Universidad de las Ciencias Informáticas
FACULTAD 2

Título: **"**Solución de Base de Datos para el Sistema de Gestión de Información de las Coordinaciones Regionales.**"**

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS.

Autor(es): Elba Camps David.
Abel Domínguez Marrero.

Tutores: Ing.Manuel de Jesús Abascal Matos.
Ing.Andrés Ballester Marsal.
Co-tutor: Ing.Dany Cuellar Rodríguez.

"**Año 53 de la Revolución"**
Ciudad de la Habana, Cuba.
Junio, 2011.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Elba Camps David

_____
Firma del Autor

Abel Domínguez Marrero

_____
Firma del Autor

Ing. Manuel de Jesús Abascal Matos

_____
Firma del Tutor

Ing. Andrés Ballester Marsal

_____
Firma del Tutor

RESUMEN

La Dirección General de Prevención del Delito de la República Bolivariana de Venezuela maneja un conjunto de información que se desea tener a la disposición de todos los miembros de la Dirección y los ciudadanos venezolanos, por lo que se necesita una herramienta informática que provea toda esta información de forma centralizada. Teniendo en cuenta dicha necesidad y con el objetivo de resolver la problemática de cómo garantizar la recopilación, el procesamiento y almacenamiento de datos en las Coordinaciones Regionales, el presente trabajo muestra el desarrollo de la Base de Datos que sustentará al Sistema de Gestión de Información de las Coordinaciones Regionales.

En este trabajo se hace referencia a conceptos importantes dentro del mundo de las bases de datos, temas relacionados con el SGBD a utilizar y la herramienta empleada para el diseño. Se analizarán y describirán los modelos de base de datos existentes, así como los patrones de diseño de base de datos propuestos para los diferentes módulos del sistema, además de definir algunos elementos que complementan la lógica de almacenamiento en el servidor de base de datos. Se explican además los principales objetos implementados que permitirán manipular la información dentro del Sistema Gestor de Base de Datos. Se brindan las vías para lograr una pronta recuperación y un funcionamiento óptimo del servidor en situaciones de fallas, así como las políticas de salvas y medidas de seguridad definidas para garantizar la integridad de los datos.

Con la realización de este trabajo se dará paso a lograr un diseño de base de datos que soporte las necesidades de almacenamiento, manipulación, persistencia y disponibilidad de la información a gestionar por el sistema.

INDICE

INTRODUCCION

Venezuela es uno de los países que se encuentra hoy en día inmerso en un gran proceso de reestructuración de la sociedad, donde se comienza a atender en primer lugar, el orden social, generando condiciones para la inclusión, al impulsar fuertemente la participación comunitaria en todos los niveles de las políticas públicas. La educación, la salud, la vivienda, el empleo, son elementos que contribuyen positivamente a las acciones emprendidas por el Estado para mejorar la seguridad ciudadana y la prevención del delito.

Partiendo de estas premisas los ciudadanos venezolanos cuentan con el Ministerio del Poder Popular para Relaciones Interiores y Justicia (MPPRIJ), que atendiendo a su misión institucional de garantizar la seguridad ciudadana contra hechos delictivos, promueve la formulación y puesta en marcha de proyectos orientados a perfeccionar el sistema de prevención del delito de la República Bolivariana de Venezuela, partiendo de que la seguridad ciudadana es una condición necesaria del desarrollo humano que propicia el mejoramiento de la calidad de vida de todos los ciudadanos que habitan en el territorio venezolano.

La DGPD se encuentra adscrita al Vice-ministerio de Seguridad Ciudadana del Ministerio del Poder Popular para Relaciones Interiores y Justicia y fue creada por decreto Nº 241, el 11 de febrero de 1970, con el objetivo de orientar hacia la creación, desarrollo y ejecución de políticas y programas enmarcados a la prevención de la violencia y la criminalidad a través del ejercicio de la corresponsabilidad entre instituciones, comunidades y el fortalecimiento de la convivencia ciudadana. La idea es facilitar las herramientas para que las comunidades puedan prevenir delitos, detectar problemas y elaborar planes, tanto en la materia preventiva como de otros tipos mediante talleres sobre Prevención Integral de la Violencia y Redes Comunitarias de Seguridad Ciudadana, charlas sobre prevención del consumo, tráfico de drogas y el rol de la familia en la prevención.

