InformePericialExperticiaContable		
	Atributo	**Tipo**
1	descripcionPeritaje	String
2	conclusiones	String

Clase 12. InformePericialRecontruccionHechos

Nombre: InformePericialRecontruccionHechos		
	Atributo	**Tipo**
1	descripcionSitioSuceso	String
2	elementosTecnicoCriminalistico	String
3	ElementosMedicoLegal	String
4	conclusiones	String
5	versionInvolucradosExterno	String

Clase 13. InformePericialSiniestro

Nombre: InformePericialSiniestro		
	Atributo	**Tipo**
1	conclusiones	String
2	analisisSiniestro	String
3	incidenteSiniestro	String
4	ubicacionFocoSiniestro	String
5	introduccion	String
6	descripcionSitioSuceso	String
7	motivoNoInclusionFoto	String

Clase 14. InformePericialTrayectoriaIntraorganica

Nombre: InformePericialTrayectoriaIntraorganica		
	Atributo	**Tipo**
1	descripcionPeritaje	String

Clase 15. InformePrevio

Nombre: InformePrevio		
	Atributo	**Tipo**
1	resenna	String
2	caracteristicasDannos	String
3	condicionesSitio	String

Clase 16. InformeRetratoHablado

Nombre: InformeRetratoHablado		
	Atributo	**Tipo**
1	edadAproximada	Int
2	pesoAproximada	Float
3	estaturaAproximada	Float
4	comentario	String

Clase 17. InformeSitioSuceso

Nombre: InformeSitioSuceso		
	Atributo	**Tipo**
1	peritaje	String
2	observaciones	String
3	conclusiones	String

Clase 18. KitDeMuestra

Nombre: KitDeMuestra		
	Atributo	**Tipo**
1	fechaConfeccion	Date
2	noKit	String
3	fechaHoraDelHecho	Date
4	fechaHoraTomaMuestra	Date

Clase 19. Comision

Nombre: Comision		
	Atributo	**Tipo**
1	primerNombre	String
2	segundoNombre	String
3	primerApellido	String
4	segundoApellido	String
5	entidad	String

Clase 20. Herida

Nombre: Herida		
	Atributo	**Tipo**
1	descripcionLugarHerida	String

Clase 21. EntidadBancaria

Nombre: EntidadBancaria		
	Atributo	**Tipo**
1	especificacion	String

Clase 22. Foto

Nombre: Foto		
	Atributo	**Tipo**
1	descripcion	String
2	archivo	Byte{}
3	clase	String

Clase 23. MarcaImagen

Nombre: MarcaImagen		
	Atributo	**Tipo**
1	x1	Int
2	y1	Int
3	x2	Int
4	y2	Int
5	color	Int

Clase 24. MontajeFotografico

Nombre: MontajeFotografico		
	Atributo	**Tipo**
1	numeroInspeccionTecnica	Int
2	fechaCreacion	Date

Clase 25. DocumentoPDF

Nombre: DocumentoPDF		
	Atributo	**Tipo**
1	nombreArchivo	String
2	contenido	Byte{}

Clase 26. RecursoSeguridad

Nombre: RecursoSeguridad		
	Atributo	**Tipo**
1	nombre	String
2	predefinido	Boolean

Clase 27. SitioSuceso

Nombre: SitioSuceso		
	Atributo	**Tipo**
1	nombre	String

Clase 28. Peculiaridad

Nombre: Peculiaridad		
	Atributo	**Tipo**
1	nombre	String
2	predefinido	Boolean

Clase 29. Solicitud

Nombre: Solicitud		
	Atributo	**Tipo**
1	noSolicitud	String
2	fechaReal	Date
3	descripcion	String

Clase 30. SolicitudIdentificacionIndividuo

Nombre: SolicitudIdentificacionIndividuo		
	Atributo	**Tipo**
1	tarjetaHuellasDactilares	Boolean
2	observaciones	String
3	noClise	String
4	solicitadoPor	String
5	noOrdenAprehension	String
6	lee	Boolean
7	escribe	Boolean
8	fecha	Date
9	fechaLlegada	Date
10	formulaDactilar	String
11	subFormulaDactilar	String
12	soporte	String

Clase 31. SolicitudExperticia

Nombre: SolicitudExperticia		
	Atributo	**Tipo**
1	motivo	String

Clase 32. UbicacionImpacto

Nombre: UbicacionImpacto		
	Atributo	**Tipo**
1	nombre	String
2	descripcion	String

2.5 Diseño de la BD. Diagramas Entidad-Relación

En general, el objetivo del diseño de una base de datos relacional es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información. Una de las técnicas para lograrlo consiste en diseñar esquemas que tengan una forma normal adecuada.

Los diagramas o modelos entidad-relación (E-R) son herramientas para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

2.5.1 SubMódulo Solicitud

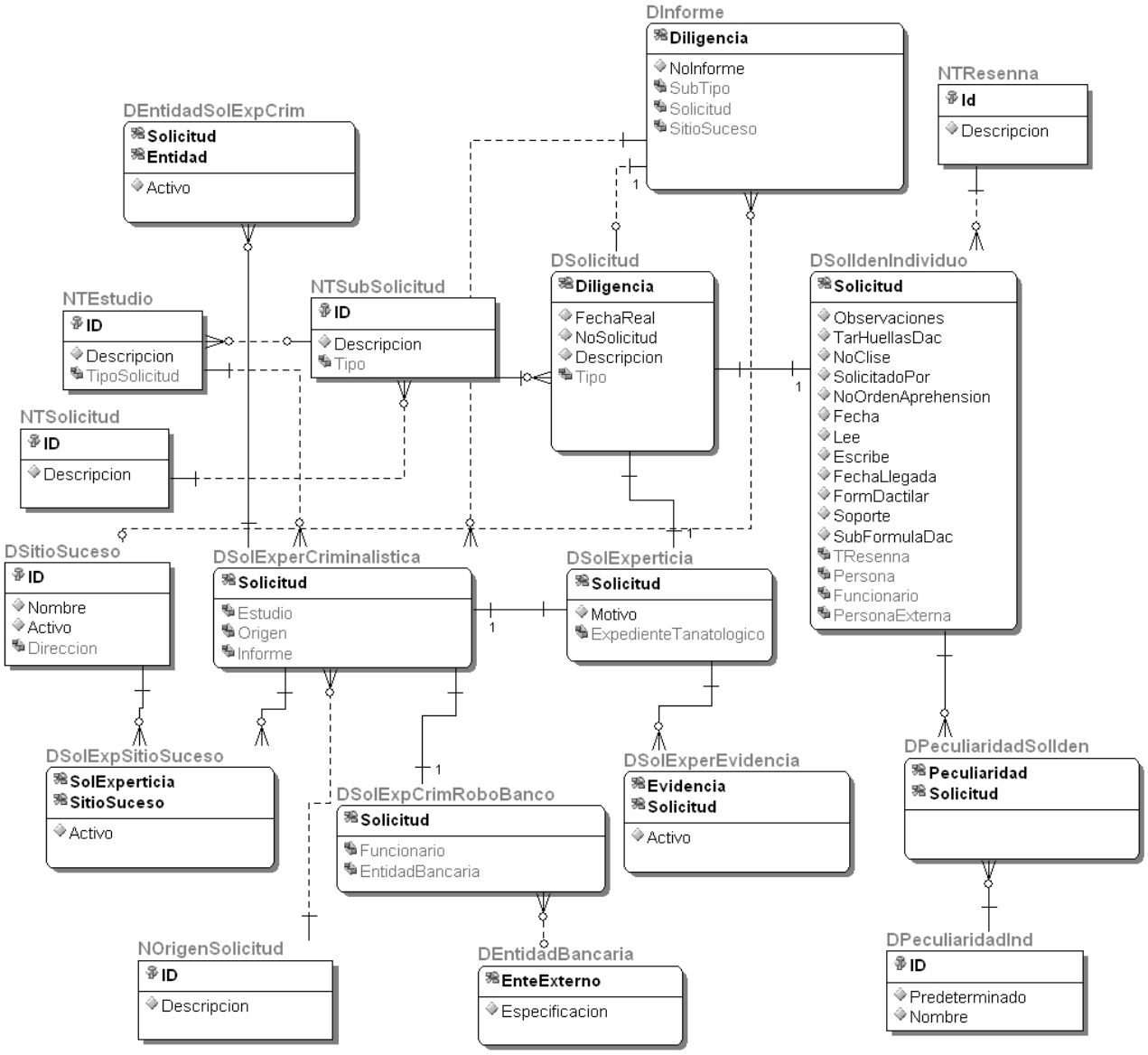

Figura 4. Diagrama Entidad – Relación del SubMódulo Solicitud.

Descripción de las Interrelaciones.

Las interrelaciones del modelo antes visto cumplen las siguientes afirmaciones. Una solicitud de experticia criminalística robo a banco es una solicitud de experticia criminalística, esta es una solicitud de experticia y esta última es una solicitud. Una solicitud de identificación de individuo es una solicitud. Una solicitud es respondida por un informe. Una solicitud tiene un subtipo de solicitud asociada. Una solicitud de identificación de individuo está asociada a varias peculiaridades de individuo, y una peculiaridad de individuo está asociada a varias solicitudes de identificación de individuo. Una solicitud de identificación de individuo tiene un tipo de reseña relacionado. Una solicitud de experticia puede estar relacionada con varias evidencias, y una evidencia puede estar relacionada a varias solicitudes de experticia. Una solicitud de experticia criminalística tiene un origen. Una solicitud de experticia criminalística puede estar relacionada con varios sitios de suceso, y un sitio se suceso puede estar relacionado con varias solicitudes de experticia criminalísticas. Una solicitud de experticia criminalística tiene un tipo de estudio asociado. Una solicitud de experticia criminalística está relacionada con varias entidades, y una entidad está relacionada con varias solicitudes de experticia criminalística. Una solicitud de experticia criminalística robo a banco tiene una entidad bancaria asociada. Un tipo de solicitud tienes varios subtipos de solicitud.

En el diagrama existen relaciones que no están descritas, debido a que fueron insertadas por necesidades de otros módulos.

2.5.2 SubMódulo Experticia.

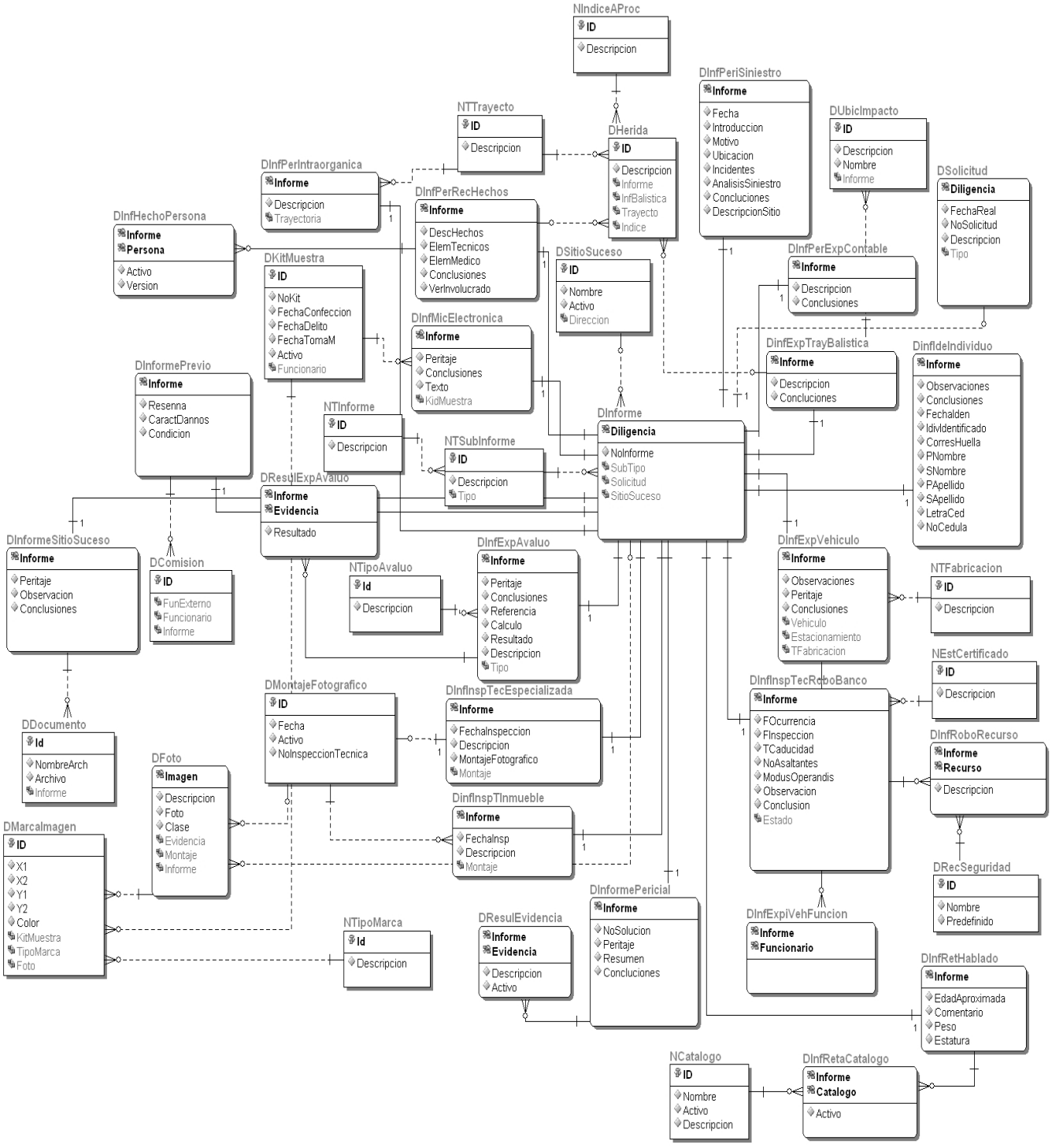

Figura 5. Diagrama Entidad – Relación del SubMódulo Experticia.

Descripción de las Interrelaciones.

Las interrelaciones del modelo antes visto cumplen las siguientes afirmaciones. Un informe de experticia de trayectoria balística tiene varias heridas asociadas. Un informe de experticia de trayectoria balística tiene varias ubicaciones de impactos asociadas. Un informe de experticia a vehículo es un informe. Sobre un informe de experticia a vehículo pueden trabajar varios funcionarios, y un funcionario puede trabajar en varios informes de experticia a vehículo. Un informe de trayectoria balística es un informe. Un informe de identificación de individuo es un informe. Un informe previo tienes varias comisiones asociadas. Un informe de inspección técnica especializada es un informe. Un informe de sitio de suceso está relacionado con muchos documentos PDF. Un informe pericial de reconstrucción de hechos está relacionado con varias heridas. Un informe de inspección de inmuebles lleva un montaje fotográfico. Un informe puede tener varias fotos relacionadas. Un informe de inspección de inmueble es un informe. Un informe de microscopía electrónica es un informe. Un informe pericial es un informe. Un informe previo es un informe. Un informe de sitio de suceso es un informe. Un informe de experticia contable es un informe. Un informe pericial de la trayectoria intraorgánica es un informe. Un informe pericial de siniestro es un informe. Un informe pericial de reconstrucción de hechos es un informe. Un informe de retrato hablado es un informe. Un informe de inspección técnica de robo de banco es un informe. Un informe de inspección técnica de robo de banco tiene varios recursos de seguridad asociados, y un recurso de seguridad tienes varios informes de inspección técnica de robo de banco. Un informe de experticia de avalúo es un informe. Un informe de experticia de avalúo está relacionado con varias evidencias. Una Foto puede estar relacionada con muchas marcas. Un informe pericial puede estar relacionado con varias evidencias, y una evidencia puede estar relacionada con varios informes periciales. Un informe pericial de reconstrucción de hechos puede estar relacionado con varias personas, y una persona puede estar relacionada con varios informes periciales de reconstrucción de hechos. Un informe de retrato hablado está relacionado con varios catálogos, y un catalogo está relacionado con varios informes de retrato hablado. Un informe de microscopia electrónica está relacionado con un kid de muestra. Un kit de muestra está relacionado con varias marcas de kid de muestras. Un montaje fotográfico tiene varias fotos. Un informe de inspección técnica especializada tiene un montaje fotográfico asociado. Un informe está relacionado con un sitio se suceso. Una solicitud es respondida por 0, 1 o dos informes. Un informe de inspección de robo de banco tiene un estado asociado. Un tipo de informe tiene varios subtipos de informe. Un informe de experticia de avalúo tiene un tipo de avalúo asociado. Una marca de kit de muestra tiene un tipo de marca. Un tipo de informe puede estar relacionado con varios informes. Un

informe pericial de trayectoria intraorgánica tiene un tipo de trayecto asociado. Una herida tiene un índice de aproximación asociado. Una herida tiene un trayecto asociado. Un informe de experticia de vehículo tiene un tipo de fabricación asociado.

En el diagrama existen relaciones que no están descritas, debido a que fueron insertadas por necesidades de otros módulos.

2.6 Descripción de las tablas de la base de datos.

A continuación se muestran las principales tablas que se utilizan en la base de datos, las demás se encuentran en el Anexo 2.

Tabla 1. DInforme

Nombre: DInforme		
Descripción: Almacena información general de los informes.		
Atributo	**Tipo**	**Descripción**
Diligencia	NUMERIC (15)	Id del informe (Llave primaria).
Solicitud	NUMERIC (15)	Id de la solicitud asociada al informe (FK).
SitioSuceso	NUMERIC (15)	Id del sitio de suceso que está relacionado con el informe (FK).
SubTipo	NUMERIC (5)	Id del tipo de informe (FK).
NoInforme	VARCHAR (50)	Número del informe.

Tabla 2. DInfExpAvaluo

Nombre: DInfExpAvaluo		
Descripción: Almacena información de los informes de experticia de avalúo.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Referencia	VARCHAR (2000)	Almacena una descripción textual que hace referencia a otros casos.
Tipo	NUMERIC (5)	Id del tipo de informe de experticia de avalúo (FK).
Descripcion	VARCHAR (255)	Descripción del informe.
Conclusiones	VARCHAR (100)	Texto que refleja las conclusiones a que llega el experto.
Peritaje	VARCHAR (100)	Texto que explica el método científico usado para determinar el resultado de lo efectuado.
Resultado	VARCHAR (100)	Resultado.
Calculo	VARCHAR (255)	Calculo.

Tabla 3. DInfExpVehiculo

Nombre: DInfExpVehiculo		
Descripción: Almacena los datos de las experticias realizadas a los vehículos		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Peritaje	VARCHAR (255)	Texto que refleja lo realizado por los expertos.
Observaciones	VARCHAR (255)	Texto que refleja las observaciones del experto durante su revisión.
Conclusiones	VARCHAR (255)	Texto que refleja las conclusiones a que llega el experto.
TFabricacion	NUMERIC (2)	Id del Tipo de fabricación (FK).
Estacionamiento	NUMERIC (15)	Id del estacionamiento en el que se encuentra el vehículo (FK).
Vehiculo	NUMERIC (15)	Id del Vehículo asociado (FK).

Tabla 4. DinfExpTrayBalistica

Nombre: DinfExpTrayBalistica		
Descripción: Almacena los datos de los informes de experticia de las trayectorias balísticas.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Lave primaria)
Conclusiones	VARCHAR (255)	Conclusiones del informe.
Descripcion	VARCHAR (255)	Descripción del informe.

Tabla 5. DInfInspTecEspecializada

Nombre: DInfInspTecEspecializada		
Descripción: Almacena los datos de los informes de inspección técnica especializada.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Lave primaria).
FechaInspeccion	DATE	Fecha en que se realiza la inspección.
MontajeFotografico	VARCHAR (255)	Motivo de no envío del montaje fotográfico.
Descripcion	VARCHAR (255)	Descripción del informe.
Montaje	NUMERIC (15)	Id del Montaje asociado (FK).