Para lograr estos objetivos la DGPD contacta a líderes comunitarios y luchadores sociales a través de las misiones, las organizaciones culturales y deportivas que operan en las comunidades y se lleva a cabo un intercambio de ideas sobre las diferentes esferas. A partir de esta información se planifican las actividades que requiere la comunidad y son orientadas a cada Coordinación Regional (CR) por la DGPD, las cuales son adaptadas y realizadas según la realidad de cada región. Las CRs deben rendir cuenta de su ejecución generando un informe mensual, trimestral y anual que contiene lo realizado en el período,

además manejan recursos materiales que le son asignados para el cumplimiento de sus actividades, así como un expediente de cada Consejo Comunal (CC) que tienen bajo su custodia, informes donde recogen las inquietudes, necesidades, problemas de la población y otras informaciones demostrando sus logros y desempeños.

La DGPD no cuenta con herramientas para la recopilación de estas informaciones. Estos procesos se realizan hoy de manera manual, mediante una agenda de trabajo del Coordinador Regional u otras formas de gestión de la información, dígase planillas o instrumentos creados como iniciativa de cada CR, en algunas ocasiones los informes son generados en formato Word o Excel. Todos estos factores pueden provocar demora en la generación de informes, generar gran volumen de información, en ocasiones el deterioro, pérdida o duplicidad de la misma, gasto innecesario de papel y otros recursos materiales, dificultando la ejecución de los procesos internos de la organización y la efectividad en el logro de resultados dirigidos al desarrollo de acciones que contribuyan a la prevención del delito. Debido a estos inconvenientes y en el marco del convenio Cuba-Venezuela los directivos de la DGPD y Albet[1] contratan la realización de una herramienta informática con el propósito de brindar a las CRs un sistema de gestión de información que permita un mejor trabajo en la recopilación de los datos que se requieren controlar y de esta manera brindar una asistencia de mayor calidad al ciudadano. Por ello se necesita de una solución de Base de Datos que facilite los procesos de almacenar y manipular la información que se maneja en dicho Sistema de Gestión de Información de las Coordinaciones Regionales de manera rápida y que garantice la disponibilidad de los datos.

Como resultado de lo analizado anteriormente surge el siguiente **Problema Científico** a resolver: ¿Cómo almacenar y manipular los datos del Sistema de Gestión de Información de las Coordinaciones Regionales?

A partir del problema científico se puede definir como **Objeto de Estudio** para la investigación: Bases de Datos para Sistemas Informáticos de Gestión de Información, teniendo como **Campo de Acción**:

[1] **Albet:** Albet Ingeniería y Sistemas, S.A. es una empresa cubana, cuyo origen y desarrollo se vincula estrechamente a la Universidad de Ciencias Informáticas (UCI).

Base de Datos Relacionales para el Sistema de Gestión de Información de las Coordinaciones Regionales.

Para dar solución al problema planteado anteriormente se propone como **Objetivo General**: Desarrollar una solución Base de Datos que permita almacenar y manipular la información del Sistema de Gestión de Información de las Coordinaciones Regionales.

Una vez definido el objetivo general y en aras de darle cumplimiento, se plantean los siguientes **Objetivos Específicos**:

- ✓ Diseñar la Base de Datos del Sistema de Gestión de Información de las Coordinaciones Regionales.
- ✓ Definir políticas de salvas (backup) y recuperación (recovery) de la Base de Datos.
- ✓ Creación de las estructuras físicas y lógicas de la base de datos.
- ✓ Implementar los diferentes objetos de la base de datos que permitan manipular la información dentro del Sistema Gestor de Base de Datos.

Para desarrollar satisfactoriamente la investigación se han trazado las siguientes **Tareas Investigativas**:

- ✓ Estudio de los procesos del negocio que serán automatizados por el Sistema de Gestión de Información de las Coordinaciones Regionales.
- ✓ Estudio de la metodología para el diseño de base de datos.
- ✓ Investigación del estado del arte del Sistema Gestor de Base de Datos a utilizar.
- ✓ Diseño de la Base de Datos del Sistema de Gestión de Información de las Coordinaciones Regionales.
- ✓ Investigación acerca de las estructuras de almacenamiento y privilegios de usuarios en el servidor de base de datos.
- ✓ Definición de la distribución de almacenamiento y los usuarios en el servidor de base de datos.
- ✓ Implementación de funciones, vistas, triggers y secuencias que permitan manipular la información dentro del Sistema Gestor de Base de Datos.
- ✓ Estudio de los métodos de salvas y recuperación de Base de Datos y selección del más adecuado.
- ✓ Definición de las políticas de salvas (backup) y recuperación (recovery) de la Base de Datos.

El presente trabajo de diploma está estructurado en cuatro capítulos:

Capítulo 1: "Fundamentación Teórica", en este capítulo se explican los elementos teóricos básicos y necesarios que soportan la investigación que se llevará a cabo, describiendo las principales características de las bases de datos, del sistema gestor de base de datos a utilizar y de la herramienta empleada para el diseño.

Capítulo 2: "Diseño de la Base de Datos", en este capítulo se describe cómo se desarrolla el diseño de la base de datos que da solución a cada uno de los módulos del sistema, destacando los elementos más relevantes del diseño, para así garantizar un correcto procesamiento de los datos y generación de informes.

Capítulo 3: "Complemento del Diseño de la Base de Datos", en este capítulo se incluyen los elementos necesarios que permiten complementar el diseño de la base de datos. Se definen algunos elementos de configuración como la distribución lógica y física de almacenamiento, los usuarios, las reglas básicas de backup y recovery definidas para la base de datos.

Capítulo 4: "Implementación de la Base de Datos", en este capítulo se incluyen los elementos necesarios que permiten realizar la implementación de los objetos de la base de datos que permitirán manipular la información dentro del Sistema Gestor de Base de Datos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se exponen los fundamentos teóricos que sustentan la investigación, definiendo los principales conceptos que están estrechamente relacionados con la solución de la Base de Datos (BD). Se describe además la metodología para efectuar el diseño de la misma, las características más importantes del Sistema Gestor de Base de Datos (SGBD) utilizado para manipular los datos y de la herramienta empleada para el diseño.

1.2. Base de Datos

Las primeras bases de datos manejaban ficheros que eran almacenados en tarjetas o soportes magnéticos. Cuando los ordenadores evolucionan, aparecen las cintas y los discos, a la vez las máquinas son dotadas de mucha más potencia y facilidad de manipulación, es por tanto en ese momento cuando las bases de datos comienzan a ser realmente útiles. (1)

Una Base de Datos es un conjunto de datos persistentes relacionados de forma lógica entre sí, que ha sido diseñada para satisfacer los requerimientos de información de una organización, almacenando en ella su descripción. En las BD se almacenan grandes cantidades de datos, que son definidos una sola vez y que pueden ser accedidos por varios usuarios al mismo tiempo, teniendo todos los datos integrados, con una mínima duplicidad y redundancia.

1.2.1. Modelos de Bases de Datos

Las BD además de clasificarse por su función, también se pueden clasificar de acuerdo a su modelo de administración de datos. Un modelo de datos es básicamente una descripción de lo conocido como contenedor de datos, así como los métodos para almacenar y recuperar la información de esos contenedores. Los modelos de datos son abstracciones que permiten la implementación de un sistema eficiente de base de datos.

1.2.1.1. Bases de Datos Jerárquicas

A principios de los años 60, coincidiendo en el tiempo con el desarrollo de los sistemas gestores de archivos IBM y North American Aviation se desarrolla el modelo jerárquico convirtiéndose en uno de los primeros modelos de los SGBD en aparecer.

Con este modelo los datos se organizan en una forma similar a un árbol visto al revés, en la que el nivel superior está ocupado por una única entidad, bajo la cual se distribuyen el resto de las entidades en niveles que se van ramificando y los diferentes niveles quedan unidos por medio de las relaciones. La representación gráfica de un modelo jerárquico está dada por un nodo padre de información que puede tener varios hijos, pero cada hijo sólo tiene un padre.