Tabla 6. DInfIdeIndividuo

Nombre: DInfIdeIndividuo		
Descripción: Almacena los datos de los informes de identificación de individuo.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Lave primaria).
PNombre	VARCHAR (15)	Primer nombre.
CorresHuella	BIT (1)	Corresponde con la huella o no.
FechaIden	DATE	Fecha de identificación.
Conclusiones	VARCHAR (255)	Conclusiones.
Observaciones	VARCHAR (255)	Observaciones.
IdvIdentificado	BIT (1)	Individuo identificado o no.
SNombre	VARCHAR (15)	Segundo nombre.
PApellido	VARCHAR (15)	Primer apellido.
SApellido	VARCHAR (15)	Segundo apellido
LetraCed	CHAR (1)	Letra de la cedula.
NoCedula	VARCHAR (8)	Número de cédula de la persona a identificar.

Tabla 7. DinfInspTInmueble

Nombre: DinfInspTInmueble		
Descripción: Almacena los datos de los informes de inspección de inmuebles.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
FechaInsp	DATE	Fecha de inspección.
Montaje	NUMERIC (15)	Id del Montaje asociado (FK).
Descripcion	VARCHAR (255)	Descripción del informe.

Tabla 8. DInfMicElectronica

Nombre: DInfMicElectronica		
Descripción: Almacena los datos de los informes de microscopia electrónica.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria)
Conclusiones	VARCHAR (255)	Conclusiones del informe.
Peritaje	VARCHAR (255)	Peritaje.
Texto	VARCHAR (255)	Texto del informe
KidMuestra	NUMERIC (15)	Id del Kid de muestra que está relacionado con el informe (FK).

Tabla 9. DInfInspTecRoboBanco

Nombre: DInfInspTecRoboBanco		
Descripción: Almacena información de los informes de inspección técnica de robo de banco.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
FOcurrencia	DATE	Fecha de ocurrencia del robo
Estado	NUMERIC (5)	Id del estado certificado de seguridad asociado (FK).
Observacion	VARCHAR (255)	Observación.
ModusOperandis	VARCHAR (255)	Modus Operandi.
TCaducidad	NUMERIC (4)	Tiempo de caducidad.
Conclusion	VARCHAR (255)	Conclusiones
NoAsaltantes	NUMERIC (4)	Numero de asaltantes.
FInspeccion	DATE	Fecha de inspección.

Tabla 10. DInformePericial

Nombre: DInformePericial		
Descripción: Almacena los datos de los informes periciales.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Resumen	VARCHAR (150)	Resumen del informe.
Conclusiones	VARCHAR (500)	Conclusiones.
NoSolucion	VARCHAR (15)	Número de la solución
Peritaje	VARCHAR (1000)	Peritaje.

Tabla 11. DInfPerExpContable

Nombre: DInfPerExpContable		
Descripción: Almacena los datos de los informes de experticia contable		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Descripcion	VARCHAR (255)	Descripción del informe.
Conclusiones	VARCHAR (255)	Conclusiones del informe.

Tabla 12. DInfPerRecHechos

Nombre: DInfPerRecHechos		
Descripción: Almacena los datos de los informes periciales de reconstrucción de hechos.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Conclusiones	VARCHAR (255)	Conclusiones del informe.
ElemMedico	VARCHAR (255)	Elementos técnicos médico legal.
VerInvolucrado	VARCHAR (4000)	Versión de los involucrados.
DescHechos	VARCHAR (1000)	Descripción de los hechos.
ElemTecnicos	VARCHAR (255)	Elementos técnicos criminalísticas.

Tabla 13. DInfHechoPersona

Nombre: DInfHechoPersona		
Descripción: Almacena la relación de los informes de reconstrucción de hechos y la personas.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Persona	NUMERIC (15)	Id de la persona (Llave primaria).
Version	VARCHAR (1000)	Versión de la persona del acontecimiento de los hechos.
Activo	BIT (1)	Si está activo o no.

Tabla 14. DKitMuestra

Nombre: DKitMuestra		
Descripción: Almacena información de los kid de muestra.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (15)	Id del kid de muestra (Llave primaria).
NoKit	VARCHAR (15)	Número de kid de muestra.
FechaDelito	DATE	Fecha del delito.
FechaConfeccion	DATE	Fecha en que se confecciona el kid
FechaTomaM	DATE	Fecha en que se toma la muestra.
Activo	BIT (1)	Si está activo o no.
Funcionario	NUMERIC (15)	Id del Funcionario que toma la muestra (FK).

Tabla 15. DInfPeriSiniestro

Nombre: DInfPeriSiniestro		
Descripción: Almacena la información relacionada con los informes periciales de siniestro.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
DescripcionSitio	VARCHAR (4000)	Especifica las condiciones del sitio.
Ubicacion	VARCHAR (4000)	Texto donde evidencia todo lo fundamental de los puntos y sitios específicos del siniestro
Introduccion	VARCHAR (4000)	Introducción.
Incidentes	VARCHAR (4000)	Incidentes del siniestro.
Motivo	VARCHAR (4000)	Motivo por el cual el despacho solicitante realiza la solicitud.
Fecha	DATE	Fecha.
AnalisisSiniestro	VARCHAR (4000)	Reflejar lo especifico del análisis realizado por los expertos
Conclusiones	VARCHAR (4000)	Conclusiones del informe.

Tabla 16. DInfPerIntraorganica

Nombre: DInfPerIntraorganica		
Descripción: Almacena los datos de los informes periciales de las trayectorias intraorgánica.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Descripcion	VARCHAR (255)	Descripción.
Trayectoria	NUMERIC (5)	Id de la trayectoria asociada (FK).

Tabla 17. DInformePrevio

Nombre: DInformePrevio		
Descripción: Almacena los datos de los informes previos.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
CaractDannos	VARCHAR (255)	Características de los daños.
Resenna	VARCHAR (255)	Reseña de los hechos.
Condicion	VARCHAR (255)	Condiciones.

Tabla 18. DInfRetHablado

Nombre: DInfRetHablado		
Descripción: Almacena información de los informe de retrato hablado.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
EdadAproximada	NUMERIC (3)	Edad aproximada de la persona.
Peso	FLOAT (8)	Peso de la persona.
Estatura	FLOAT (8)	Estatura de la persona.
Comentario	VARCHAR (255)	Comentario referente al retrato hablado.

Tabla 19. DInformeSitioSuceso

Nombre: DInformeSitioSuceso		
Descripción: Almacena la información de los informes de sitio de suceso.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Conclusiones	VARCHAR (255)	Conclusiones.
Peritaje	VARCHAR (255)	Peritaje.
Observacion	VARCHAR (255)	Observación.

Tabla 20. DComision

Nombre: DComision		
Descripción: Almacena información de las comisiones presentes en los informes previos de siniestro.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (15)	Id de la comisión (Llave primaria).
FunExterno	NUMERIC (15)	Id del funcionario externo asociado (FK).
Funcionario	NUMERIC (15)	Id del Funcionario que participa (FK)
Informe	NUMERIC (15)	Id del informe asociado (FK).

Tabla 21. DInfExpiVehFuncion

Nombre: DInfExpiVehFuncion		
Descripción: Almacena la información de la relación entre los funcionarios y el informe de inspección al vehículo.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Funcionario	NUMERIC (15)	Id del funcionario (Llave primaria).

Tabla 22. DHerida

Nombre: DHerida		
Descripción: Almacena los datos de las heridas.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (15)	Id de la herida.
InfRecHechos	NUMERIC (15)	Id del informe pericial de reconstrucción de hechos
Descripcion	VARCHAR (255)	Descripción de la herida.
Trayecto	NUMERIC (5)	Id del trayecto asociado.
InfBalistica	NUMERIC (15)	Id del informe de experticia de la trayectoria balística asociado.
Indice	NUMERIC (5)	Id del índice asociado.

Tabla 23. DFoto

Nombre: DFoto		
Descripción: Almacena información de las fotos.		
Atributo	**Tipo**	**Descripción**
Imagen	NUMERIC (15)	Id de la foto (Llave primaria).
Evidencia	NUMERIC (15)	Id de la evidencia relacionada con la foto (FK).
Montaje	NUMERIC (15)	Id del montaje relacionado con la foto (FK).
Informe	NUMERIC (15)	Id del informe asociado (FK).
Descripcion	VARCHAR (255)	Descripción de la foto.
Foto	PICTURE (18)	Especifica la foto.
Clase	VARCHAR (1)	Especifica el tipo de foto.

Tabla 24. DMarcaImagen

Nombre: DMarcaImagen		
Descripción: Almacena las coordenadas de las marcas.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id de la marca (Llave primaria).
KitMuestra	NUMERIC (15)	Id del kit de muestra asociado (FK).
Y2	NUMERIC (3)	Coordenadas.
Fotografia	NUMERIC (15)	Id de la fotografía (FK).
Color	NUMERIC (8)	Color de la marca.
TipoMarca	NUMERIC (5)	Especifica el tipo de muestra (FK).
Y1	NUMERIC (3)	Coordenadas.
X1	NUMERIC (3)	Coordenadas.
X2	NUMERIC (3)	Coordenadas.

Tabla 25. DEntidadBancaria

Nombre: DEntidadBancaria		
Descripción: Almacena los datos de las entidades bancarias.		
Atributo	**Tipo**	**Descripción**
EnteExterno	NUMERIC (15)	Id de la entidad bancaria (Llave primaria)
Especificacion	VARCHAR (150)	Especificación de la entidad bancaria

Tabla 26. DEntidadSolExpCrim

Nombre: DEntidadSolExpCrim		
Descripción: Almacena los datos de las dependencias que están relacionadas con las solicitudes de experticia criminalísticas.		
Atributo	**Tipo**	**Descripción**
Solicitud	NUMERIC (15)	Solicitud que está relacionada con la dependencia (Llave primaria).
Entidad	NUMERIC (15)	Entidad que está relacionada con la solicitud de experticia criminalística (Llave primaria).
Activo	BIT (1)	Si está activo o no.

Tabla 27. DMontajeFotografico

Nombre: DMontajeFotografico		
Descripción: Almacena los datos de los montajes fotográficas.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (15)	Id del montaje (Llave primaria).
Activo	BIT (1)	Si está activo o no.
NoInspeccionTecnica	VARCHAR (18)	Numero de inspección técnica
Fecha	DATE	Fecha de creación del montaje fotográfico.

Tabla 28. DDocumento

Nombre: DDocumento		
Descripción: Almacena los datos de los documentos.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (15)	Id del documento (Llave primaria).
NombreArch	VARCHAR (60)	Nombre del archivo.
Archivo	VARBINARY/BLOG(18)	Archivo que se almacena.
Informe	NUMERIC (15)	Informe asociado (FK).

Tabla 29. DInfRetaCatalogo

Nombre: DInfRetaCatalogo		
Descripción: Almacena los datos de la relación entre catálogos y informes.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Catalogo	NUMERIC (15)	Id del catalogo (Llave primaria).
Activo	BIT (1)	Si esta activo o no.

Tabla 30. DInfRoboRecurso

Nombre: DInfRoboRecurso		
Descripción: Almacena la relación entre informe de robo de banco y los recursos de seguridad.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Recurso	NUMERIC (5)	Id del recurso (Llave primaria).
Descripcion	VARCHAR (255)	Descripción de elemento de seguridad.

Tabla 31. DRecSeguridad

Nombre: DRecSeguridad		
Descripción: Almacena los datos de recursos de seguridad.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del recurso de seguridad (Llave primaria).
Nombre	VARCHAR (60)	Nombre del recurso.
Predefinido	BIT (1)	Especifica si el valor esta predefinido.

Tabla 32. DResulEvidencia

Nombre: DResulEvidencia		
Descripción: Almacena la información de los resultados de la experticias que se le realizan a las evidencias.		
Atributo	**Tipo**	**Descripción**
Evidencia	NUMERIC (15)	Id de la evidencia (Llave primaria).
Informe	NUMERIC (15)	Id del informe pericial (Llave primaria).
Activo	BIT (1)	Si está activo o no.
Descripcion	VARCHAR (255)	Descripción.

Tabla 33. DResulExpAvaluo

Nombre: DResulExpAvaluo		
Descripción: Almacena los datos de la relación entre informes de experticias de avaluó y evidencia.		
Atributo	**Tipo**	**Descripción**
Informe	NUMERIC (15)	Id del informe (Llave primaria).
Evidencia	NUMERIC (15)	Id de la evidencia (Llave primaria).
Resultado	VARCHAR (255)	Resultado.

Tabla 34. DSitioSuceso

Nombre: DSitioSuceso		
Descripción: Almacena información del sitio del suceso		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (15)	Id del sitio del suceso (Llave primaria).
Nombre	VARCHAR (150)	Nombre del sitio del suceso.
Activo	BIT (1)	Si esta activo o no.
Direccion	NUMERIC (15)	Dirección asociada (FK).

Tabla 35. DPeculiaridadInd

Nombre: DPeculiaridadInd		
Descripción: Almacena las peculiaridades del individuo asociado a la solicitud.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (3)	Id de la Peculiaridad (Llave primaria).
Predeterminado	BIT (1)	Define si es predeterminado o no.
Nombre	VARCHAR (30)	Nombre de la peculiaridad.

Tabla 36. DPeculiaridadSolIden

Nombre: DPeculiaridadSolIden		
Descripción: Almacena información de la relación entre la Solicitud de identificación de individuo y las peculiaridades.		
Atributo	**Tipo**	**Descripción**
Peculiaridad	NUMERIC (3)	Peculiaridad asociada (Llave primaria).
Solicitud	NUMERIC (15)	Solicitud de identificación de individuo (Llave primaria).

Tabla 37. DSolicitud

Nombre: DSolicitud		
Descripción: Almacena información general de las solicitudes.		
Atributo	**Tipo**	**Descripción**
Diligencia	NUMERIC (15)	Id de la solicitud (Llave primaria).
FechaReal	DATE	Fecha en que se hace realmente la solicitud.
Descripcion	VARCHAR (255)	Descripción de la solicitud
Tipo	NUMERIC (5)	Tipo de la solicitud (FK)
NoSolicitud	VARCHAR (15)	Numero de la solicitud

Tabla 38. DSolIdenIndividuo

Nombre: DSolIdenIndividuo		
Descripción: Almacena los datos de las solicitudes de identificación de las personas.		
Atributo	**Tipo**	**Descripción**
Solicitud	NUMERIC (15)	Id de la solicitud de identificación de individuo (Llave primaria).
TarHuellasDac	BIT (1)	Tarjeta de huellas dactilares.
NoClise	VARCHAR (15)	Número de Clise.
SolicitadoPor	VARCHAR (60)	Nombre del que solicita.
NoOrdenAprehension	VARCHAR (15)	Número de la orden de aprehensión.
Fecha	DATE	Fecha de toma de datos.
Lee	BIT (1)	Si la persona sabe leer.
Escribe	BIT (1)	Si la persona sabe escribir.
FechaLlegada	DATE	Fecha de llegada del individuo.
FormDactilar	VARCHAR (1000)	Fórmula dactilar.
Soporte	VARCHAR (2000)	Soporte.
Funcionario	NUMERIC (15)	Funcionario que toma el rastro (FK).
PerExterno	NUMERIC (15)	Persona externa (FK).
Observaciones	VARCHAR (255)	Observaciones.
SubFormulaDac	VARCHAR (1000)	Sub fórmula dactilar
Persona	NUMERIC (15)	Persona (FK)
TResenna	NUMERIC (5)	Reseña asociada (FK).

Tabla 39. DSolExpSitioSuceso

Nombre: DSolExpSitioSuceso		
Descripción: Almacena información de las solicitudes de experticia criminalística y los sitios de suceso.		
Atributo	**Tipo**	**Descripción**
SitioSuceso	NUMERIC (15)	Sito de suceso que está relacionado con la solicitud de experticia criminalística (Llave primaria).
SolExperticia	NUMERIC (15)	Solicitud de experticia criminalística que está relacionada con el sitio se suceso (Llave primaria).
Activo	BIT (1)	Si esta Activo o no.

Tabla 40. DSolExperticia

Nombre: DSolExperticia		
Descripción: Almacena la información de las solicitudes de las experticias criminalísticas.		
Atributo	**Tipo**	**Descripción**
Solicitud	NUMERIC (15)	Id de la solicitud de experticia (Llave primaria).
Motivo	VARCHAR (300)	Motivo de la solicitud de experticia.
ExpedienteTanatologico	NUMERIC (15)	Expediente tanatológico asociado (FK).

Tabla 41. DSolExperEvidencia

Nombre: DSolExperEvidencia		
Descripción: Almacena la relación entre las solicitudes de experticias y las evidencias.		
Atributo	**Tipo**	**Descripción**
Evidencia	NUMERIC (15)	Una evidencia puede estar relacionada con muchas solicitudes de experticias (Llave primaria).
Solicitud	NUMERIC (15)	Una solicitud de experticia puede estar relacionada con muchas evidencias (Llave primaria).
Activo	BIT (1)	Si esta activo o no.

Tabla 42. DSolExperCriminalistica

Nombre: DSolExperCriminalistica		
Descripción: Almacena los datos de las solicitudes criminalísticas.		
Atributo	**Tipo**	**Descripción**
Solicitud	NUMERIC (15)	Id de la solicitud (Llave primaria).
Origen	NUMERIC (5)	Origen asociado (FK).
Informe	NUMERIC (15)	Informe asociado (FK).
Estudio	NUMERIC (5)	Estudio asociado (FK).

Tabla 43. DSolExpCrimRoboBanco

Nombre: DSolExpCrimRoboBanco		
Descripción: Almacena la información de las solicitudes de experticia criminalística robo a banco.		
Atributo	**Tipo**	**Descripción**
Solicitud	NUMERIC (15)	Id de la solicitud (Llave primaria).
Funcionario	NUMERIC (15)	Funcionario asociado (FK).
EntidadBancaria	NUMERIC (15)	Entidad bancaria asociada (FK).

Tabla 44. DUbicImpacto

Nombre: DUbicImpacto		
Descripción: Almacena los datos de la ubicación de los impactos.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (15)	Id de la ubicación de los impactos (Llave primaria).
Descripcion	VARCHAR (255)	Descripción.
Nombre	VARCHAR (60)	Nombre.
Informe	NUMERIC (15)	Id del informe de experticia trayectoria balística asociado (FK).

2.7 Descripción de los Procedimientos Almacenados.

Un procedimiento almacenado es un elemento de base de datos, reutilizable que realiza alguna operación en la base de datos. En el proyecto CICPC se emplea la herramienta Hibernate para gestionar el acceso a datos, lo que facilita realizar un gran número de consultas SQL. Sólo los procedimientos más complejos fueron realizados en PL/SQL. A continuación se muestran los procedimientos almacenados utilizados. Para ver el código de los principales remitirse al Anexo 3.

Procedimientos de almacenados del módulo Estadísticas.