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. (2)

1.2.1.2. Bases de Datos de Red

El surgimiento de este modelo a mediados de 1960 dirigido por uno de los pioneros en los sistemas de bases de datos Charles Bachman fue una evolución del modelo jerárquico. Una BD de red se encuentra conformada por una colección de registros, los cuales están conectados entre sí por medio de enlaces en una red y abarcan más que la estructura de árbol, porque un nodo hijo en la estructura red puede tener más de un nodo padre, permitiendo así la conexión de los nodos en forma multidireccional.

Las bases de datos de red están concebidas como un modo flexible de representar objetos y su relación, además ofrecen una solución eficiente al problema de redundancia de datos, pero 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. (3)

1.2.1.3. Bases de Datos Relacionales

A pesar del éxito del modelo de datos en red, muchos diseñadores de software reconocieron que la interfaz de programación para navegación por los registros era de demasiado bajo nivel. En 1970 Edgar Frank Codd, en los laboratorios IBM en San José (California), basándose en el álgebra y la teoría de conjuntos, propone un nuevo modelo de datos llamado modelo relacional, que no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos, teniendo como idea fundamental el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas, esto es pensando en cada relación como si fuese una tabla que está compuesta por registros que representarían las tuplas y por campos.

La información en estas bases de datos puede ser recuperada o almacenada mediante consultas, ofreciendo una amplia flexibilidad y poder para administrar la información. El lenguaje más habitual para construir las consultas de las bases de datos relacionales es Structured Query Language (SQL) o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. (3)

En esencia este modelo es utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente donde el lugar y la forma en que se almacenen los datos no tienen relevancia, teniendo la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la BD.

1.2.1.4. Bases de Datos Orientadas a Objetos

Como respuesta a la creciente complejidad de las aplicaciones que requieren BD surgió el modelo de datos orientado a objetos, representando esta evolución la tercera generación de los sistemas de bases de datos.

Un objeto en una base dato orientada a objeto es una entidad identificable unívocamente que describe tanto el estado como el comportamiento de una entidad del mundo real. El estado de un objeto es descrito mediante atributos mientras que su comportamiento es definido mediante métodos. (4)

Una BD orientada a objetos incorpora los conceptos importantes del paradigma de objetos: (2)

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.

1.2.1.5. Bases de Datos Deductivas

Un sistema de base de datos deductivo permite hacer deducciones a través de inferencias y se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas bases de datos lógicas, a raíz de que se basan en lógica matemática. (2)

Una BD deductiva es en esencia un mapeo de relaciones base hacia hechos y reglas que son usadas para definir nuevas relaciones en términos de las relaciones base y el procesamiento de consultas, utilizando mecanismos internos para la evaluación y la optimización.

1.2.1.6. Bases de Datos Distribuidas

En este modelo la base de datos está almacenada en varias computadoras conectadas en red, las mismas surgen debido a la existencia física de organismos descentralizados, lo que le brinda la capacidad de unir las bases de datos de cada localidad y acceder así a distintos centros. (5)

Una BD distribuida es entonces una colección de datos que pertenecen lógicamente a un sólo sistema, pero se encuentra físicamente esparcido en varios sitios de la red.

1.2.2. Clasificaciones de las Bases de Datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, o la utilidad de la misma: (2)

Según la variabilidad de los datos almacenados:

1.2.2.1. 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. (2)

1.2.2.2. Bases de Datos Dinámicas

Son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado 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. (2)

Según el contenido:

1.2.2.3. 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 y edición de una determinada publicación. Puede contener un resumen o extracto de la publicación original,

pero nunca el texto completo. Como su nombre lo indica, el contenido son cifras o números, por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

1.2.2.4. Bases de Datos de Texto Completo

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas. (2)

1.2.2.5. Directorios

Un ejemplo son las guías telefónicas en formato electrónico. (2)

1.2.2.6. Bases de Datos o "Bibliotecas" de Información Biológica

Son bases de datos que almacenan diferentes tipos de información proveniente de las ciencias de la vida o médicas.

1.2.3. Metodología para el Diseño de Bases de Datos

Una BD correctamente diseñada permite obtener acceso a información exacta y actualizada, siendo este elemento esencial para lograr que la BD termine adaptándose a sus necesidades y pueda modificarse fácilmente.