Procedimiento Almacenado	Descripción
SP_Principales_Delitos(cursor ref_cursor, fecha_inicial date, fecha_fin date)	Listado con los principales delitos ordenados por fecha y hora.
SP_Delitos_Totales(cursor ref_cursor, fecha_inicial date, fecha_fin date)	Listado con todos los delitos ordenados por fecha y hora.
SP_Dias_Mayor_Indice_Delictivo(cursor ref_cursor, fecha_inicial date, fecha_fin date, cant_mostrar numeric)	Listado ordenado por cantidad de delitos que ocurrieron por días (descendentemente).
SP_Zonas_Mayor_Ind_Delictivo(cursor ref_cursor, fecha_inicial date, fecha_fin date, cant_mostrar numeric)	Listado ordenado por cantidad de delitos que ocurrieron por Zonas (descendentemente).
SP_Homicidios_Discrim_Por_Tipo(cursor ref_cursor, fecha_inicial date, fecha_fin date)	Un listado con todos los datos de los delitos de homicidios que ocurrieron entre las fechas entradas (ordenados por fecha).
SP_Robo_Autos_Discrim_Por_Tipo(cursor ref_cursor, fecha_inicial date, fecha_fin date)	Listado con todos los datos de los delitos de robo de autos que ocurrieron entre las fechas entradas (ordenados por fecha).
SP_Delitos_Mas_Frecuentes(cursor ref_cursor, fecha_inicial date, fecha_fin date)	Listado con los delitos más frecuentes por zonas.
SP_Horas_Mayor_Incidencia(cursor ref_cursor, fecha_inicial date, fecha_fin date, cant_mostrar numeric)	Listado ordenado por cantidad de delitos que ocurrieron a una hora determinada (descendentemente).
SP_Zona_Hurto_Robo_Vehiculo(cursor ref_cursor, fecha_inicial date, fecha_fin date, cant_mostrar numeric)	Listado ordenado por cantidad de hurtos y robos de vehículos que ocurrieron por parroquias (descendentemente).
SP_Zona_Homicidio(cursor ref_cursor, fecha_inicial date, fecha_fin date, cant_mostrar numeric)	Listado ordenado por cantidad de homicidios que ocurrieron por parroquias (descendentemente).
SP_Promedio_Delitos_Diario(cursor ref_cursor, fecha_inicial date, fecha_fin date)	Promedio de delitos diario en la fecha comprendida.
SP_Delitos_Mas_Frec_Por_Parro(cursor ref_cursor, fecha_inicial date, fecha_fin date,	Listado de los delitos más frecuentes, y la

parroquia numeric, cant_mostrar numeric)	cantidad.
SP_Listado_Parroquias(cursor ref_cursor)	Listado con todas las Parroquias.
SP_Buscar_Delito_Criteria**(**cursor ref_cursor, criterio varchar2, fecha_inicio date, fecha_fin date)	Procedimiento de almacenado que pasándole una cadena con los criterios de búsqueda, este realiza la búsqueda estadística.
SP_Exped_Remit_Fiscalia(cursor ref_cursor, fecha_inicial date , fecha_fin date, cant_mostrar numeric)	Listado con los expedientes remitidos a fiscalía entre esas fechas, ordenados por fecha.
SP_Casos_Iniciados_Por_Deleg(cursor ref_cursor, fecha_inicial date, fecha_fin date, delegación varchar2, cant_mostrar numeric)	Listado con los casos iniciados por la delegación entrada, ordenados por fecha.
SP_Casos_Concluidos_Por_Deleg(cursor ref_cursor, fecha_inicial date, fecha_fin date, delegación varchar2, cant_mostrar numeric)	Listado con los casos concluidos por la delegación entrada, ordenados por fecha.
SP_Casos_Inic_Discri_Delito(cursor ref_cursor, fecha_inicial date, fecha_fin date, tipo_delito varchar2, cant_mostrar numeric)	Listado con los casos iniciados por la delegación entrada, ordenados por fecha, y discriminados por delitos.
SP_Casos_Conclu_Discri_Delito(cursor ref_cursor, fecha_inicial date, fecha_fin date, tipo_delito varchar2, cant_mostrar numeric)	Listado con los casos concluidos por la delegación entrada, ordenados por fecha, y discriminados por delitos.
SP_Casos_En_Curso(cursor ref_cursor, fecha_inicial date, fecha_fin date, delegación varchar2, cant_mostrar numeric)	Listado con los casos en curso, ordenados por fecha.
SP_Diligencias_Por_Delegacion(cursor ref_cursor, fecha_inicial date, fecha_fin date, delegación varchar2)	Listado con las diligencias practicadas por delegación, ordenados por fecha.
SP_Diligencias_En_Curso(cursor ref_cursor, fecha_inicial date, fecha_fin date, delegación varchar2)	Listado con las diligencias en curso por delegación, ordenados por fecha.
SP_Funcionarios_Asig_Por_Caso(cursor ref_cursor, fecha_inicial date, fecha_fin date, dependencia varchar2 , Pnombre varchar2,	Listado ordenado por fecha y hora, con los casos asignados al funcionario.

Snombre varchar2, credencial varchar2)	
SP_Funciona_Mas_Casos_Llevan(cursor ref_cursor , fecha_inicial date, fecha_fin date, dependencia varchar2, cant_mostrar numeric)	Listado ordenado por cantidades con el nombre del funcionario y la cantidad de casos que lleva.
SP_Func_Menos_Casos_Llevan(cursor ref_cursor , fecha_inicial date, fecha_fin date, dependencia varchar2, cant_mostrar numeric)	Listado ordenado por cantidades con el nombre del funcionario y la cantidad de casos que lleva.
SP_Diligencias_Mas_Hacen(cursor ref_cursor, fecha_inicial date, fecha_fin date, cant_mostrar numeric)	Listado de las diligencias que más se hacen.
SP_Diligencias_Menos_Hacen(cursor ref_cursor, fecha_inicial date, fecha_fin date, cant_mostrar numeric)	Listado de las diligencias que menos se hacen
SP_Casos_Mayor_Importancia(cursor ref_cursor, fecha_inicial date, fecha_fin date , dependencia varchar2, cant_mostrar numeric)	Listado con los casos de mayor importancia, ordenados por fecha.
SP_Funcionari_Casos_Sin_Cerrar(cursor ref_cursor, fecha_inicial date, fecha_fin date, dependencia varchar2)	Listado de funcionarios con casos sin cerrar.
SP_Casos_Mas_Rapido_Cerrado(cursor ref_cursor, fecha_inicial date, fecha_fin date, dependencia varchar2, cant_mostrar numeric)	Listado de casos que más rápido se han cerrado.
SP_Casos_Mas_Llevan_Abiertos(cursor ref_cursor, fecha_inicial date, fecha_fin date, dependencia varchar2, cant_mostrar numeric)	Listado de casos que más tiempo llevan de apertura dos.
SP_Func_Rapido_Cerrado_Casos(cursor ref_cursor, fecha_inicial date, fecha_fin date, dependencia varchar2, cant_mostrar numeric)	Listado de funcionarios que más rápido han cerrado los casos.
SP_Func_Lento_Cerrado_Casos(cursor ref_cursor, fecha_inicial date, fecha_fin date, dependencia varchar2, cant_mostrar numeric)	Listado de funcionarios que más lento han cerrado los casos.
SP_Buscar_Casos_Criteria(cursor ref_cursor, criterio varchar2, fecha_inicial date, fecha_fin date)	Búsqueda personalizada de los casos.

Procedimientos de almacenados del módulo Investigación Criminalística.

Procedimiento de almacenado	Descripción
SP_Buscar_Informe2(cursor ref_cursor, no_informe varchar2, no_solicitud varchar2, no_acta_procesal varchar2, estado_solicitud numeric, estado_informe numeric, fecha_remision date, no_evidencia varchar2, funcionario numeric, despacho_solicitante numeric, dependencia numeric, tipos_informe varchar2)	Buscar informes por una serie de criterios pasados por parámetro.
SP_Buscar_Solicitud(cursor ref_cursor, no_solicitud varchar2, dependencia numeric, estado_solicitud numeric, criterio varchar2)	Buscar solicitudes por una serie de criterios pasados por parámetro.
SP_Solicitudes_Por_Tipo(cursor ref_cursor, fecha_inicio date, fecha_fin date, cant_mostrar numeric)	Lista las solicitudes por tipos y subtipos.
SP_Informes_Por_Tipo(cursor ref_cursor, fecha_inicio date, fecha_fin date, cant_mostrar numeric)	Lista los informes por tipos y subtipos
SP_Informes_Por_Fecha(cursor ref_cursor, fecha_inicio date, fecha_fin date, dependencia numeric, cant_mostrar numeric)	Lista los informes por fecha.
SP_Numero_Infomre(no_despacho varchar2, secuencia out varchar2)	Genera el número del informe dado la dependencia.
SP_Numero_Solicitud(no_despacho varchar2, secuencia out varchar2)	Genera el número de la solicitud dado la dependencia.
SP_Buscar_Solicitud_Vehiculo(cursor ref_cursor , importancia numeric, no_acta_procesal varchar2, no_evidencia varchar2, placa varchar2, no_solicitud varchar2, serial_carroseria varchar2, serial_motor varchar2, tipo_vehiculo numeric, fecha_remision_inicial date, fecha_remision_final date, funcionario	Busca las solicitudes de vehículo por los criterios pasados por parámetros.

numeric, cadena_dependencias varchar2)	
SP_Buscar_Persona_Identificada(cursor ref_cursor, pnombre varchar2, snombre varchar2, papellido varchar2, sapellido varchar2, lcedula char, no_cedula varchar2, no_pasaporte varchar2)	Busca las personas por los criterios pasados por parámetro.

Funciones utilizadas en el módulo Estadística e Investigación Criminalística.

Función	**Descripción**
FN_Parsear_Cadena_Estadisticas (criterio varchar2)	Dada una cadena entrada por parámetros se obtiene las condiciones que se ponen en el WHERE del procedimiento almacenado desde donde se llame a esta función.
FN_Quote (cadena varchar2)	A la cadena pasada por parámetro se le ponen 4 comillas delante y 4 detrás.

Conclusiones

Se concluyó la etapa de análisis y diseño, se crearon los diagramas utilizando el Visual Paradigm y ER/Studio 7.0, así como también las descripciones de cada entidad o clase necesaria para dichos diagramas, se hicieron los procedimientos almacenados utilizando el PL/SQL Developer, logrando obtener finalmente la construcción de la base de datos, a través del gestor de bases de datos Oracle.

CAPÍTULO 3. Validación del diseño realizado.

Introducción.

En este capítulo se brinda información sobre la validación realizada al diseño descrito en el Capítulo 2, El mismo presenta la validación teórica del diseño, la integridad, así como la normalización y el análisis de la redundancia. También se expone la validación funcional, la optimización de consultas y los resultados de las pruebas realizadas.

3.1Validación teórica del diseño.

Para realizar un buen diseño de BD hay que tener en cuenta varios aspectos importantes tales como: la integridad de los datos, la normalización del diseño y la redundancia de información. Otro aspecto fundamental y el más importante es la seguridad de la base de datos, sobre este en nuestro proyecto se creó un módulo Administración que vela por el control del acceso a los datos y además prevé que los mismos no se corrompan.

3.1.1 Integridad de los datos.

El término integridad de datos se refiere a la corrección y completitud de los datos en una base de datos. Cuando los contenidos de una base de datos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes.

- ✓ Pueden añadirse datos no válidos a la base de datos.
- ✓ Pueden modificarse datos existentes tomando un valor incorrecto.
- ✓ Los cambios en la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de energía.
- ✓ Los cambios pueden ser aplicados parcialmente.

Una de las funciones importantes de un DBMS relacional es preservar la integridad de sus datos almacenados en la mayor medida posible.

La exigencia de integridad de los datos garantiza la calidad de los datos de la base de datos. Dos pasos importantes en el diseño de las tablas son la identificación de valores válidos para una columna y la determinación de cómo forzar la integridad de los datos en la columna.

Tipos de Restricciones de Integridad en Bases de Datos Relacionales

Datos Requeridos: Establece que una columna tenga un valor no nulo. Se define efectuando 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.

En la tabla DSitioSuceso se puede ver un ejemplo de datos requeridos, ya que el atributo Activo y Dirección no permiten valores nulos.

Nombre: DSitioSuceso			
Descripción: Almacena información del sitio del suceso			
Atributo	**Tipo**	**Requerido**	**Descripción**
ID	NUMERIC (15)	NOT NULL	Id del sitio del suceso (Llave primaria).
Nombre	VARCHAR (150)	NULL	Nombre del sitio del suceso.
Activo	BIT (1)	NOT NULL	Si esta activo o no.
Direccion	NUMERIC (15)	NOT NULL	Dirección asociada (FK).

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

A continuación se muestra un ejemplo de cómo el DBMS lanza el error cuando al tratar de insertar una letra en el campo subtipo de la tabla DInforme que se definió que es de tipo numeric.

Figura 6. Ejemplo de chequeo de validez.

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 DBMS 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.

A continuación se muestran dos secuencias creadas, en este caso para DKitMuestra y DPeculiaridadInd:

Figura 7. Ejemplo de secuencia de DKitMuestra.

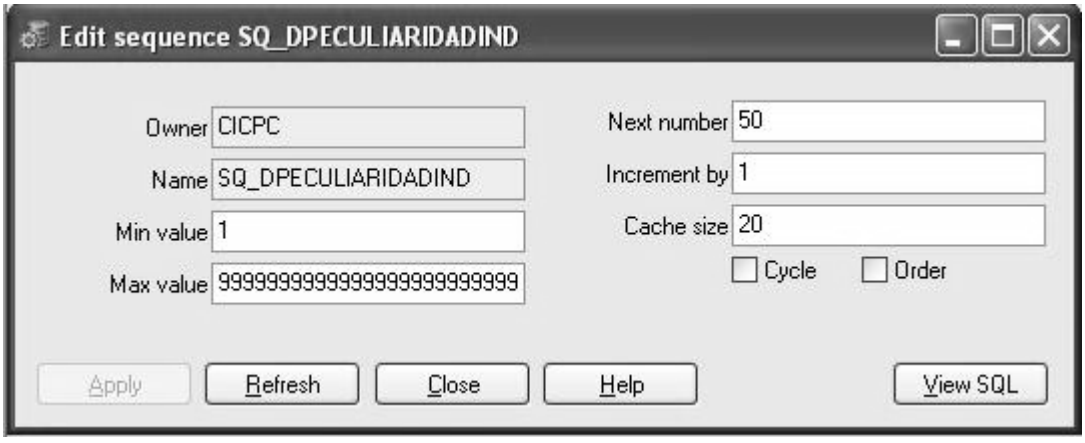

Figura 8. Ejemplo de secuencia de DPeculiaridadInd.

Integridad de dominio: La integridad de dominio viene dada por la validez de las entradas para una columna determinada. Puede exigir la integridad de dominio para restringir el tipo mediante tipos de datos, el formato mediante reglas y restricciones CHECK, o el intervalo de valores posibles mediante restricciones FOREIGN KEY, restricciones CHECK, definiciones DEFAULT, definiciones NOT NULL y reglas.

A continuación se muestran tres ejemplos de CHECK para las tablas DInforme, DSolicitud, DInfIdeIndividuo.

CHECK DINFORME

```
ALTER TABLE DINFORME
 ADD CONSTRAINT CK_DINFORME_NOINFOMRE
 CHECK (REGEXP_LIKE (NOINFORME, 'I-[0-9]{2}-[0-9]{3}-[0-9]{4}$'));
```

CHECK DSOLICITUD

```
ALTER TABLE DSOLICITUD
 ADD CONSTRAINT CK_DSOLICITUD_NOSOLICITUD
 CHECK (REGEXP_LIKE(NOSOLICITUD, 'S-[0-9]{2}-[0-9]{3}-[0-9]{4}$'));
```

CHECK DINFIDEINDIVIDUO

```
ALTER TABLE DINFIDEINDIVIDUO
 ADD CONSTRAINT CK_DINFIDEINDIVIDUO
 CHECK (LETRACED IN ('D','V'));
```

Integridad referencial: Asegura la integridad entre las claves ajenas y primarias (relaciones padre/hijo). Garantiza que cuando una entidad se vaya a relacionar con otras estas sean válidas.

3.1.2 Normalización de la Base de Datos.

La normalización de la BD es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener. Son una serie de reglas que sirven para ayudar a los diseñadores de BD a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede.

Las bases de datos relacionales se normalizan para: evitar la redundancia de los datos, evitar problemas de actualización de los datos en las tablas y para proteger la integridad de los datos.

Otra ventaja de la normalización de base de datos es el consumo de espacio. Una base de datos normalizada ocupa menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia un menor uso de espacio en disco (9).

Las formas normales son aplicadas a las tablas de una base de datos. Una base de datos está en la forma normal N es decir que todas sus tablas están en la forma normal N. No siempre es una buena idea tener una base de datos conformada en el nivel más alto de normalización, puede llevar a un nivel de complejidad que pudiera ser evitado si estuviera en un nivel más bajo de normalización.

Primera Forma Normal (1FN): Incluye la eliminación de todos los grupos repetidos, es decir, establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas. Es decir, prohíbe los atributos multivaluados, los atributos compuestos y sus combinaciones. Esto indica que todos los atributos deben tener valores atómicos.

Segunda Forma Normal (2FN): Asegura que todas las columnas que no son llave sean completamente dependientes de la llave primaria (PK), es decir, establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un término que describe a aquellos datos que no dependen de la llave primaria de la tabla para identificarlos.

Tercera Forma Normal (3FN): Elimina cualquier dependencia transitiva. Una dependencia transitiva es aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave. Una tabla está normalizada en esta forma si todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas.

Forma Normal de Boyce-Codd (FNBC): Es una versión ligeramente más fuerte de la Tercera forma normal (3FN). La forma normal de Boyce-Codd requiere que no existan dependencias funcionales no triviales de los atributos que no sean un conjunto de la clave candidata.

La base de datos del proyecto CICPC se decidió normalizarla hasta la Forma Normal de Boyce-Codd (FNBC). En el caso específico del módulo Investigación Criminalística todas sus tablas están en FNBC, ellas cumplen con todas las reglas de la 1FN, 2FN, 3FN y FNBC. En caso de que alguna tabla no cumpla con estas condiciones es debido a la búsqueda de rendimiento en la base de datos.

3.1.3 Análisis de redundancia de la información.

La mínima redundancia es una de las características básicas de las buenas bases de datos. Como se mostró en el epígrafe anterior la normalización ayuda a eliminar la redundancia de la información aunque no es posible evitarla al 100%, sus consecuencias son terribles:

- ✓ A más redundancia, más posibilidades de inconsistencias.
- ✓ Se duplica la información, requiriendo más espacio del necesario.

En el módulo de Investigación Criminalística se trató de eliminar en la medida de lo posible la redundancia de la información, pero siempre se nos quedan detalles de redundancia que la mayoría de los casos es para buscar rendimiento en la base de datos. Por ejemplo la tabla DInfIdeIndividuo tiene varios atributos de la persona y en lugar de establecer una relación directamente con DPersona, se hace de esta manera, ya que DPersona tiene 35 atributos de los cuales solamente se necesitan 6. Lo mismo ocurre con la tabla DInfRetHablado.

3.2Validación funcional.

3.2.1 Herramientas para un llenado voluminoso e inteligente de la base de datos.

Para llevar a cabo el llenado de los datos en las tablas de la BD, el PL/SQL Developer tiene la herramienta PL/SQL Data Generator, la cual tiene el propósito de almacenar datos aleatorios, garantizando la integridad referencial y teniendo en cuenta la validez de los datos en cada campo de las tablas. Una vez que la base de datos está completamente construida, se necesita hacerle pruebas. El objetivo de estas pruebas es poder simular una carga de producción real, conexiones del usuario y observar cómo se comporta la base de datos bajo cargas extremas. Esto permite solucionar los problemas potenciales de rendimiento, antes de poner la misma en producción. El propósito general es ejecutar las consultas y procedimientos almacenados más complejos y los que se supone que sean los más utilizados.

3.2.2 Descripción de las Pruebas

Para validar la solución propuesta se decidió realizar dos pruebas: Prueba de Stress y de Volumen. Para realizar dichas pruebas a esta base de datos se diseñó e implementó una aplicación en C# la cual simula un número de conexiones indicado por el usuario.

La Prueba de Volumen somete a la base de datos a grandes cantidades de datos para determinar si se alcanzan límites que causen la falla de la misma. Esta identifica la carga máxima continua que puede manejar la base de datos a prueba en un período dado.

Objetivo de la prueba

Verificar que el software funciona correctamente con volúmenes de datos grandes:

- ✓ Máximo (real o físicamente posible) número de clientes conectados, o simulados, todos realizando la misma operación (peor caso de operación) por un período de tiempo extenso.
- ✓ Máximo tamaño de base de datos y múltiples consultas ejecutadas simultáneamente.

A continuación se muestra la siguiente tabla con el resultado de la prueba de Volumen realizada a la base de datos para ello se ejecutaron cuatro procedimientos almacenados, primero con 500000 registros a consultar, luego con 1000000 y por último con 1500000.

Resultados de la Prueba de Volumen.

Procedimiento Almacenado	Registros Consultados	Registros devueltos	Tiempo	Nº Conexiones
SP_Buscar_Informe	500000	19334	0.655	50
	1000000	40.916	0.932	50
SP_Buscar_Solicitud	500000	17160	0.77	50
	1000000	40000	0.820	50
SP_Solicitudes_por_Tipo	500000	10	0.416	50
	1000000	10	0.745	50
SP_Informes_Por_Tipo	500000	10	0.537	50
	1000000	10	0.734	50

La prueba de esfuerzo es en un tipo de prueba implementada y ejecutada para encontrar errores cuando hay pocos recursos o cuando hay competencia por recursos. Poca memoria o poco espacio de disco pueden revelar fallas en la base de datos que no aparecen bajo condiciones normales de cantidad de recursos. Otras fallas pueden resultar al competir por recursos compartidos como bloqueos de bases de datos o ancho de banda de red. La prueba de esfuerzo también puede usarse para identificar el trabajo máximo que la base de datos puede manejar.

Objetivo de la prueba

Verificar que el software funciona apropiadamente y sin error bajo condiciones de esfuerzo, como son:

- ✓ Poca memoria o sin disponibilidad de memoria en el servidor.
- ✓ Cantidad máxima de clientes conectados.
- ✓ Múltiples usuarios realizando la misma operación sobre los mismos datos.

El objetivo de la prueba de esfuerzo es también identificar y documentar las condiciones bajo las cuales el sistema falla y no continúa funcionando apropiadamente.

A continuación se muestra el resultado de la prueba de Stress realizada a la base de datos. Para ello se ejecutaron cuatro procedimientos almacenados con diferentes números de conexiones, primero se ejecutaron con 2 conexiones, luego con 50 y por último con 100.

Resultados de la 1ra iteración de la prueba de Stress.

Procedimiento Almacenado	Nº Conexiones	Registros Consultados	Registros Devueltos	Tiempo
SP_Buscar_Informe	2	500000	19334	0.344
SP_Buscar_Solicitud	2	500000	17160	0.672
SP_Solicitudes_por_Tipo	2	500000	10	0.265
SP_Informes_por_Fecha	2	500000	20	0.34
SP_Informes_Por_Tipo	2	500000	10	0.459

Resultados de la 2da iteración de la prueba de Stress.

Procedimiento Almacenado	Nº Conexiones	Registros Consultados	Registros Devueltos	Tiempo
SP_Buscar_Informe	50	500000	19334	0.655
SP_Buscar_Solicitud	50	500000	17160	0.77
SP_Solicitudes_por_Tipo	50	500000	10	0.416
SP_Informes_por_Fecha	50	500000	20	0.603
SP_Informes_Por_Tipo	50	500000	10	0.537

Resultados de la 3ra iteración de la prueba de Stress.

Procedimiento Almacenado	Nº Conexiones	Registros Consultados	Registros Devueltos	Tiempo
SP_Buscar_Informe	100	500000	19334	0.844
SP_Buscar_Solicitud	100	500000	17160	0.916
SP_Solicitudes_por_Tipo	100	500000	10	0.844
SP_Informes_por_Fecha	100	500000	20	0.797
SP_Informes_Por_Tipo	100	500000	10	0.830

Las condiciones en que se realizaron las pruebas son: el servidor donde está montada la base de datos tiene 1.5 GB de memoria RAM y un procesador Intel Pentium IV a 3.00 GHz. El cliente usado para ejecutar las pruebas estaba instalado en máquinas con similares características sólo cambiaba en la memoria RAM que es de 1.0GB.

La aplicación actual que utiliza el CICPC, tiene como tiempo de respuesta máximo 0.48 segundos, siendo esto un requisito a cumplir por la nueva BD.

Para analizar el resultado de las pruebas y validar el cumplimiento del requisito anterior es importante considerar las condiciones que posee el sistema actual y las que tendrá el nuevo:

- ✓ La aplicación antigua usa una interfaz de texto; una base de datos ADABAS de alto rendimiento, por tener los datos en texto plano, logrando tiempos de respuesta muy bajos, incluso desde los puntos de acceso más lejanos que trabajan haciendo peticiones directamente al centro de datos (0,48 segundos). Por otro lado el servidor tiene 8 GB de RAM y está montado en tecnología de Sun con Sistema Operativo Solaris.
- ✓ El sistema nuevo debe contar con dos servidores con las siguientes características:
  - Servidor principal: HP Integrity RX 7640, Sistema Operativo: P-UX 11iv2, 40GB de RAM, 8 Procesadores Intel Itanium 2 núcleos, 1.8 tecnología IA 64Bit, más de 30 TB de Disco Duro
  - Servidor de contingencia: HP Integrity RX 7640, Sistema Operativo: P-UX 11iv2, 40GB de RAM, 2 Procesadores Intel Itanium 2 núcleos, 1.8 tecnología IA 64Bit 16.5 TB de Disco Duro

Después de realizadas las pruebas se concluye que la base de datos obtenida en este trabajo se encuentra en correspondencia con el objetivo propuesto.

3.2.3 Análisis de optimización de consultas.

Optimización es el proceso de elegir la vía más eficiente para resolver una consulta. (10) Cuando se habla de optimización de consultas se refiere a mejorar los tiempos de respuesta en un sistema de gestión de bases de datos relacional, pues la optimización es el proceso de modificar un sistema para mejorar su eficiencia o también el uso de los recursos disponibles.

En bases de datos relacionales el lenguaje de consultas SQL es el más utilizado por el común de los programadores y desarrolladores para obtener información desde la base de datos. La complejidad que pueden alcanzar algunas consultas puede ser tal, que el diseño de una consulta puede tomar un tiempo considerable, obteniendo no siempre una respuesta óptima

A continuación se muestran varios ejemplos con consultas pequeñas y sencillas, de cómo se pueden optimizar las mismas.

La siguiente consulta muestra el número de solicitud, la descripción, la fecha, el subtipo, el motivo y el estudio de las solicitudes de experticia criminalística que se hicieron en el 2008 utilizando la cláusula LIKE.

```
SELECT DSOLICITUD.NOSOLICITUD NOSOLICITUD,
  DSOLICITUD.DESCRIPCION,
  DSOLICITUD.FECHAREAL,
  NTSUBSOLICITUD.DESCRIPCION TIPOSOLICITUD,
  DSOLEXPERTICIA.MOTIVO MOTIVO,
  NTESTUDIO.DESCRIPCION ESTUDIO
FROM DSOLICITUD INNER JOIN DSOLEXPERTICIA
        INNER JOIN DSOLEXPERCRIMINALISTICA
         INNER JOIN NTESTUDIO
         ON NTESTUDIO.ID = DSOLEXPERCRIMINALISTICA.ESTUDIO
        ON DSOLEXPERCRIMINALISTICA.SOLICITUD = DSOLEXPERTICIA.SOLICITUD
       ON DSOLEXPERTICIA.SOLICITUD = DSOLICITUD.DILIGENCIA
       INNER JOIN NTSUBSOLICITUD
       ON NTSUBSOLICITUD.ID = DSOLICITUD.TIPO
WHERE TO_CHAR (DSOLICITUD.FECHAREAL, 'yyyy') LIKE ('2008')
```

La consulta se demoró 0.25 segundos, para devolver 15000 registros. A continuación se presenta otra forma de hacer esto utilizando la sentencia BETWEEN.

```
SELECT DSOLICITUD.NOSOLICITUD NOSOLICITUD,
   DSOLICITUD.DESCRIPCION,
   DSOLICITUD.FECHAREAL,
   NTSUBSOLICITUD.DESCRIPCION TIPOSOLICITUD,
   DSOLEXPERTICIA.MOTIVO MOTIVO,
```

```
    NTESTUDIO.DESCRIPCION ESTUDIO
FROM DSOLICITUD INNER JOIN DSOLEXPERTICIA
        INNER JOIN DSOLEXPERCRIMINALISTICA
         INNER JOIN NTESTUDIO
         ON NTESTUDIO.ID = DSOLEXPERCRIMINALISTICA.ESTUDIO
        ON DSOLEXPERCRIMINALISTICA.SOLICITUD = DSOLEXPERTICIA.SOLICITUD
       ON DSOLEXPERTICIA.SOLICITUD = DSOLICITUD.DILIGENCIA
       INNER JOIN NTSUBSOLICITUD
       ON NTSUBSOLICITUD.ID = DSOLICITUD.TIPO
WHERE DSOLICITUD.FECHAREAL BETWEEN TO_DATE ('2007', 'yyyy') AND TO_DATE ('2009', 'yyyy')
```

Esta demoró 0.063 segundos para devolver 15000 registros, como se pudo apreciar se mejoró considerablemente.

La consulta que aparece a continuación devuelve el número de solicitud, la descripción, la fecha, el subtipo, el motivo y el estudio de las solicitudes de experticia criminalística que están entre el subtipo 1 y 5.

```
SELECT DSOLICITUD.NOSOLICITUD NOSOLICITUD,
    DSOLICITUD.DESCRIPCION,
    DSOLICITUD.FECHAREAL,
    NTSUBSOLICITUD.DESCRIPCION TIPOSOLICITUD,
    DSOLEXPERTICIA.MOTIVO MOTIVO,
    NTESTUDIO.DESCRIPCION ESTUDIO
FROM DSOLICITUD INNER JOIN DSOLEXPERTICIA
        INNER JOIN DSOLEXPERCRIMINALISTICA
         INNER JOIN NTESTUDIO
         ON NTESTUDIO.ID = DSOLEXPERCRIMINALISTICA.ESTUDIO
        ON DSOLEXPERCRIMINALISTICA.SOLICITUD = DSOLEXPERTICIA.SOLICITUD
       ON DSOLEXPERTICIA.SOLICITUD = DSOLICITUD.DILIGENCIA
       INNER JOIN NTSUBSOLICITUD
       ON NTSUBSOLICITUD.ID = DSOLICITUD.TIPO
WHERE NTSUBSOLICITUD.ID >= 1 AND NTSUBSOLICITUD.ID <= 5
```

Esta consulta se demoró 0.062 segundos, para devolver 7500 registros. A continuación se muestra otra variante utilizando la sentencia BETWEN:

```
SELECT DSOLICITUD.NOSOLICITUD NOSOLICITUD,
    DSOLICITUD.DESCRIPCION,
    DSOLICITUD.FECHAREAL,
    NTSUBSOLICITUD.DESCRIPCION TIPOSOLICITUD,
    DSOLEXPERTICIA.MOTIVO MOTIVO,
    NTESTUDIO.DESCRIPCION ESTUDIO
FROM DSOLICITUD INNER JOIN DSOLEXPERTICIA
        INNER JOIN DSOLEXPERCRIMINALISTICA
         INNER JOIN NTESTUDIO
         ON NTESTUDIO.ID = DSOLEXPERCRIMINALISTICA.ESTUDIO
```

```
        ON DSOLEXPERCRIMINALISTICA.SOLICITUD = DSOLEXPERTICIA.SOLICITUD
       ON DSOLEXPERTICIA.SOLICITUD = DSOLICITUD.DILIGENCIA
       INNER JOIN NTSUBSOLICITUD
       ON NTSUBSOLICITUD.ID = DSOLICITUD.TIPO
WHERE NTSUBSOLICITUD.ID BETWEEN 1 AND 5
```

Esta variante se demoró 0.043 segundos en devolver 7500 registros, ahorrando 0.019 segundos con respecto a la anterior.

La siguiente consulta muestra el número de solicitud, la fecha, el subtipo, el motivo y el estudio de las solicitudes de experticia criminalística que son de tipo 1, 2 ó 3.

```
SELECT DSOLICITUD.NOSOLICITUD NOSOLICITUD,
   DSOLICITUD.DESCRIPCION,
   DSOLICITUD.FECHAREAL,
   NTSUBSOLICITUD.DESCRIPCION TIPOSOLICITUD,
   DSOLEXPERTICIA.MOTIVO MOTIVO,
   NTESTUDIO.DESCRIPCION ESTUDIO
FROM DSOLICITUD INNER JOIN DSOLEXPERTICIA
        INNER JOIN DSOLEXPERCRIMINALISTICA
         INNER JOIN NTESTUDIO
         ON NTESTUDIO.ID = DSOLEXPERCRIMINALISTICA.ESTUDIO
        ON DSOLEXPERCRIMINALISTICA.SOLICITUD = DSOLEXPERTICIA.SOLICITUD
       ON DSOLEXPERTICIA.SOLICITUD = DSOLICITUD.DILIGENCIA
       INNER JOIN NTSUBSOLICITUD
       ON NTSUBSOLICITUD.ID = DSOLICITUD.TIPO
WHERE DSOLICITUD.TIPO = 1
      OR DSOLICITUD.TIPO = 2
      OR DSOLICITUD.TIPO = 3
```

Esta consulta se demoró 0.062 segundos en devolver 4500 registros, la siguiente consulta utiliza la cláusula IN.

```
SELECT DSOLICITUD.NOSOLICITUD NOSOLICITUD,
   DSOLICITUD.DESCRIPCION,
   DSOLICITUD.FECHAREAL,
   NTSUBSOLICITUD.DESCRIPCION TIPOSOLICITUD,
   DSOLEXPERTICIA.MOTIVO MOTIVO,
   NTESTUDIO.DESCRIPCION ESTUDIO
FROM DSOLICITUD INNER JOIN DSOLEXPERTICIA
        INNER JOIN DSOLEXPERCRIMINALISTICA
         INNER JOIN NTESTUDIO
         ON NTESTUDIO.ID = DSOLEXPERCRIMINALISTICA.ESTUDIO
        ON DSOLEXPERCRIMINALISTICA.SOLICITUD = DSOLEXPERTICIA.SOLICITUD
       ON DSOLEXPERTICIA.SOLICITUD = DSOLICITUD.DILIGENCIA
       INNER JOIN NTSUBSOLICITUD
       ON NTSUBSOLICITUD.ID = DSOLICITUD.TIPO
```

```
WHERE DSOLICITUD.TIPO IN (1, 2, 3)
```

Esta variante se tardó 0.043 segundos en retornar 4500 registros.

De los resultados obtenidos se concluyó que si se tiene un dominio bastante amplio de las maneras de hacer las consultas y con las herramientas que permitan calcular tiempos de ejecución de dichas consultas, se pueden hacer peticiones al servidor de Base de Datos con mejores resultados en cuanto al tiempo de respuesta.

Conclusiones.

En el capítulo se han analizado varios aspectos y consideraciones que se deben tener en cuenta para realizar un diseño eficiente de una BD, como son: la integridad del diseño de la BD, la redundancia de información presente en la misma, además de la normalización, la cual en la BD propuesta se llevó hasta la FNBC. Además se realizó un llenado de la BD con la herramienta PL/SQL Data Generator, seleccionándose algunos procedimientos almacenados que se valoran con mayor grado de ocurrencia, utilizándose para la realización de pruebas. Por último se mostraron una serie de consultas como ejemplo de la optimización de las mismas.

Conclusiones Generales

En el presente trabajo ha sido propuesto el modelo lógico y físico de los módulos de Investigación Criminalística y Estadística de la base de datos de CICPC

Luego del estudio de las herramientas más usadas para diseñar bases de datos se concluyó que ORACLE es el SGBD más adecuado para resolver el problema planteado debido a las características y ventajas que posee, además de haber sido seleccionado por el propio cliente, se utilizaron además otras herramientas como el ER/Studio 7.0 y el PL/SQL Developer.

Como parte de la propuesta de solución se generaron una serie de diagramas que permiten visualizar de forma más entendible dicha solución, estos diagramas son: diagramas de clases persistentes, diagramas Entidad-Relación empleando para ello el Visual Paradigm y el ER/Studio 7.0 como herramientas de modelado. Además se programaron varios procedimientos almacenados utilizando como herramienta el PL/SQL Developer. Se validaron los diseños realizados; tanto teórico como funcional, donde se analizó la integridad de la información, la normalización de las tablas, la redundancia de los datos y también se realizaron pruebas para demostrar la consistencia de la base de datos realizada.

Con la terminación de este trabajo se puede plantear que se le ha dado cumplimiento a todos los objetivos del mismo.

Recomendaciones

Con la realización del presente trabajo podemos recomendar:

- ✓ Refinamiento constante del modelo lógico propuesto, durante los ciclos de desarrollo restantes del proyecto CICPC.
- ✓ El empleo del gestor Oracle para almacenar y gestionar la información en otros sistemas.

Referencias Bibliográficas.

1. **desarrollo, Ministerio del Poder Popular para la planificación y el.** Problemas Sociales en Venezuela. . *Problemas Sociales en Venezuela.* [En línea] [Citado el: 09 de 04 de 2008.] http://fegs.msinfo.info/opac/php/documento_presentar.php?back=S&problema=Delitos+con.

2. **Justicia, Ministerio del Poder Popular para Relaciones Interiores y.** Cuerpo de Investigaciones Científicas, Penales y Criminalísticas. *Cuerpo de Investigaciones Científicas, Penales y Criminalísticas.* [En línea] [Citado el: 09 de 04 de 2008.] http://www.cicpc.gov.ve/.

3. **Date C., J.** *Introducción a los sistemas de bases de datos, Capítulos 1, 2, 13.* La Habana : Félix Varela, 2003.

4. **Marqués, Mercedes.** Apuntes de Ficheros y Bases de Datos. *Apuntes de Ficheros y Bases de Datos.* [En línea] [Citado el: 10 de 04 de 2008.] http://www3.uji.es/~mmarques/f47/apun/apun.pdf.

5. Oracle SQL Y PL/SQL. [En línea] [Citado el: 10 de 04 de 2008.] http://mioracle.blogspot.com/2008/02/tunning-optimizacin.html.

6. Sistema Gestor de Base Datos . [En línea] 04 de 2004. [Citado el: 10 de 04 de 2008.] http://www.error500.net/garbagecollector/bases_de_datos/sistema_gestor_de_base_de_dato.php .

7. Sitio de Descargas de Software. *Sitio de Descargas de Software.* [En línea] [Citado el: 15 de 02 de 2008.] http://www.freedownloadmanager.org/es/downloads/SME_Gerente_de_SQL_2007_para_Or%C3%A1culo_46847_p/..

8. DanySoft. *DanySoft.* [En línea] [Citado el: 10 de 02 de 2008.] http://www.danysoft.com/bol/050310.htm.

9. **Eduardo.** Normalización de bases de datos. *Normalización de bases de datos.* [En línea] 30 de 3 de 2003. [Citado el: 18 de 2 de 2008.] http://www.mysql-hispano.org/page.php?id=16&pag=1.