Cualidades de un Buen Diseño de Base de Datos: (6)

- ✓ Reflejar la estructura del problema en el mundo real.
- ✓ Ser capaz de representar todos los datos esperados, incluso con el paso del tiempo.
- ✓ Evitar el almacenamiento de información redundante.
- ✓ Proporcionar un acceso eficaz a los datos.
- ✓ Mantener la integridad de los datos a lo largo del tiempo.
- ✓ Ser claro, coherente y de fácil comprensión.

Diseñar una BD es un proceso complejo que abarca decisiones en distintos niveles, la complejidad se controla mejor si se descompone el problema en partes y se resuelve cada uno de manera independientemente, utilizando técnicas específicas. El diseño de una BD se descompone en diseño conceptual, diseño lógico y diseño físico.

1.2.3.1. Diseño Conceptual

El diseño conceptual parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la BD. El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que esta maneja, la notación más popular utilizada en el esquema es el modelo entidad-relación.

Al construir el esquema, los diseñadores descubren la semántica de los datos, encontrando entidades, atributos, relaciones y su objetivo es comprender: (7)

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

El diseño conceptual es una descripción de alto nivel de la estructura de la BD y es completamente independiente de los aspectos de implementación, como puede ser el SGBD que se vaya a utilizar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física.

El esquema conceptual es una fuente de información para el diseño lógico de la base de datos. (7)

1.2.3.2. Diseño Lógico

El diseño lógico parte del esquema conceptual y ofrece como resultado un esquema lógico. El esquema lógico es la representación de la estructura de la BD en función de las estructuras de datos que puede procesar un tipo de SGBD, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos, por lo que el diseño lógico depende del tipo de SGBD que se vaya a utilizar y no del producto concreto. A medida que se va desarrollando el esquema lógico, este se va probando y validando con los requisitos de usuario. La normalización es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, porque asegura que las relaciones obtenidas no tienen datos redundantes.

El esquema lógico es una fuente de información para el diseño físico. (7)

1.2.3.3. Diseño Físico

El diseño físico parte del esquema lógico y ofrece como resultado un esquema físico. Un esquema físico es una descripción de la implementación de una BD en memoria secundaria: las estructuras de

almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos.

En esencia, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Por ejemplo, en el modelo relacional, esto consiste en: (7)

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

Para realizar el diseño de una BD se deben de tener en cuenta algunos elementos fundamentales y necesarios que a continuación se plantean:

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener la base de datos de un sistema de información determinado. En el caso relacional, esta estructura será un conjunto de esquemas de relación con sus atributos, dominios de atributos, claves primarias, claves foráneas. (8)

Las entidades son el fundamento del modelo entidad relación y se definen como objetos reales o abstractos relevantes que pueden ser identificadas unívocamente y sobre los cuales se recoge información, usualmente denotan una persona, lugar, cosa, concepto o evento de interés informacional.

Los atributos representan características interesantes o propiedades que definen o identifican a una entidad. Para cada atributo, existe un dominio del mismo, este hace referencia al tipo de datos que será almacenado o a restricciones en los valores que el atributo puede tomar.

Las claves o llaves son un subconjunto del conjunto de atributos comunes en una colección de entidades, que permite identificar unívocamente cada una de las entidades pertenecientes a dicha colección, además permiten distinguir entre sí las relaciones de un conjunto de relaciones.

Dentro de los conjuntos de entidades existen los siguientes tipos de claves:

- ✓ **Superclave:** Es un subconjunto de atributos que permite distinguir unívocamente cada una de las entidades de un conjunto de entidades. Si se añade un atributo al anterior subconjunto, el resultado seguirá siendo una superclave.

- ✓ **Clave candidata:** Conjunto de atributos no vacío, que identifica en forma única una entidad dentro de un conjunto de entidades.

- ✓ **Clave primaria:** Es la clave candidata escogida por el diseñador. Atributo o conjunto de atributos que permiten identificar en forma única una tupla en la tabla y ningún subconjunto de ella posee esta propiedad.

- ✓ **Clave foránea:** Es un atributo que existiendo como dependiente en una entidad, es a su vez llave en otra entidad con la cual se relaciona.