10. **Obando Torres, Milton Antonio.** OPTIMIZACION DE CONSULTAS SQL. *OPTIMIZACION DE CONSULTAS SQL.* [En línea] [Citado el: 18 de 2 de 2008.] http://www.emagister.com/frame.cfm?id_centro=579530300529575648666666952674548&id_curso=4654710007116648495555515265455 2&url_frame=http://www.emagister.com/uploads_courses/Comunidad_Emagister_62485_62485.pdf.

11. **Sanchez, María A. Mendoza.** Metodologías De Desarrollo De Software. [En línea] 07 de 06 de 2007. [Citado el: 07 de 02 de 2008.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

12. **Guevara Injoque, Marco A. y Flores Nazario, César R.** *Conceptos Básicos de Modelamineto Lógico.* 2002.

13. **Hansen, Gary W. y Hansen, James V.** *Diseño y administración de Bases de Datos.* España : Prentice Hall, 2000. Disponbible en http://bibliodoc.uci.cu/pdf/reg00071.pdf.

14. **Rob, Peter y Coronel, Carlos.** *Sistemas de bases de datos: Diseño, implementación y Administración.* s.l. : Thomson Learning Ibero, 2002. ISBN 9706862862.

15. Sistemas Gestores de Bases de Datos. *Sistemas Gestores de Bases de Datos.* [En línea] [Citado el: 1 de 3 de 2008.] http://www.objetiweb.com/sgbd/?p=5.

Bibliografía

1. Matthew Hart & Scott Jesse. Oracle Database 10g: High Availablity with RAC Flashback & Data Guard. ISBN:0072254289
2. Jason Price. Oracle Database 10g SQL. ISBN:0072229810
3. Dan Tow. SQL Tuning, noviembre de 2003. ISBN: 0-596-00573-3
4. Konrad King and Kris Jamsa, SQL Tips and Techniques. ISBN:1931841454
5. Alex Kriegel & Boris M. Trukhnov SQL Bible. ISBN:0764525840
6. Kroenke, Dr. David M. (2000). Database Processing—Fundamentals, Design, & Design, Seventh Edition. Upper Saddle River, NJ: Prentice Hall.
7. Hoffer, Jeffrey A., Mary B. Prescott, and Fred R. McFadden. (2002). Modern Database Management, Sixth Edition. Upper Saddle River, NJ: Prentice Hall.
8. C.J.Date (1994), *"An Introduction to Database Systems"*, Addison-Wesley.
9. Piattini Mario, Adoración de Miguel, Marcos Esperanza. DISEÑO DE BASES DE DATOS RELACIONALES. Ed. Alfaomega
10. Ronald Fagin (1981) A Normal Form for Relational Databases That Is Based on Domains and Keys, Communications of the ACM, vol. 6, pp. 387-415

Glosario de Términos

CICPC: Cuerpo de Investigaciones Científicas, Penales y Criminalísticas de la República de Venezuela.

MPIJ: la Sala de Prensa del Ministerio del Poder Popular para las relaciones del Interior y Justicia

SGBD: Sistema de Gestión de Bases de Datos.

BD: Bases de Datos.

SQL: Lenguaje de consulta estructurado (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones.

PL/SQL: Lenguaje de programación embebido en Oracle. Soporta todas las consultas y manipulación de datos que se usan en SQL, pero incluye nuevas características:

JOIN: Un JOIN cláusula combina registros de dos tablas en una base de datos relacional

EXISTS: Es un operador especial que simplemente verifica si la consulta interna arroja alguna fila. Si lo hace, entonces la consulta externa procede. De no hacerlo, la consulta externa no se ejecuta, y la totalidad de la instrucción

Atributo: Es cualidades, propiedades o características de un elemento.

Automatización: Ejecución automática de tareas industriales, administrativas o científicas haciendo más ágil y efectivo el trabajo y ayudando al ser humano.

Escalabilidad: La capacidad del sistema informático de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.

Confiabilidad: Posibilidad que tiene un sistema de realizar las funciones para las que fue diseñado sin fallos.

Interfaz: Una interfaz es la parte de un programa informático que permite a éste comunicarse con el usuario o con otras aplicaciones permitiendo el flujo de información.

Normalización: Proceso de reducción sobre una estructura de datos que procura aumentar la integridad, disminuir la redundancia y las dependencias funcionales de esa estructura.

Redundancia: Repetición de la información.

PK: Llave primaria

FK: Lave foránea.

Inconsistencia: Falta de consistencia, Una base de datos está inconsistente si dos datos que deberían ser iguales no lo son.

Cliente-Servidor: Esta arquitectura consiste básicamente en que un programa, el Cliente informático realiza peticiones a otro programa, el servidor.

Modelo E-R: Modelo de diseño de base de datos gráfica, que nos muestra información relativa a los datos y la relación existente entre ellos.

Experticia: Es un informe donde se plasma los resultados obtenidos al realizar un análisis pericial.

Comisión: Expertos adscritos al departamento que se desplacen a un determinado lugar con el fin de realizar estudio correspondiente.

Informe Pericial: Es el informe donde se deja plasmado todo lo relacionado con los análisis efectuados a las evidencias.

Montaje fotográfico: Es un montaje de las fotos tomadas en el sitio de suceso con una leyenda correspondiente a cada foto.

Solicitudes: Se encuentra relacionado con el tipo de experticias que ha de realizarse a un tipo de evidencia.

Anexos

Anexo 1. Descripción de las clases persistentes.

Clase 33. NEstadoCertificadoSeguridad

Nombre: NEstadoCertificadoSeguridad		
	Atributo	**Tipo**
1	descripcion	String

Clase 34. NCatalogo

Nombre: NCatalogo		
	Atributo	**Tipo**
1	nombre	String
2	valor	String

Clase 35. NIndiceProximidad

Nombre: NIndiceProximidad		
	Atributo	**Tipo**
1	indice	String

Clase 36. NTipoFabricacion

Nombre: NTipoFabricacion		
	Atributo	**Tipo**
1	tipo	String

Clase 37. SolicitudExperticiaCriminalistica

Nombre: SolicitudExperticiaCriminalistica		
	Atributo	**Tipo**

Clase 38. SolicitudExperticiaCriminalisticaRoboABanco

Nombre: SolicitudExperticiaCriminalisticaRoboABanco		
	Atributo	**Tipo**

Clase 39. TipoInforme

Nombre: TipoInforme		
	Atributo	**Tipo**
1	tipo	String

Clase 40. NTipoAvaluo

Nombre: NTipoAvaluo		
	Atributo	**Tipo**
1	tipo	String

Clase 41. NTipoMarcaImagen

Nombre: NTipoMarcaImagen		
	Atributo	**Tipo**
1	nombre	String

Clase 42. NTrayectoIntraorganico

Nombre: NTrayectoIntraorganico		
	Atributo	**Tipo**
1	tipoTrayecto	String

Clase 43. SubInforme

Nombre: SubInforme		
	Atributo	**Tipo**
2	nombre	String

Clase 44. SubSolicitud

Nombre: SubSolicitud		
	Atributo	**Tipo**
1	nombre	String

Clase 45. TipoSolicitud

Nombre: TipoSolicitud		
	Atributo	**Tipo**
1	tipo	String

Clase 46. NTipoResenna

Nombre: NTipoResenna		
	Atributo	**Tipo**
1	descripcion	String

Clase 47. NTipoEstudioSobreExperticia

Nombre: NTipoEstudioSobreExperticia		
	Atributo	**Tipo**
1	nombre	String

Clase 48. NOrigenSolicitud

Nombre: NOrigenSolicitud		
	Atributo	**Tipo**
1	origen	String

Anexo 2. Descripción de las tablas de la Base de Datos.

Tabla 45. NCatalogo

Nombre: NCatalogo		
Descripción: Almacena información de los catálogos.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (15)	Id del catalogo (Llave primaria).
Activo	BIT (1)	Si está activo o no.
Nombre	VARCHAR (50)	Nombre del catalogo.
Descripcion	VARCHAR (250)	Descripción del catalogo.

Tabla 46. NEstCertificado

Nombre: NEstCertificado		
Descripción: Almacena la información de los estados certificados de seguridad.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del estado certificado de seguridad (Llave primaria).
Descripcion	VARCHAR (60)	Descripción del estado certificado de seguridad.

Tabla 47. NIndiceAProc

Nombre: NIndiceAProc		
Descripción: Almacena la información de los índices de aproximación.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del índice de aproximación (Llave primaria).
Descripcion	VARCHAR (60)	Descripción del índice de aproximación.

Tabla 48. NTFabricacion

Nombre: NTFabricacion		
Descripción: Almacena la información de los tipos de fabricación.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (2)	Id del tipo de fabricación (Llave primaria).
Descripcion	VARCHAR (30)	Descripción del tipo de fabricación.

Tabla 49. NTInforme

Nombre: NTInforme		
Descripción: Almacena los datos de los tipos de informe.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del tipo de informe (Llave primaria).
Descripcion	VARCHAR (250)	Descripción del tipo de informe.

Tabla 50. NTipoAvaluo

Nombre: NTipoAvaluo		
Descripción: Almacena los tipos de informe de avaluó.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del tipo informe de avaluó (Llave primaria).
Descripcion	VARCHAR (30)	Descripción del tipo informe de avaluó.

Tabla 51. NTipoMarca

Nombre: NTipoMarca		
Descripción: Almacena los tipos de marca.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del tipo de marca (Llave primaria).
Descripcion	VARCHAR (30)	Descripción del tipo de marca.

Tabla 52. NTTrayecto

Nombre: NTTrayecto		
Descripción: Almacena los dados de los trayectos		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del trayecto (Llave primaria).
Descripcion	VARCHAR (30)	Descripción del trayecto.

Tabla 53. NTSubInforme

Nombre: NTSubInforme		
Descripción: Almacena los dados de los subtipos de informe.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del subtipo de informe (Llave primaria).
Descripcion	VARCHAR (30)	Descripción del subtipo de informe.
Tipo	NUMERIC (5)	Id del tipo (FK).

Tabla 54. NTSubSolicitud

Nombre: NTSubSolicitud		
Descripción: Almacena los subtipos de los tipos de las solicitudes		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id de los subtipos de solicitudes (Llave primaria).
Tipo	NUMERIC (5)	Tipo asociado (FK).
Descripcion	VARCHAR (30)	Descripción de los subtipos.

Tabla 55. NTSolicitud

Nombre: NTSolicitud		
Descripción: Almacena información de los posibles tipos de solicitud.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id de los tipos de solicitudes (Llave primaria).
Descripcion	VARCHAR (150)	Descripción de los tipos de solicitudes.

Tabla 56. NTResenna

Nombre: NTResenna		
Descripción: Almacena los tipos de reseña.		
Atributo	**Tipo**	**Descripción**
Id	NUMERIC (5)	Id de la reseña (Llave primaria).
Descripcion	VARCHAR (60)	Descripción de la reseña.

Tabla 57. NTEstudio

Nombre: NTEstudio		
Descripción: Almacena los tipos de estudios que se le hacen a las experticias criminalísticas.		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del estudio (Llave primaria).
TipoSolicitud	NUMERIC (5)	Tipo de solicitud asociada.
Descripcion	VARCHAR (150)	Descripción del estudio.

Tabla 58. NOrigenSolicitud

Nombre: NOrigenSolicitud		
Descripción: Almacena los datos de los orígenes de las solicitudes de experticia criminalística		
Atributo	**Tipo**	**Descripción**
ID	NUMERIC (5)	Id del origen (Llave primaria).
Descripcion	VARCHAR (200)	Descripción del origen.

Anexo 3. Descripción de los principales procedimientos almacenados.

Procedimiento almacenado SP_Delitos_Totales

```
PROCEDURE SP_DELITOS_TOTALES(IO_ALLREC OUT REF_CURSOR,
                        FECHA_INICIO IN DDELITO.FECHA%TYPE,
                        FECHA_FIN IN DDELITO.FECHA%TYPE)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO  DDELITO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DDELITO.FECHA%TYPE := FECHA_FIN;
BEGIN
  OPEN SINGLEREC FOR
    SELECT
      DDELITO.MODUSOPERANDI MODUSOPERANDI,
       TO_CHAR(DDELITO.FECHA,'YYYY-MM-DD') FECHA,
       TO_CHAR(DDELITO.HORA , 'HH:MI:SS AM') HORA,
       NTDELITO.DESCRIPCION TTIPODELITO,
       NNIVELIMPORTANCIA.IMPORTANCIA NIVELIMPORTANCIA,
       NNATURALEZADELITO.NATURALEZA NATURALEZADELITO,
       NGRADODELITO.GRADO GRADODELITO,
       DDIRECCION.CASERIO CASERIO,
       DDIRECCION.CALLE CALLE,
       DDIRECCION.NORESIDENCIA NORESIDENCIA,
       DDIRECCION.ENTRECALLEINF ENTRECALLES,
       DDIRECCION.ENTRECALLESUP ENTRECALLESSUP,
       NMUNICIPIO.DESCRIPCION MUNICIPIO,
       NPARROQUIA.DESCRIPCION PARROQUIA,
       NESTADO.DESCRIPCION ESTADO,
       NTDIRECCION.DESCRIPCION DIRECCION,
       NTRESIDENCIA.DESCRIPCION RESIDENCIA
    FROM DDELITO INNER JOIN DSITIOSUCESO
          LEFT JOIN DDIRECCION
           INNER JOIN NMUNICIPIO
           ON DDIRECCION.MUNICIPIO = NMUNICIPIO.ID
           LEFT JOIN NTRESIDENCIA
           ON DDIRECCION.TRESIDENCIA = NTRESIDENCIA.ID
           LEFT JOIN NPARROQUIA
           ON DDIRECCION.PARROQUIA = NPARROQUIA.ID
           INNER JOIN NESTADO
           ON DDIRECCION.ESTADO = NESTADO.ID
           LEFT JOIN NTDIRECCION
           ON DDIRECCION.TDIRECCION = NTDIRECCION.ID
          ON DDIRECCION.ID = DSITIOSUCESO.DIRECCION
         ON DSITIOSUCESO.ID = DDELITO.SITIOSUCESO
          LEFT JOIN NGRADODELITO
          ON DDELITO.GRADO = NGRADODELITO.ID
          LEFT JOIN NTDELITO
          ON DDELITO.TIPODELITO = NTDELITO.ID
```

```
          LEFT JOIN NNATURALEZADELITO
          ON DDELITO.NATURALEZADELITO = NNATURALEZADELITO.ID
          LEFT JOIN NNIVELIMPORTANCIA
          ON DDELITO.NIVELIMPORTANCIA = NNIVELIMPORTANCIA.ID
    WHERE DDELITO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
       AND DDELITO.ACTIVO = 1
    ORDER BY DDELITO.FECHA ASC, TO_CHAR(DDELITO.HORA , 'HH:MI:SS AM')
ASC ;
  IO_ALLREC := SINGLEREC;
END SP_DELITOS_TOTALES;
```

Procedimiento almacenado SP_Mayor_Indice_Delictivo

```
PROCEDURE SP_DIAS_MAYOR_INDICE_DELICTIVO(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DDELITO.FECHA%TYPE,
                FECHA_FIN IN DDELITO.FECHA%TYPE,
                CANT_MOSTRAR IN NUMERIC DEFAULT NULL)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DDELITO.FECHA%TYPE :=FECHA_INICIO;
 PRM_FECHA_FIN DDELITO.FECHA%TYPE := FECHA_FIN;
 PRM_CANT NUMERIC(5) := CANT_MOSTRAR;
 VVALOR_MINIMO NUMERIC(10);
BEGIN
  SELECT MIN(D.C ) INTO VVALOR_MINIMO
  FROM
   (SELECT T.FECHA F, T.CANTIDAD C
    FROM
     (SELECT DDELITO.FECHA FECHA, COUNT(DDELITO.ID) CANTIDAD
     FROM DDELITO
     GROUP BY DDELITO.FECHA
     ORDER BY CANTIDAD DESC) T
    WHERE ROWNUM <=PRM_CANT) D;
  OPEN SINGLEREC FOR
    SELECT T.CANT CANTDELITOS, T.DDIAS DIA
    FROM
      (SELECT COUNT (DDELITO.ID) CANT , TO_CHAR(DDELITO.FECHA, 'YYYY-
MM-DD') DDIAS
      FROM DDELITO
      WHERE DDELITO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
        AND DDELITO.ACTIVO = 1
      GROUP BY DDELITO.FECHA
      ORDER BY CANT DESC) T
    WHERE ROWNUM <= PRM_CANT OR T.CANT = VVALOR_MINIMO OR
PRM_CANT IS NULL;
```

```
   IO_ALLREC := SINGLEREC;
END SP_DIAS_MAYOR_INDICE_DELICTIVO;
```

Procedimiento almacenado SP_Mas_Frecuentes

```
PROCEDURE SP_DELITOS_MAS_FRECUENTES(IO_ALLREC OUT REF_CURSOR,
                FECHA_INICIO IN DDELITO.FECHA%TYPE,
                FECHA_FIN IN DDELITO.FECHA%TYPE)
 IS
  SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
  PRM_FECHA_INICIO DDELITO.FECHA%TYPE :=FECHA_INICIO;
  PRM_FECHA_FIN DDELITO.FECHA%TYPE := FECHA_FIN;
 BEGIN
   OPEN SINGLEREC FOR

     SELECT COUNT(DDELITO.ID) CANT,
       NPARROQUIA.DESCRIPCION PARROQUIA,
       NMUNICIPIO.DESCRIPCION MUNICICPIO ,
       NESTADO.DESCRIPCION ESTADO,
       NTDELITO.DESCRIPCION DELITO
     FROM DDELITO INNER JOIN DSITIOSUCESO
          LEFT JOIN DDIRECCION
           LEFT JOIN NPARROQUIA
           ON NPARROQUIA.ID = DDIRECCION.PARROQUIA
           INNER JOIN NMUNICIPIO
           ON NMUNICIPIO.ID = DDIRECCION.MUNICIPIO
           INNER JOIN NESTADO
           ON NESTADO.ID = DDIRECCION.ESTADO
          ON DDIRECCION.ID = DSITIOSUCESO.DIRECCION
          ON DSITIOSUCESO.ID = DDELITO.SITIOSUCESO
          LEFT JOIN NTDELITO
          ON NTDELITO.ID = DDELITO.TIPODELITO
     WHERE DDELITO.FECHA BETWEEN PRM_FECHA_INICIO AND PRM_FECHA_FIN
     AND DDELITO.ACTIVO = 1
     GROUP BY
NPARROQUIA.DESCRIPCION,NTDELITO.DESCRIPCION,NMUNICIPIO.DESCRIPCION,NESTADO.
DESCRIPCION
     ORDER BY CANT DESC;
   IO_ALLREC := SINGLEREC;
END SP_DELITOS_MAS_FRECUENTES;
```

Procedimiento almacenado SP_Horas_Mayor_Incidencia

```
PROCEDURE SP_HORAS_MAYOR_INCIDENCIA(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DDELITO.FECHA%TYPE,
                FECHA_FIN IN DDELITO.FECHA%TYPE,
                CANT_MOSTRAR IN NUMERIC DEFAULT NULL)
```

```
 IS
  SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
  PRM_FECHA_INICIO DDELITO.FECHA%TYPE :=FECHA_INICIO;
  PRM_FECHA_FIN DDELITO.FECHA%TYPE := FECHA_FIN;
  PRM_CANT NUMERIC(5) := CANT_MOSTRAR;
 BEGIN
   OPEN SINGLEREC FOR
     SELECT T.CANTDELITO CANT, T.HHORA HORA
     FROM
       (SELECT COUNT(DDELITO.ID) CANTDELITO , TO_CHAR(DDELITO.HORA,
'HH AM') HHORA
       FROM DDELITO
       WHERE DDELITO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
         AND DDELITO.ACTIVO = 1
       GROUP BY TO_CHAR(DDELITO.HORA, 'HH AM')
       ORDER BY CANTDELITO DESC) T
     WHERE ROWNUM <= PRM_CANT OR PRM_CANT IS NULL;
    IO_ALLREC := SINGLEREC;
 END SP_HORAS_MAYOR_INCIDENCIA;
```

Procedimiento almacenado SP_Buscar_Delitos_Criteria

```
PROCEDURE SP_BUSCAR_DELITO_CRITERIA(IO_ALLREC OUT REF_CURSOR,
                CRITERIO IN VARCHAR2 DEFAULT NULL,
                FECHA_INICIO IN DDELITO.FECHA%TYPE,
                FECHA_FIN IN DDELITO.FECHA%TYPE)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_CADENA1 VARCHAR2(32767) := CRITERIO;
 PRM_CADENA2 VARCHAR2(32767);
 PRM_FECHA_INICIO DDELITO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DDELITO.FECHA%TYPE := FECHA_FIN;
 SENTENCIA VARCHAR2(32767) ;
BEGIN
   IF(PRM_CADENA1 IS NULL)THEN
    PRM_CADENA2 := '1=1';
    ELSIF(PRM_CADENA1 IS NOT NULL)THEN
    PRM_CADENA2 := FN_PARSEAR_CADENA_ESTADISTICAS(PRM_CADENA1);
   END IF;
   SENTENCIA := 'SELECT
           DDELITO.MODUSOPERANDI MODUSOPERANDI,
           TO_CHAR(DDELITO.FECHA,"YYYY-MM-DD") FECHA,
           TO_CHAR(DDELITO.HORA , "HH:MI:SS AM") HORA,
           NTDELITO.DESCRIPCION TTIPODELITO,
           NNIVELIMPORTANCIA.IMPORTANCIA NIVELIMPORTANCIA,
           NNATURALEZADELITO.NATURALEZA NATURALEZADELITO,
           NGRADODELITO.GRADO GRADODELITO,
```

```
          DDIRECCION.CASERIO CASERIO,
          DDIRECCION.CALLE CALLE,
          DDIRECCION.NORESIDENCIA NORESIDENCIA,
          DDIRECCION.ENTRECALLEINF ENTRECALLES,
          DDIRECCION.ENTRECALLESUP ENTRECALLESSUP,
          NMUNICIPIO.DESCRIPCION MUNICIPIO,
          NPARROQUIA.DESCRIPCION PARROQUIA,
          NESTADO.DESCRIPCION ESTADO,
          NTDIRECCION.DESCRIPCION DIRECCION,
          NTRESIDENCIA.DESCRIPCION RESIDENCIA
        FROM DDELITO INNER JOIN DSITIOSUCESO
             LEFT JOIN DDIRECCION
              INNER JOIN NMUNICIPIO
              ON DDIRECCION.MUNICIPIO = NMUNICIPIO.ID
              LEFT JOIN NTRESIDENCIA
              ON DDIRECCION.TRESIDENCIA = NTRESIDENCIA.ID
              LEFT JOIN NPARROQUIA
              ON DDIRECCION.PARROQUIA = NPARROQUIA.ID
              INNER JOIN NESTADO
              ON DDIRECCION.ESTADO = NESTADO.ID
              LEFT JOIN NTDIRECCION
              ON DDIRECCION.TDIRECCION = NTDIRECCION.ID
             ON DDIRECCION.ID = DSITIOSUCESO.DIRECCION
            ON DSITIOSUCESO.ID = DDELITO.SITIOSUCESO
            LEFT JOIN NGRADODELITO
            ON DDELITO.GRADO = NGRADODELITO.ID
            LEFT JOIN NTDELITO
            ON DDELITO.TIPODELITO = NTDELITO.ID
            LEFT JOIN NNATURALEZADELITO
            ON DDELITO.NATURALEZADELITO =   NNATURALEZADELITO.ID
            LEFT JOIN NNIVELIMPORTANCIA
            ON DDELITO.NIVELIMPORTANCIA = NNIVELIMPORTANCIA.ID
        WHERE (DDELITO.ACTIVO = 1) AND DDELITO.FECHA BETWEEN '
||''''||PRM_FECHA_INICIO ||''''||' AND '||''''||PRM_FECHA_FIN||''''||' AND ';
  SENTENCIA := SENTENCIA || PRM_CADENA2 ;
  OPEN SINGLEREC FOR
    SENTENCIA;
  IO_ALLREC:=SINGLEREC;
END SP_BUSCAR_DELITO_CRITERIA;
```

Procedimiento almacenado SP_Casos_Iniciados_Por_Delegacion

```
PROCEDURE SP_CASOS_INICIADOS_POR_DELEG(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DCASO.FECHA%TYPE,
                FECHA_FIN IN DCASO.FECHA%TYPE,
                DEPENCENCIA DENTIDAD.NOMBRE%TYPE,
                CANT_MOSTRAR IN NUMERIC DEFAULT NULL)
```

```
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DCASO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DCASO.FECHA%TYPE := FECHA_FIN;
 PRM_DEPENDENCIA DENTIDAD.NOMBRE%TYPE := DEPENCENCIA;
 PRM_CANT NUMERIC(5) := CANT_MOSTRAR;
BEGIN

  OPEN SINGLEREC FOR
   SELECT DISTINCT
      T.FFECHAAPERTURA FECHAAPERTURA,
      T.HHORAAPERTURA HORAAPERTURA,
      T.FFECHACONCLUCION FECHACONCLUCION,
      T.NNOACTAPROCESAL NOACTAPROCESAL,
      T.EESTADOCASO ESTADOCASO,
      T.FFISCAL FISCAL,
      T.NNOFISCALIA NOFISCALIA,
      T.CCASOIMPORTANCIA CASOIMPORTANCIA,
      T.FFUNCIONARIODENUNCIA FUNCIONARIODENUNCIA,
      T.TTIPODENUNCIA TIPODENUNCIA,
      T.TTIPODELITO TIPODELITO,
      T.DDEPENDENCIA DEPENDENCIA,
      T.TTIPODEPENDENCIA TTIPODEPENDENCIA
    FROM
     (SELECT
        TO_CHAR(DCASO.FECHA, 'YYYY-MM-DD') FFECHAAPERTURA,
        TO_CHAR(DCASO.HORA, 'HH:MI:SS AM') HHORAAPERTURA,
        TO_CHAR(DCASO.FECHACONCLUSION, 'YYYY-MM-DD')
FFECHACONCLUCION,
        DCASO.NOACTAPROCESAL NNOACTAPROCESAL,
        NESTADOCASO.ESTADO EESTADOCASO,
        (FISCAL.PNOMBRE || ' ' || FISCAL.SNOMBRE || ' ' || FISCAL.PAPELLIDO || ' ' ||
FISCAL.SAPELLIDO) FFISCAL,
        DFISCALIA.NOFISCALIA NNOFISCALIA,
        NCASOIMPORTANCIA.DESCRIPCION CCASOIMPORTANCIA,
        (CASE WHEN DDENUNCIA.TIPO = 1 THEN 'DENUNCIA'
          WHEN DDENUNCIA.TIPO = 0 THEN 'OFICIO'
        END) TTIPODENUNCIA,
        (DENUNCIANTE.PNOMBRE || ' ' || DENUNCIANTE.SNOMBRE || ' ' ||
DENUNCIANTE.PAPELLIDO || ' ' || DENUNCIANTE.SAPELLIDO)
FFUNCIONARIODENUNCIA,
        NTDELITO.DESCRIPCION TTIPODELITO,
        DENTIDAD.NOMBRE DDEPENDENCIA,
        NTDEPENDENCIA.DESCRIPCION TTIPODEPENDENCIA
     FROM DCASO      INNER JOIN NESTADOCASO
              ON NESTADOCASO.ID = DCASO.ESTADO
              LEFT JOIN DFISCAL
               INNER JOIN DPERSONA FISCAL
               ON FISCAL.ID = DFISCAL.PERSONA
```

```
            ON DFISCAL.PERSONA = DCASO.FISCAL
            INNER JOIN DFISCALIA
            ON DFISCALIA.ENTIDAD = DCASO.FISCALIA
            LEFT JOIN NCASOIMPORTANCIA
            ON NCASOIMPORTANCIA.ID = DCASO.CASOIMPORTANCIA
            LEFT JOIN DFUNCIONARIOCASO
             INNER JOIN DFUNCIONARIO
              INNER JOIN DPERSONA FUNCIONARIOCASO
              ON FUNCIONARIOCASO.ID = DFUNCIONARIO.PERSONA
              LEFT JOIN DDEPENDENCIA
               INNER JOIN DENTIDAD
                LEFT JOIN NTDEPENDENCIA
                 ON NTDEPENDENCIA.ID = DENTIDAD.TIPO
                ON DENTIDAD.ID = DDEPENDENCIA.ENTIDAD
               ON DDEPENDENCIA.ENTIDAD = DFUNCIONARIO.DEPENDENCIA
             ON DFUNCIONARIO.PERSONA =
DFUNCIONARIOCASO.FUNCIONARIO
            ON DFUNCIONARIOCASO.CASO = DCASO.ID
            LEFT JOIN DDENUNCIA
             INNER JOIN DFUNCIONARIO FUNDENUNCIA
              INNER JOIN DPERSONA DENUNCIANTE
              ON DENUNCIANTE.ID = FUNDENUNCIA.PERSONA
             ON FUNDENUNCIA.PERSONA = DDENUNCIA.FUNCIONARIO
             INNER JOIN DDELITO
              INNER JOIN NTDELITO
              ON NTDELITO.ID = DDELITO.TIPODELITO
             ON DDELITO.ID = DDENUNCIA.DELITO
            ON DDENUNCIA.ID = DCASO.DENUNCIA
    WHERE DCASO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
       AND UPPER(DENTIDAD.NOMBRE) = UPPER(PRM_DEPENDENCIA)
       AND UPPER(NESTADOCASO.ESTADO) = UPPER('INICIADO')
       AND DCASO.ACTIVO = 1
    ORDER BY DCASO.FECHA ASC, TO_CHAR(DCASO.HORA, 'HH:MI:SS
AM')ASC) T
   WHERE ROWNUM <= PRM_CANT OR PRM_CANT IS NULL;
  IO_ALLREC := SINGLEREC;
END SP_CASOS_INICIADOS_POR_DELEG;
```

Procedimiento almacenado SP_Iniciados_Discriminados_Por_Delito

```
PROCEDURE SP_CASOS_INIC_DISCRI_DELITO(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DCASO.FECHA%TYPE,
                FECHA_FIN IN DCASO.FECHA%TYPE,
                TIPO_DELITO NTDELITO.DESCRIPCION%TYPE,
                CANT_MOSTRAR IN NUMERIC DEFAULT NULL)
IS
```

```
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DCASO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DCASO.FECHA%TYPE := FECHA_FIN;
 PRM_TIPO_DELITO NTDELITO.DESCRIPCION%TYPE := TIPO_DELITO;
 PRM_CANT NUMERIC(5) := CANT_MOSTRAR;
BEGIN
  OPEN SINGLEREC FOR
   SELECT DISTINCT
      T.FFECHAAPERTURA FECHAAPERTURA,
      T.HHORAAPERTURA HORAAPERTURA,
      T.FFECHACONCLUCION FECHACONCLUCION,
      T.NNOACTAPROCESAL NOACTAPROCESAL,
      T.EESTADOCASO ESTADOCASO,
      T.FFISCAL FISCAL,
      T.NNOFISCALIA NOFISCALIA,
      T.CCASOIMPORTANCIA CASOIMPORTANCIA,
      T.FFUNCIONARIODENUNCIA FUNCIONARIODENUNCIA,
      T.TTIPODENUNCIA TIPODENUNCIA,
      T.TTIPODELITO TIPODELITO,
      T.DDEPENDENCIA DEPENDENCIA,
      T.TTIPODEPENDENCIA TTIPODEPENDENCIA
    FROM
     (SELECT
       TO_CHAR(DCASO.FECHA, 'YYYY-MM-DD') FFECHAAPERTURA,
       TO_CHAR(DCASO.HORA, 'HH:MI:SS AM') HHORAAPERTURA,
       TO_CHAR(DCASO.FECHACONCLUSION, 'YYYY-MM-DD')
FFECHACONCLUCION,
       DCASO.NOACTAPROCESAL NNOACTAPROCESAL,
       NESTADOCASO.ESTADO EESTADOCASO,
       (FISCAL.PNOMBRE || ' ' || FISCAL.SNOMBRE || ' ' || FISCAL.PAPELLIDO || ' ' ||
FISCAL.SAPELLIDO) FFISCAL,
       DFISCALIA.NOFISCALIA NNOFISCALIA,
       NCASOIMPORTANCIA.DESCRIPCION CCASOIMPORTANCIA,
       (CASE WHEN DDENUNCIA.TIPO = 1 THEN 'DENUNCIA'
         WHEN DDENUNCIA.TIPO = 0 THEN 'OFICIO'
       END) TTIPODENUNCIA,
       (DENUNCIANTE.PNOMBRE || ' ' || DENUNCIANTE.SNOMBRE || ' ' ||
DENUNCIANTE.PAPELLIDO || ' ' || DENUNCIANTE.SAPELLIDO)
FFUNCIONARIODENUNCIA,
       NTDELITO.DESCRIPCION TTIPODELITO,
       DENTIDAD.NOMBRE DDEPENDENCIA,
       NTDEPENDENCIA.DESCRIPCION TTIPODEPENDENCIA
    FROM DCASO LEFT JOIN NESTADOCASO
         ON NESTADOCASO.ID = DCASO.ESTADO
         LEFT JOIN DFISCAL
          INNER JOIN DPERSONA FISCAL
          ON FISCAL.ID = DFISCAL.PERSONA
         ON DFISCAL.PERSONA = DCASO.FISCAL
         LEFT JOIN DFISCALIA
```

```
          ON DFISCALIA.ENTIDAD = DCASO.FISCALIA
          LEFT JOIN NCASOIMPORTANCIA
          ON NCASOIMPORTANCIA.ID = DCASO.CASOIMPORTANCIA
          LEFT JOIN DFUNCIONARIOCASO
           INNER JOIN DFUNCIONARIO
            INNER JOIN DPERSONA FUNCIONARIOCASO
            ON FUNCIONARIOCASO.ID = DFUNCIONARIO.PERSONA
            INNER JOIN DDEPENDENCIA
             INNER JOIN DENTIDAD
              INNER JOIN NTDEPENDENCIA
              ON NTDEPENDENCIA.ID = DENTIDAD.TIPO
             ON DENTIDAD.ID = DDEPENDENCIA.ENTIDAD
            ON DDEPENDENCIA.ENTIDAD = DFUNCIONARIO.DEPENDENCIA
           ON DFUNCIONARIO.PERSONA = DFUNCIONARIOCASO.FUNCIONARIO
          ON DFUNCIONARIOCASO.CASO = DCASO.ID
          LEFT JOIN DDENUNCIA
           INNER JOIN DFUNCIONARIO
            INNER JOIN DPERSONA DENUNCIANTE
            ON DENUNCIANTE.ID = DFUNCIONARIO.PERSONA
           ON DFUNCIONARIO.PERSONA = DDENUNCIA.FUNCIONARIO
           INNER JOIN DDELITO
            INNER JOIN NTDELITO
            ON NTDELITO.ID = DDELITO.TIPODELITO
           ON DDELITO.ID = DDENUNCIA.DELITO
          ON DDENUNCIA.ID = DCASO.DENUNCIA
    WHERE DCASO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
       AND UPPER(NTDELITO.DESCRIPCION) = UPPER(PRM_TIPO_DELITO)
       AND UPPER(NESTADOCASO.ESTADO) = UPPER('INICIADO')
       AND DCASO.ACTIVO = 1
     ORDER BY DCASO.FECHA ASC, TO_CHAR(DCASO.HORA, 'HH:MI:SS
AM')ASC) T
   WHERE ROWNUM <= PRM_CANT OR PRM_CANT IS NULL;
  IO_ALLREC := SINGLEREC;
END SP_CASOS_INIC_DISCRI_DELITO;
```

Procedimiento almacenado SP_Diligencia_Por_Delegacion

```
PROCEDURE SP_DILIGENCIAS_POR_DELEGACION(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DDILIGENCIA.FECHAHORA%TYPE,
                FECHA_FIN IN DDILIGENCIA.FECHAHORA%TYPE,
                DEPENDENCIA IN DENTIDAD.NOMBRE%TYPE)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DDILIGENCIA.FECHAHORA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DDILIGENCIA.FECHAHORA%TYPE := FECHA_FIN;
 PRM_DEPENDENCIA DENTIDAD.NOMBRE%TYPE := DEPENDENCIA;
```

```
BEGIN
  OPEN SINGLEREC FOR
   SELECT DISTINCT
      T.FFECHADILIGENCIA FECHADILIGENCIA,
      T.TTIPODILIGENCIA TIPODILIGENCIA,
      T.NNOACTAPROCESAL NOACTAPROCESAL,
      T.EESTADOCASO ESTADOCASO,
      T.CCASOIMPORTANCIA CASOIMPORTANCIA,
      T.DDEPENDENCIA DEPENDENCIA,
      T.TTIPODEPENDENCIA TIPODEPENDENCIA
    FROM
    (SELECT
       DDILIGENCIA.ID IIDDILIGENCIA,
       TO_CHAR(DDILIGENCIA.FECHAHORA, 'YYYY-MM-DD HH:MI:SS AM')
FFECHADILIGENCIA,
       NTDILIGENCIA.TIPO TTIPODILIGENCIA,
       DCASO.ID IIDCASO,
       DCASO.NOACTAPROCESAL NNOACTAPROCESAL,
       NESTADOCASO.ESTADO EESTADOCASO,
       NCASOIMPORTANCIA.DESCRIPCION CCASOIMPORTANCIA,
       DENTIDAD.NOMBRE DDEPENDENCIA,
       NTDEPENDENCIA.DESCRIPCION TTIPODEPENDENCIA
    FROM DDILIGENCIA INNER JOIN NTDILIGENCIA
          ON NTDILIGENCIA.ID = DDILIGENCIA.TIPO
          LEFT JOIN DCASO
           INNER JOIN NESTADOCASO
           ON NESTADOCASO.ID = DCASO.ESTADO
           INNER JOIN NCASOIMPORTANCIA
           ON NCASOIMPORTANCIA.ID = DCASO.CASOIMPORTANCIA
           LEFT JOIN DFUNCIONARIOCASO
            INNER JOIN DFUNCIONARIO
             INNER JOIN DPERSONA
             ON DPERSONA.ID = DFUNCIONARIO.PERSONA
             INNER JOIN DDEPENDENCIA
              INNER JOIN DENTIDAD
               LEFT JOIN NTDEPENDENCIA
               ON NTDEPENDENCIA.ID = DENTIDAD.TIPO
              ON DENTIDAD.ID = DDEPENDENCIA.ENTIDAD
             ON DDEPENDENCIA.ENTIDAD = DFUNCIONARIO.DEPENDENCIA
            ON DFUNCIONARIO.PERSONA = DFUNCIONARIOCASO.FUNCIONARIO
           ON DFUNCIONARIOCASO.CASO = DCASO.ID
          ON DCASO.ID = DDILIGENCIA.CASO
    WHERE DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
       AND UPPER(DENTIDAD.NOMBRE) = UPPER(PRM_DEPENDENCIA)
       AND DDILIGENCIA.ACTIVO = 1
    ORDER BY TO_CHAR(DDILIGENCIA.FECHAHORA, 'YYYY-MM-DD')) T ;
  IO_ALLREC := SINGLEREC;
END SP_DILIGENCIAS_POR_DELEGACION;
```