Las interrelaciones son asociaciones o conexiones que existen entre dos o más entidades. Las asociaciones son distinguidas por el número de posibles relaciones que una entidad determinada puede tener sobre esta, este rango es llamado cardinalidad y es especificado por la cantidad mínima y máxima de instancias de la asociación.

Dado un conjunto de relaciones binarias y los conjuntos de entidades A y B, la forma de transformar las interrelaciones binarias está muy influenciada por la cardinalidad mínima de los dos conjuntos de entidades asociados:

- ✓ **Uno a Uno:** Una entidad en A se asocia con a lo sumo una entidad en B y una entidad en B se asocia con a lo sumo una entidad en A. La llave primaria de uno de los conjuntos de entidades se incluye como llave foránea en el otro conjunto, aquel que sea más natural y si hay participación opcional debe analizarse donde se generen menos nulos.

- ✓ **Uno a Muchos:** Una entidad en A se asocia con cualquier número de entidades en B y una entidad en B, sin embargo, se puede asociar con a lo sumo una entidad en A. Se incluye la llave primaria del lado Uno en la tabla correspondiente al lado Muchos como llave foránea y si la interrelación tiene atributos se incluyen también en el lado Muchos.

- ✓ **Muchos a Uno:** Una entidad en A se asocia con a lo sumo una entidad en B y una entidad en B, sin embargo, se puede asociar con cualquier número de entidades en A.

- ✓ **Muchos a Muchos:** Una entidad en A se asocia con cualquier número de entidades en B y una entidad en B se asocia con cualquier número de entidades en A. Siempre se crea una nueva tabla que tendrá como llave primaria las llaves primarias de las tablas que corresponden a ambos lados. Esta nueva tabla incluirá como columnas los atributos de la interrelación si los tiene.

La participación de una entidad en una relación es obligatoria (total) si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante, sino, la participación es opcional (parcial).

Las relaciones entre entidades pueden clasificarse en:

- ✓ **Relación de identificación:** Relación mediante la cual se identifica una instancia de entidad secundaria a través de su asociación con una entidad principal, los atributos de clave primaria de la entidad principal se convierten en atributos de clave primaria de la entidad secundaria. En una relación de identificación, la entidad dependiente no puede existir sin la entidad principal. (9)

- ✓ **Relación de no identificación:** Relación mediante la cual no se identifica una instancia de entidad secundaria a través de su asociación con una entidad principal, los atributos de clave primaria de la entidad principal no se convierten en atributos de clave de la entidad secundaria. (9)

- ✓ **Relación de generalización/especialización:** Es el principio de la herencia donde se crea una tabla para la generalizada y una por cada especializada así las entidades de menor nivel heredan todos los atributos de las entidades de mayor nivel y la llave primaria en las especializadas es la llave primaria de la generalizada.

1.3. Patrones de Diseño de Bases de Datos

Desde el surgimiento de la humanidad los seres humanos han estado buscando en la naturaleza características que repetidamente se presentan entre los fenómenos naturales. De igual manera en el diseño de las BD y en el modelado de datos en general se presentan elementos repetitivos en disimiles modelos, los que correctamente identificados pasan a ser patrones de diseño. (10)

Un patrón es una plantilla que ya ha sido evaluada como la responsable de resolver un problema, es una guía para apoyarse en realizar un trabajo.

1.3.1. Árboles.

Un árbol es un término de la teoría de grafos y es un conjunto de nodos conectados en la estructura de hijo a padre. Un nodo puede tener muchos nodos hijos pero solo un padre, con excepción de un nodo raíz y no existen los ciclos por lo que un camino solo conecta a dos nodos. (10)

Los árboles son sumamente utilizados en los diseños actuales por lo que existen varios patrones de los mismos:

1.3.1.1. Árboles Fuertemente Codificados (Hardcoded tree)

En este patrón existe una entidad para cada nivel del árbol, normalmente son relaciones de uno a muchos. Usualmente es utilizado para representar jerarquías donde es bien conocida la estructura, es importante representar la correspondencia, por ejemplo las estructuras organizacionales. Es importante destacar que este patrón debe utilizarse sólo en los casos en que los cambios en la estructura a representar sean poco probables y que el patrón admite tantos niveles como requiera la jerarquía que se vaya a representar.