Procedimiento almacenado SP_Funcionarios_Mas_Casos_Llevan

```
PROCEDURE SP_FUNCIONA_MAS_CASOS_LLEVAN(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DCASO.FECHA%TYPE,
                FECHA_FIN IN DCASO.FECHA%TYPE,
                DEPENDENCIA IN DENTIDAD.NOMBRE%TYPE,
                CANT IN NUMERIC DEFAULT NULL)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DCASO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DCASO.FECHA%TYPE := FECHA_FIN;
 PRM_DEPENDENCIA DENTIDAD.NOMBRE%TYPE := DEPENDENCIA;
 PRM_CANT NUMERIC(5) := CANT;
BEGIN
  OPEN SINGLEREC FOR
    SELECT
      T.CANT1 CANT,
      T.FUN FUNCIONARIO,
      T.RRANGO RANGO,
      T.CCARGO CARGO,
      T.CCREDENCIAL CREDENCIAL,
      T.DDDEPENDENCIA DEPENDENCIA,
      T.TTIPODEPENDENCIA TIPODEPENDENCIA
    FROM
     (SELECT
       COUNT(DCASO.ID) CANT1,
       (DPERSONA.PNOMBRE||' '||DPERSONA.SNOMBRE||'
'||DPERSONA.PAPELLIDO||' '||DPERSONA.SAPELLIDO) FUN,
       NRANGO.RANGO RRANGO,
       NCARGO.CARGO CCARGO,
       DFUNCIONARIO.CREDENCIAL CCREDENCIAL,
       DENTIDAD.NOMBRE DDDEPENDENCIA,
       NTDEPENDENCIA.DESCRIPCION TTIPODEPENDENCIA
     FROM DCASO INNER JOIN DFUNCIONARIOCASO
          INNER JOIN DFUNCIONARIO
           INNER JOIN DPERSONA
           ON DPERSONA.ID = DFUNCIONARIO.PERSONA
           LEFT JOIN NRANGO
           ON NRANGO.ID = DFUNCIONARIO.RANGO
           LEFT JOIN NCARGO
           ON NCARGO.ID = DFUNCIONARIO.CARGO
           INNER JOIN DDEPENDENCIA
            INNER JOIN DENTIDAD
             INNER JOIN NTDEPENDENCIA
             ON NTDEPENDENCIA.ID = DENTIDAD.TIPO
            ON DENTIDAD.ID = DDEPENDENCIA.ENTIDAD
```

```
            ON DDEPENDENCIA.ENTIDAD = DFUNCIONARIO.DEPENDENCIA
           ON DFUNCIONARIO.PERSONA = DFUNCIONARIOCASO.FUNCIONARIO
          ON DFUNCIONARIOCASO.CASO = DCASO.ID
          INNER JOIN NESTADOCASO
          ON NESTADOCASO.ID = DCASO.ESTADO
          LEFT JOIN NCASOIMPORTANCIA
          ON NCASOIMPORTANCIA.ID = DCASO.CASOIMPORTANCIA
      WHERE DCASO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
        AND UPPER(DENTIDAD.NOMBRE) = UPPER(PRM_DEPENDENCIA)
        AND DCASO.ACTIVO = 1
      GROUP BY (DPERSONA.PNOMBRE||' '||DPERSONA.SNOMBRE||'
'||DPERSONA.PAPELLIDO||' '||DPERSONA.SAPELLIDO), NRANGO.RANGO ,
NCARGO.CARGO , DFUNCIONARIO.CREDENCIAL, DENTIDAD.NOMBRE,
NTDEPENDENCIA.DESCRIPCION
      ORDER BY CANT1 DESC) T
   WHERE ROWNUM <= PRM_CANT OR PRM_CANT IS NULL;
  IO_ALLREC := SINGLEREC;
END SP_FUNCIONA_MAS_CASOS_LLEVAN;
```

Procedimiento almacenado SP_Diligencias_Mas_Hacen

```
PROCEDURE SP_DILIGENCIAS_MAS_HACEN(IO_ALLREC OUT REF_CURSOR,
              FECHA_INICIO IN DDILIGENCIA.FECHAHORA%TYPE,
              FECHA_FIN IN DDILIGENCIA.FECHAHORA%TYPE,
              CANT IN NUMERIC DEFAULT NULL)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DDILIGENCIA.FECHAHORA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DDILIGENCIA.FECHAHORA%TYPE := FECHA_FIN;
 PRM_CANT NUMERIC(5) := CANT;
BEGIN
  OPEN SINGLEREC FOR
    SELECT R.CANTDILIGENCIA CANT , R.TIPODILIGENCIA TIPO,
R.SUBTIPODILIGENCIA SUBTIPO
    FROM
    (SELECT T.CANT CANTDILIGENCIA, T.TIPO TIPODILIGENCIA, T.SUBTIPO
SUBTIPODILIGENCIA
    FROM
     ((SELECT COUNT(DDILIGENCIA.ID) CANT , NTDILIGENCIA.TIPO TIPO ,
NTSUBINFORME.DESCRIPCION SUBTIPO FROM DDILIGENCIA LEFT JOIN
NTDILIGENCIA ON NTDILIGENCIA.ID = DDILIGENCIA.TIPO
                                    LEFT JOIN DINFORME ON
DINFORME.DILIGENCIA = DDILIGENCIA.ID
                                     LEFT JOIN NTSUBINFORME ON
NTSUBINFORME.ID = DINFORME.SUBTIPO
      WHERE UPPER(NTDILIGENCIA.TIPO) = UPPER('INFORME') AND
DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
```

```
PRM_FECHA_FIN
      GROUP BY NTDILIGENCIA.TIPO,NTSUBINFORME.DESCRIPCION)

      UNION

      (SELECT COUNT(DDILIGENCIA.ID) CANT ,NTDILIGENCIA.TIPO TIPO ,
NTSUBSOLICITUD.DESCRIPCION SUBTIPO FROM DDILIGENCIA LEFT JOIN
NTDILIGENCIA ON NTDILIGENCIA.ID = DDILIGENCIA.TIPO
                                          LEFT JOIN DSOLICITUD ON
DSOLICITUD.DILIGENCIA = DDILIGENCIA.ID
                                          LEFT JOIN NTSUBSOLICITUD ON
NTSUBSOLICITUD.ID = DSOLICITUD.TIPO
      WHERE UPPER(NTDILIGENCIA.TIPO) = UPPER('SOLICITUD') AND
DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
      GROUP BY NTDILIGENCIA.TIPO,NTSUBSOLICITUD.DESCRIPCION)

      UNION

     (SELECT COUNT(DDILIGENCIA.ID) CANT ,NTDILIGENCIA.TIPO TIPO,
NTORDEN.TIPO SUBTIPO FROM DDILIGENCIA LEFT JOIN NTDILIGENCIA ON
NTDILIGENCIA.ID = DDILIGENCIA.TIPO
                                    LEFT JOIN DORDEN ON DORDEN.DILIGENCIA
= DDILIGENCIA.ID
                                     LEFT JOIN NTORDEN ON NTORDEN.ID =
DORDEN.TIPO
     WHERE UPPER(NTDILIGENCIA.TIPO) = UPPER('ORDEN') AND
DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
     GROUP BY NTDILIGENCIA.TIPO , NTORDEN.TIPO)

     UNION

     (SELECT COUNT(DDILIGENCIA.ID) CANT , NTDILIGENCIA.TIPO TIPO, "'NO
EXISTE"' SUBTIPO FROM DDILIGENCIA LEFT JOIN NTDILIGENCIA ON
NTDILIGENCIA.ID = DDILIGENCIA.TIPO
                                    LEFT JOIN DENTREVISTA ON
DENTREVISTA.DILIGENCIA = DDILIGENCIA.ID
     WHERE UPPER(NTDILIGENCIA.TIPO) = UPPER('ENTREVISTA') AND
DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
     GROUP BY NTDILIGENCIA.TIPO)

     UNION

    (SELECT COUNT(DDILIGENCIA.ID) CANT , NTDILIGENCIA.TIPO TIPO, "'NO
EXISTE"' SUBTIPO FROM DDILIGENCIA LEFT JOIN NTDILIGENCIA ON
NTDILIGENCIA.ID = DDILIGENCIA.TIPO
                                    LEFT JOIN DACTINVESPENAL ON
```

```
DACTINVESPENAL.DILIGENCIA = DDILIGENCIA.ID
    WHERE UPPER(NTDILIGENCIA.TIPO) = UPPER('ACTA DE INVESTIGACION
PENAL') AND DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
    GROUP BY NTDILIGENCIA.TIPO)

    UNION

    (SELECT COUNT(DDILIGENCIA.ID) CANT , NTDILIGENCIA.TIPO TIPO, '"NO
EXISTE"' SUBTIPO FROM DDILIGENCIA LEFT JOIN NTDILIGENCIA ON
NTDILIGENCIA.ID = DDILIGENCIA.TIPO
                                    LEFT JOIN DACTAINSPTECNICA ON
DACTAINSPTECNICA.DILIGENCIA = DDILIGENCIA.ID
    WHERE UPPER(NTDILIGENCIA.TIPO) = UPPER('ACTA DE IMPECCION
TECNICA') AND DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
    GROUP BY NTDILIGENCIA.TIPO)

    UNION

    (SELECT COUNT(DDILIGENCIA.ID) CANT , NTDILIGENCIA.TIPO TIPO, '"NO
EXISTE"' SUBTIPO FROM DDILIGENCIA LEFT JOIN NTDILIGENCIA ON
NTDILIGENCIA.ID = DDILIGENCIA.TIPO
                                    LEFT JOIN DBOLETACITACION ON
DBOLETACITACION.DILIGENCIA = DDILIGENCIA.ID
    WHERE UPPER(NTDILIGENCIA.TIPO) = UPPER('BOLETA DE CITACION') AND
DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
    GROUP BY NTDILIGENCIA.TIPO)

    UNION

   (SELECT COUNT(DDILIGENCIA.ID) CANT , NTDILIGENCIA.TIPO TIPO, '"NO
EXISTE"' SUBTIPO FROM DDILIGENCIA LEFT JOIN NTDILIGENCIA ON
NTDILIGENCIA.ID = DDILIGENCIA.TIPO
                                    LEFT JOIN DINSPLUGSUCESO ON
DINSPLUGSUCESO.DILIGENCIA = DDILIGENCIA.ID
    WHERE UPPER(NTDILIGENCIA.TIPO) = UPPER('INSPECCION DE LUGAR DE
SUCESO') AND DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
    GROUP BY NTDILIGENCIA.TIPO)

   )T
    ORDER BY CANTDILIGENCIA DESC, TIPODILIGENCIA ASC) R
  WHERE ROWNUM <= PRM_CANT OR PRM_CANT IS NULL;

 IO_ALLREC := SINGLEREC;
END SP_DILIGENCIAS_MAS_HACEN;
```

Procedimiento almacenado SP_Funcionarios_Casos_Sin_Cerrar

```
PROCEDURE SP_FUNCIONARI_CASOS_SIN_CERRAR(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DCASO.FECHA%TYPE,
                FECHA_FIN IN DCASO.FECHA%TYPE,
                DEPENDENCIA IN DENTIDAD.NOMBRE%TYPE)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DCASO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DCASO.FECHA%TYPE := FECHA_FIN;
 PRM_DEPENDENCIA DENTIDAD.NOMBRE%TYPE := DEPENDENCIA;
BEGIN
  OPEN SINGLEREC FOR
    SELECT
      COUNT(DCASO.ID) CANTCASOSSINCERRAR,
      (DPERSONA.PNOMBRE||' '||DPERSONA.SNOMBRE||'
'||DPERSONA.PAPELLIDO||' '||DPERSONA.SAPELLIDO) FUNCIONARIO,
      DFUNCIONARIO.CREDENCIAL CREDENCIAL,
      NRANGO.RANGO RANGO,
      NCARGO.CARGO CARGO,
      DENTIDAD.NOMBRE DEPENDENCIA,
      NTDEPENDENCIA.DESCRIPCION TIPODEPENDENCIA
   FROM DCASO
   INNER JOIN DDEPENDENCIA
     INNER JOIN DENTIDAD
      INNER JOIN NTDEPENDENCIA
      ON NTDEPENDENCIA.ID = DENTIDAD.TIPO
     ON DENTIDAD.ID = DDEPENDENCIA.ENTIDAD
    ON DDEPENDENCIA.ENTIDAD = DCASO.DEPENDENCIA
   INNER JOIN DFUNCIONARIOCASO
          INNER JOIN DFUNCIONARIO
           INNER JOIN DPERSONA
           ON DPERSONA.ID = DFUNCIONARIO.PERSONA
           LEFT JOIN NRANGO
           ON NRANGO.ID = DFUNCIONARIO.RANGO
           LEFT JOIN NCARGO
           ON NCARGO.ID = DFUNCIONARIO.CARGO
          ON DFUNCIONARIO.PERSONA = DFUNCIONARIOCASO.FUNCIONARIO
         ON DFUNCIONARIOCASO.CASO = DCASO.ID
        INNER JOIN NESTADOCASO
         ON NESTADOCASO.ID = DCASO.ESTADO
         LEFT JOIN NCASOIMPORTANCIA
         ON NCASOIMPORTANCIA.ID = DCASO.CASOIMPORTANCIA
   WHERE DCASO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
     AND UPPER(DENTIDAD.NOMBRE) = UPPER(PRM_DEPENDENCIA)
     AND UPPER(NESTADOCASO.ESTADO) <> UPPER('CONCLUIDO')
     AND DCASO.FECHACONCLUSION IS NULL
```

```
      AND DCASO.ACTIVO = 1
    GROUP BY (DPERSONA.PNOMBRE||' '||DPERSONA.SNOMBRE||'
'||DPERSONA.PAPELLIDO||' '||DPERSONA.SAPELLIDO) , NRANGO.RANGO,
NCARGO.CARGO, DFUNCIONARIO.CREDENCIAL, DENTIDAD.NOMBRE,
NTDEPENDENCIA.DESCRIPCION
    ORDER BY CANTCASOSSINCERRAR DESC;
   IO_ALLREC := SINGLEREC;
END SP_FUNCIONARI_CASOS_SIN_CERRAR;
```

Procedimiento almacenado SP_Casos_Mas_Rapido_Cerrado

```
PROCEDURE SP_CASOS_MAS_RAPIDO_CERRADO(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DCASO.FECHA%TYPE,
                FECHA_FIN IN DCASO.FECHA%TYPE,
                DEPENDENCIA IN DENTIDAD.NOMBRE%TYPE,
                CANT_MOSTAR IN NUMERIC DEFAULT NULL)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DCASO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DCASO.FECHA%TYPE := FECHA_FIN;
 PRM_DEPENDENCIA DENTIDAD.NOMBRE%TYPE := DEPENDENCIA;
 PRM_CANT NUMERIC(5) := CANT_MOSTAR;
BEGIN
   OPEN SINGLEREC FOR
    SELECT DISTINCT
      T.TTIEMPO_EN_DIAS TIEMPO_EN_DIAS,
      T.FFECHAAPERTURA FECHAAPERTURA,
      T.HHORAAPERTURA HORAAPERTURA,
      T.FFECHACONCLUCION FECHACONCLUCION,
      T.NNOACTAPROCESAL NOACTAPROCESAL,
      T.EESTADOCASO ESTADOCASO,
      T.FFISCAL FISCAL,
      T.NNOFISCALIA NOFISCALIA,
      T.CCASOIMPORTANCIA CASOIMPORTANCIA,
      T.FFUNCIONARIODENUNCIA FUNCIONARIODENUNCIA,
      T.TTIPODENUNCIA TIPODENUNCIA,
      T.TTIPODELITO TIPODELITO,
      T.DDEPENDENCIA DEPENDENCIA,
      T.TTIPODEPENDENCIA TIPODEPENDENCIA
    FROM
     (SELECT
       DCASO.FECHACONCLUSION - DCASO.FECHA TTIEMPO_EN_DIAS,
       TO_CHAR(DCASO.FECHA, 'YYYY-MM-DD') FFECHAAPERTURA,
       TO_CHAR(DCASO.HORA, 'HH:MI:SS AM') HHORAAPERTURA,
       TO_CHAR(DCASO.FECHACONCLUSION, 'YYYY-MM-DD')
FFECHACONCLUCION,
       DCASO.NOACTAPROCESAL NNOACTAPROCESAL,
```

```
        NESTADOCASO.ESTADO EESTADOCASO,
        (FISCAL.PNOMBRE || ' ' || FISCAL.SNOMBRE || ' ' || FISCAL.PAPELLIDO || ' ' ||
FISCAL.SAPELLIDO) FFISCAL,
        DFISCALIA.NOFISCALIA NNOFISCALIA,
        NNIVELIMPORTANCIA.IMPORTANCIA CCASOIMPORTANCIA,
        (CASE WHEN DDENUNCIA.TIPO = 1 THEN 'DENUNCIA'
          WHEN DDENUNCIA.TIPO = 0 THEN 'OFICIO'
        END) TTIPODENUNCIA,
        (DENUNCIANTE.PNOMBRE || ' ' || DENUNCIANTE.SNOMBRE || ' ' ||
DENUNCIANTE.PAPELLIDO || ' ' || DENUNCIANTE.SAPELLIDO)
FFUNCIONARIODENUNCIA,
        NTDELITO.DESCRIPCION TTIPODELITO,
        DENTIDAD.NOMBRE DDEPENDENCIA,
        NTDEPENDENCIA.DESCRIPCION TTIPODEPENDENCIA
    FROM DCASO
    INNER JOIN DDEPENDENCIA
      INNER JOIN DENTIDAD
       INNER JOIN NTDEPENDENCIA
       ON NTDEPENDENCIA.ID = DENTIDAD.TIPO
      ON DENTIDAD.ID = DDEPENDENCIA.ENTIDAD
    ON DDEPENDENCIA.ENTIDAD = DCASO.DEPENDENCIA
    LEFT JOIN NESTADOCASO
         ON NESTADOCASO.ID = DCASO.ESTADO
         LEFT JOIN DFISCAL
          INNER JOIN DPERSONA FISCAL
          ON FISCAL.ID = DFISCAL.PERSONA
         ON DFISCAL.PERSONA = DCASO.FISCAL
         LEFT JOIN DFISCALIA
         ON DFISCALIA.ENTIDAD = DCASO.FISCALIA
         LEFT JOIN DFUNCIONARIOCASO
          INNER JOIN DFUNCIONARIO
           INNER JOIN DPERSONA FUNCIONARIOCASO
           ON FUNCIONARIOCASO.ID = DFUNCIONARIO.PERSONA
          ON DFUNCIONARIO.PERSONA = DFUNCIONARIOCASO.FUNCIONARIO
         ON DFUNCIONARIOCASO.CASO = DCASO.ID
         LEFT JOIN DDENUNCIA
          INNER JOIN DFUNCIONARIO
           INNER JOIN DPERSONA DENUNCIANTE
           ON DENUNCIANTE.ID = DFUNCIONARIO.PERSONA
          ON DFUNCIONARIO.PERSONA = DDENUNCIA.FUNCIONARIO
          INNER JOIN DDELITO
            INNER JOIN NNIVELIMPORTANCIA
            ON NNIVELIMPORTANCIA.ID = DDELITO.NIVELIMPORTANCIA
            INNER JOIN NTDELITO
            ON NTDELITO.ID = DDELITO.TIPODELITO
          ON DDELITO.ID = DDENUNCIA.DELITO
         ON DDENUNCIA.ID = DCASO.DENUNCIA
    WHERE DCASO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
```

```
      AND DCASO.FECHACONCLUSION IS NOT NULL
      AND UPPER(DENTIDAD.NOMBRE) = UPPER(PRM_DEPENDENCIA)
      AND DCASO.ACTIVO = 1
    ) T
   WHERE ROWNUM <= PRM_CANT OR PRM_CANT IS NULL
   ORDER BY TTIEMPO_EN_DIAS ASC;
  IO_ALLREC := SINGLEREC;
END SP_CASOS_MAS_RAPIDO_CERRADO;
```

Procedimiento almacenado SP_Funcionarios_Rapido_Cerrado_Casos

```
PROCEDURE SP_FUNC_RAPIDO_CERRADO_CASOS(IO_ALLREC OUT
REF_CURSOR,
                FECHA_INICIO IN DCASO.FECHA%TYPE,
                FECHA_FIN IN DCASO.FECHA%TYPE,
                DEPENDENCIA IN DENTIDAD.NOMBRE%TYPE,
                CANT_MOSTAR IN NUMERIC DEFAULT NULL)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_FECHA_INICIO DCASO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DCASO.FECHA%TYPE := FECHA_FIN;
 PRM_DEPENDENCIA DENTIDAD.NOMBRE%TYPE := DEPENDENCIA;
 PRM_CANT NUMERIC(5) := CANT_MOSTAR;
BEGIN
  OPEN SINGLEREC FOR
    SELECT
      T.TTIEMPO_EN_DIAS TIEMPO_EN_DIAS,
      T.NNOACTAPROCESAL NOACTAPROCESAL,
      T.FFUNCIONARIO FUNCIONARIO,
      T.CCREDENCIAL CREDENCIAL,
      T.RRANGO RANGO,
      T.CCARGO CARGO,
      T.DDDEPENDENCIA DEPENDENCIA,
      T.TTIPODEPENDENCIA TIPODEPENDENCIA
    FROM
     (SELECT
       DCASO.NOACTAPROCESAL NNOACTAPROCESAL,
       (CASE WHEN DCASO.FECHACONCLUSION IS NULL THEN SYSDATE -
DCASO.FECHA ELSE DCASO.FECHACONCLUSION - DCASO.FECHA END)
TTIEMPO_EN_DIAS,
        (DPERSONA.PNOMBRE||' '||DPERSONA.SNOMBRE||'
'||DPERSONA.PAPELLIDO||' '||DPERSONA.SAPELLIDO) FFUNCIONARIO,
        NRANGO.RANGO RRANGO,
        NCARGO.CARGO CCARGO,
        DFUNCIONARIO.CREDENCIAL CCREDENCIAL,
        DENTIDAD.NOMBRE DDDEPENDENCIA,
        NTDEPENDENCIA.DESCRIPCION TTIPODEPENDENCIA
      FROM DCASO LEFT JOIN DFUNCIONARIOCASO
```

```
          INNER JOIN DFUNCIONARIO
           LEFT JOIN NRANGO
           ON NRANGO.ID = DFUNCIONARIO.RANGO
           LEFT JOIN NCARGO
           ON NCARGO.ID = DFUNCIONARIO.CARGO
           LEFT JOIN DDEPENDENCIA
            INNER JOIN DENTIDAD
             INNER JOIN NTDEPENDENCIA
             ON NTDEPENDENCIA.ID = DENTIDAD.TIPO
            ON DENTIDAD.ID = DDEPENDENCIA.ENTIDAD
           ON DDEPENDENCIA.ENTIDAD = DFUNCIONARIO.DEPENDENCIA
           LEFT JOIN DPERSONA
           ON DPERSONA.ID = DFUNCIONARIO.PERSONA
          ON DFUNCIONARIO.PERSONA = DFUNCIONARIOCASO.FUNCIONARIO
         ON DFUNCIONARIOCASO.CASO = DCASO.ID
     WHERE DCASO.FECHA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
       AND UPPER(DENTIDAD.NOMBRE) = UPPER(PRM_DEPENDENCIA)
       AND DCASO.ACTIVO = 1
     ORDER BY TTIEMPO_EN_DIAS ASC) T
   WHERE ROWNUM <= PRM_CANT OR PRM_CANT IS NULL;
  IO_ALLREC := SINGLEREC;
END SP_FUNC_RAPIDO_CERRADO_CASOS;
```

Procedimiento almacenado SP_Buscar_Casos_Criteria

```
PROCEDURE SP_BUSCAR_CASOS_CRITERIA(IO_ALLREC OUT REF_CURSOR,
              CRITERIO IN VARCHAR2 DEFAULT NULL,
              FECHA_INICIO IN DCASO.FECHA%TYPE,
              FECHA_FIN IN DCASO.FECHA%TYPE)
IS
 SINGLEREC PKG_ESTADISTICAS.REF_CURSOR;
 PRM_CADENA VARCHAR2(32670):= CRITERIO;
 PRM_CRITERIA VARCHAR2(32670);
 PRM_FECHA_INICIO DCASO.FECHA%TYPE := FECHA_INICIO;
 PRM_FECHA_FIN DCASO.FECHA%TYPE := FECHA_FIN;
 SENTENCIA VARCHAR2(32670) := 'SELECT DISTINCT
               DCASO.NOACTAPROCESAL NNOACTAPROCESAL
               FROM DCASO LEFT JOIN NESTADOCASO
                 ON NESTADOCASO.ID = DCASO.ESTADO
                 LEFT JOIN DFISCAL
                  INNER JOIN DPERSONA FISCAL
                  ON FISCAL.ID = DFISCAL.PERSONA
                 ON DFISCAL.PERSONA = DCASO.FISCAL
                 LEFT JOIN DFISCALIA
                 ON DFISCALIA.ENTIDAD = DCASO.FISCALIA
                 LEFT JOIN DFUNCIONARIO FUNCIONARIOCREA
                  LEFT JOIN DDEPENDENCIA DEPENDENCIACREA
```

```
                    INNER JOIN DENTIDAD
                     INNER JOIN DDIRECCION
                      INNER JOIN NESTADO
                      ON NESTADO.ID = DDIRECCION.ESTADO
                      INNER JOIN NPARROQUIA
                      ON NPARROQUIA.ID = DDIRECCION.PARROQUIA
                      INNER JOIN NMUNICIPIO
                      ON NMUNICIPIO.ID=DDIRECCION.MUNICIPIO
                     ON DDIRECCION.ID = DENTIDAD.DIRECCION
                    ON DENTIDAD.ID = DEPENDENCIACREA.ENTIDAD
                   ON DEPENDENCIACREA.ENTIDAD =
FUNCIONARIOCREA.DEPENDENCIA
                  ON FUNCIONARIOCREA.PERSONA = DCASO.FUNCIONARIO
                  LEFT JOIN NCASOIMPORTANCIA
                  ON NCASOIMPORTANCIA.ID = DCASO.CASOIMPORTANCIA
                  LEFT JOIN DFUNCIONARIOCASO
                   LEFT JOIN DFUNCIONARIO
                    LEFT JOIN NRANGO
                    ON NRANGO.ID = DFUNCIONARIO.RANGO
                    LEFT JOIN NCARGO
                    ON NCARGO.ID = DFUNCIONARIO.CARGO
                    INNER JOIN DPERSONA FUNCIONARIOCASO
                    ON FUNCIONARIOCASO.ID = DFUNCIONARIO.PERSONA
                    LEFT JOIN DDEPENDENCIA
                     INNER JOIN DENTIDAD
                      INNER JOIN NTDEPENDENCIA
                      ON NTDEPENDENCIA.ID = DENTIDAD.TIPO
                     ON DENTIDAD.ID = DDEPENDENCIA.ENTIDAD
                    ON DDEPENDENCIA.ENTIDAD = DFUNCIONARIO.DEPENDENCIA
                   ON DFUNCIONARIO.PERSONA =
DFUNCIONARIOCASO.FUNCIONARIO
                  ON DFUNCIONARIOCASO.CASO = DCASO.ID
                  LEFT JOIN DDENUNCIA
                   INNER JOIN DFUNCIONARIO
                    INNER JOIN DPERSONA DENUNCIANTE
                    ON DENUNCIANTE.ID = DFUNCIONARIO.PERSONA
                   ON DFUNCIONARIO.PERSONA = DDENUNCIA.FUNCIONARIO
                   INNER JOIN DDELITO
                   INNER JOIN NTDELITO
                   ON NTDELITO.ID = DDELITO.TIPODELITO
                   INNER JOIN NGRADODELITO
                   ON NGRADODELITO.ID = DDELITO.GRADO
                   INNER JOIN NNATURALEZADELITO
                  ON NNATURALEZADELITO.ID = DDELITO.NATURALEZADELITO
                   ON DDELITO.ID = DDENUNCIA.DELITO
                  ON DDENUNCIA.ID = DCASO.DENUNCIA
    WHERE DCASO.ACTIVO = 1 AND DCASO.FECHA BETWEEN
'||''''||PRM_FECHA_INICIO||''''||' AND '||''''||PRM_FECHA_FIN||''''||' AND ';
BEGIN
```

```
   IF(PRM_CADENA IS NULL)THEN
    PRM_CRITERIA := '1=1';
   ELSIF(PRM_CADENA IS NOT NULL)THEN
    PRM_CRITERIA := FN_PARSEAR_CADENA_ESTADISTICAS(CRITERIO);
   END IF;
   SENTENCIA := SENTENCIA || PRM_CRITERIA;
   OPEN SINGLEREC FOR
     SENTENCIA;
   IO_ALLREC := SINGLEREC;

END SP_BUSCAR_CASOS_CRITERIA;
```

Procedimiento almacenado SP_Buscar_Solicitud

```
PROCEDURE SP_BUSCAR_SOLICITUD(IO_ALLREC OUT REF_CURSOR,
            NO_SOLICITUD IN DSOLICITUD.NOSOLICITUD%TYPE DEFAULT NULL,
            DEPENDENCIA IN DDEPENDENCIA.ENTIDAD%TYPE DEFAULT NULL,
            ESTADO_SOLICITUD IN NESTDILIGENCIA.ID%TYPE DEFAULT NULL,
            CRITERIO IN VARCHAR2 DEFAULT NULL)
 IS
 SINGLEREC PKG_CRIMINALISTICA.REF_CURSOR;
 PRM_NO_SOLICITUD DSOLICITUD.NOSOLICITUD%TYPE := NO_SOLICITUD;
 PRM_ESTADO_SOLICITUD NESTDILIGENCIA.ID%TYPE := ESTADO_SOLICITUD;
 PRM_DEPENDENCIA DDEPENDENCIA.ENTIDAD%TYPE := DEPENDENCIA;
 PRM_CRITERIO VARCHAR2(32767);
 SENTENCIA VARCHAR2(32767);
BEGIN
   IF(CRITERIO IS NOT NULL )THEN
   PRM_CRITERIO := FN_PARSEAR_CADENA_ESTADISTICAS(CRITERIO);
   ELSIF(CRITERIO IS NULL )THEN
   PRM_CRITERIO := '(1=1)';
   END IF;
   SENTENCIA:='SELECT DDILIGENCIA.FECHAHORA FECHA,
   DSOLICITUD.NOSOLICITUD NOSOLICITUD,
   DSOLICITUD.DESCRIPCION DESCRIPCION,
   NESTDILIGENCIA.DESCRIPCION ESTADO,
   NTSOLICITUD.DESCRIPCION TIPO,
   NTSUBSOLICITUD.DESCRIPCION SUBTIPO
FROM DSOLICITUD INNER JOIN DDILIGENCIA
       INNER JOIN NESTDILIGENCIA
       ON NESTDILIGENCIA.ID = DDILIGENCIA.ESTADO
       INNER JOIN DREVISIONDILIGENCIA
        INNER JOIN DFUNCIONARIO
         INNER JOIN DDEPENDENCIA
         ON DDEPENDENCIA.ENTIDAD = DFUNCIONARIO.DEPENDENCIA
        ON DFUNCIONARIO.PERSONA = DREVISIONDILIGENCIA.FUNCIONARIO
       ON DREVISIONDILIGENCIA.DILIGENCIA = DDILIGENCIA.ID
      ON DDILIGENCIA.ID = DSOLICITUD.DILIGENCIA
```

```
        INNER JOIN NTSUBSOLICITUD
         INNER JOIN NTSOLICITUD
         ON NTSOLICITUD.ID = NTSUBSOLICITUD.TIPO
        ON NTSUBSOLICITUD.ID = DSOLICITUD.TIPO
         WHERE DDILIGENCIA.ACTIVO= 1
           AND DDEPENDENCIA.ENTIDAD = '||""||PRM_DEPENDENCIA||""||' OR
'||""||PRM_DEPENDENCIA||""||' IS NULL
           AND (UPPER(DSOLICITUD.NOSOLICITUD) LIKE
UPPER('||""||'%'||PRM_NO_SOLICITUD||'%'||""||') OR '||""||PRM_NO_SOLICITUD
||""||' IS NULL)
           AND (UPPER(NESTDILIGENCIA.ID) =
UPPER('||""||PRM_ESTADO_SOLICITUD||""||') OR
'||""||PRM_ESTADO_SOLICITUD||""||' IS NULL)';

    OPEN SINGLEREC FOR
    SENTENCIA||'AND '||PRM_CRITERIO;
    IO_ALLREC:=SINGLEREC;
END SP_BUSCAR_SOLICITUD;
```

Procedimiento almacenado SP_Informes_Por_Tipo

```
PROCEDURE SP_INFORMES_POR_TIPO(IO_ALLREC OUT REF_CURSOR,
              FECHA_INICIO IN DDILIGENCIA.FECHAHORA%TYPE,
              FECHA_FIN IN DDILIGENCIA.FECHAHORA%TYPE,
              CANT_MOSTRAR IN NUMERIC DEFAULT NULL)
 IS
  SINGLEREC PKG_CRIMINALISTICA.REF_CURSOR;
  PRM_FECHA_INICIO DDILIGENCIA.FECHAHORA%TYPE := FECHA_INICIO;
  PRM_FECHA_FIN DDILIGENCIA.FECHAHORA%TYPE := FECHA_FIN;
  PRM_CANT_MOSTRAR NUMERIC(15) := CANT_MOSTRAR;
 BEGIN
    OPEN SINGLEREC FOR
     SELECT T.CCANTIDAD CANTIDAD,
       T.TTIPO TIPO,
       T.SSUBTIPO SUBTIPO
     FROM
     (SELECT COUNT(DINFORME.DILIGENCIA) CCANTIDAD,
       NTINFORME.DESCRIPCION TTIPO,
       NTSUBINFORME.DESCRIPCION SSUBTIPO
     FROM DINFORME INNER JOIN NTSUBINFORME
           INNER JOIN NTINFORME
           ON NTINFORME.ID = NTSUBINFORME.TIPO
          ON NTSUBINFORME.ID = DINFORME.SUBTIPO
          INNER JOIN DDILIGENCIA
          ON DDILIGENCIA.ID = DINFORME.DILIGENCIA
     WHERE DDILIGENCIA.ACTIVO = 1
       AND DDILIGENCIA.FECHAHORA BETWEEN PRM_FECHA_INICIO AND
PRM_FECHA_FIN
```

```
    GROUP BY NTSUBINFORME.DESCRIPCION , NTINFORME.DESCRIPCION
    ORDER BY COUNT(DINFORME.DILIGENCIA) ASC)T
    WHERE ROWNUM <= PRM_CANT_MOSTRAR OR PRM_CANT_MOSTRAR IS
NULL ;
  IO_ALLREC := SINGLEREC;
END SP_INFORMES_POR_TIPO;
```

Procedimiento almacenado SP_Buscar_Persona_Identificada

```
PROCEDURE SP_BUSCAR_PERSONA_IDENTIFICADA(IO_ALLREC OUT
REF_CURSOR,
                PRNOMBRE IN DPERSONA.PNOMBRE%TYPE DEFAULT NULL,
                SGNOMBRE IN DPERSONA.SNOMBRE%TYPE DEFAULT NULL,
                PRAPELLIDO IN DPERSONA.PAPELLIDO%TYPE DEFAULT NULL,
                SGAPELLIDO IN DPERSONA.SAPELLIDO%TYPE DEFAULT NULL,
                LCEDULA IN DPERSONA.LETRACED%TYPE DEFAULT NULL,
                NOCEDULA IN DPERSONA.CEDULA%TYPE DEFAULT NULL,
                NOPASAP IN DPERSONA.NOPASAPORTE%TYPE DEFAULT NULL)
IS
 SINGLEREC PKG_CRIMINALISTICA.REF_CURSOR;
 prmPRNOMBRE DPERSONA.PNOMBRE%TYPE:= PRNOMBRE;
 prmSGNOMBRE DPERSONA.SNOMBRE%TYPE:= SGNOMBRE;
 prmPRAPELLIDO DPERSONA.PAPELLIDO%TYPE:= PRAPELLIDO;
 prmSGAPELLIDO DPERSONA.SAPELLIDO%TYPE:= SGAPELLIDO;
 prmLCEDULA DPERSONA.LETRACED%TYPE:= LCEDULA;
 prmNOCEDULA DPERSONA.CEDULA%TYPE:= NOCEDULA;
 prmNOPASAP DPERSONA.NOPASAPORTE%TYPE:= NOPASAP;
BEGIN
 OPEN SINGLEREC FOR
   SELECT DPERSONA.ID ID,
     DPERSONA.PNOMBRE||' '||DPERSONA.SNOMBRE||'
'||DPERSONA.PAPELLIDO||' '||DPERSONA.SAPELLIDO NOMBRECOMPLETO,
     DPERSONA.LETRACED||DPERSONA.CEDULA CEDULA,
     DPERSONA.NOPASAPORTE NOPASAPORTE,
     DPERSONA.SEXO SEXO,
     NESTADOPERSONA.DESCRIPCION ESTADO,
     NPAIS.NOMBRE NATURALDE,
     (CASE WHEN DPERSONA.VALIDADO = 1 THEN 'SI' ELSE 'NO' END)
VALIDADOSAIME,
     (CASE WHEN DPERSONA.NOINDOCUMENTADA IS NOT NULL THEN 'SI'
ELSE 'NO' END) INDOCUMENTADA
   FROM DPERSONA
    INNER JOIN NESTADOPERSONA
    ON DPERSONA.ESTADO = NESTADOPERSONA.ID
    LEFT JOIN DORIGEN
      INNER JOIN NPAIS
      ON DORIGEN.PAIS = NPAIS.ID
    ON DPERSONA.ORIGEN = DORIGEN.ID
```

```
   WHERE (UPPER(DPERSONA.PNOMBRE) = UPPER(prmPRNOMBRE) OR
prmPRNOMBRE IS NULL)
      AND (UPPER(DPERSONA.SNOMBRE) = UPPER(prmSGNOMBRE) OR
prmSGNOMBRE IS NULL)
      AND (UPPER(DPERSONA.PAPELLIDO) = UPPER(prmPRAPELLIDO) OR
prmPRAPELLIDO IS NULL)
      AND (UPPER(DPERSONA.SAPELLIDO) = UPPER(prmSGAPELLIDO) OR
prmSGAPELLIDO IS NULL)
      AND (UPPER(DPERSONA.LETRACED) = UPPER(prmLCEDULA) OR
prmLCEDULA IS NULL)
      AND (DPERSONA.CEDULA = prmNOCEDULA OR prmNOCEDULA IS NULL)
      AND (DPERSONA.NOPASAPORTE = prmNOPASAP OR prmNOPASAP IS
NULL)
      AND DPERSONA.ACTIVO = 1;
IO_ALLREC := SINGLEREC;
END SP_BUSCAR_PERSONA_IDENTIFICADA;
```