1.3.1.2. Árboles Simples

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

1.3.1.3. Árbol Estructurado

Este patrón es usado cuando se necesita diferenciar los nodos hojas (leaf), de aquellos que generan una nueva rama (branch), porque ambos tipos de nodos tienen diferentes atributos, relaciones y/o semántica. No pueden existir ciclos, un hijo no puede ser su propio padre. La generalización tiene cubrimiento total y exclusivo, cada elemento de una entidad Node, debe tener su correspondiente elemento en la entidad Leaf o en la entidad Branch.

1.3.2. Grafos

Los grafos dirigidos son otro elemento de la teoría de grafos. Un grafo dirigido es un conjunto de nodos y un conjunto de caminos dirigido entre los nodos. Desde un mismo nodo pueden salir un sinnúmero de caminos. El conjunto de modelos o patrones de grafos es similar al de los árboles, solo que las relaciones de uno a mucho son sustituidas por relaciones de muchos a muchos, surgiendo así una nueva entidad producto de la relación.

1.3.2.1. Grafo Dirigido Simple

Se utiliza cuando todos los nodos contienen el mismo tipo de datos. Es similar al árbol simple, la diferencia es que en este caso la relación recursiva sobre Node tiene cardinalidad de muchos a muchos. (10)

1.3.2.2. Grafo Dirigido Estructurado

Este modelo es usado cuando se necesita diferenciar los nodos hojas (leaf), de aquellos que generan una nueva rama (branch), porque ambos tipos de nodos tienen diferentes atributos, relaciones y/o semántica. Es similar al árbol estructurado, la diferencia es que en este caso la relación recursiva sobre Node tiene cardinalidad de muchos a muchos. (10)

1.3.3. Patrones para Flujos de Trabajo

Se encuentra la necesidad de representar y persistir flujos de trabajos por lo que existen dos patrones definidos para diseñar un flujo de trabajo.

1.3.3.1. Máquinas de Estado para un Tipo de Entidad

El patrón máquina de estado para un tipo de entidad representa los posibles cambios de estado por lo que puede atravesar un tipo de entidad. Este modelo es usado para representar cuales son los estados por los que puede pasar un tipo de entidad, no refleja el almacenamiento de las ocurrencias sino las propiedades del flujo de trabajo, por lo tanto este es el diseño del diagrama de flujo. (10)

1.3.3.2. Máquina de Estado para Escenarios (Control de Flujo)

Este modelo representa la ocurrencia del cambio de estado en un escenario de una entidad dada, por lo tanto considera el tiempo y la persistencia del mismo en las tablas resultantes. También representa la ocurrencia de un estímulo en una fecha y los estados por los que ha pasado, caracterizados por la fecha de inicio y la fecha fin. (10)

1.3.4. Modelo Entidad-Atributo-Valor

La mayoría de las aplicaciones de BD mapean directamente conceptos a tablas. El enfoque directo es eficaz para aplicaciones con tipos de entidades y atributos bien definidos, sin embargo, esto falla para aplicaciones cuyos requerimientos puedan variar a lo largo del desarrollo, sobre todo si estas variaciones implican la creación de nuevas entidades.

El modelo entidad-atributo-valor es la representación de un modelo flexible donde se pueden representar objetos con sus atributos, es un acercamiento al modelo orientado a objeto representado en el modelo relacional, donde la entidad Class representa las clases, la entidad Attribute representa los atributos de las clases, por su parte la entidad Object representa las instancias de las clases, mientras que la entidad Value representa los valores de cada atributo para cada objeto dado. (10)

1.3.5. Patrón de Llaves Subrogadas

Este patrón es muy utilizado pues se decide generar una llave primara única para cada entidad en vez de usar un atributo identificador en el contexto dado. Normalmente se usa enteros en columnas que no se repitan o con una probabilidad extremadamente baja. Permite además que las tablas sean más fáciles de consultar por el identificador dado.

Los patrones de diseño de base de datos se utilizan con el objetivo de permitir crear una BD más fortalecida ya que brindan una guía para especificar cómo deben ser las BD proporcionando una solución probada a un problema. Uno de los patrones a emplear son los árboles fuertemente codificados y se muestra reflejado en el diseño del módulo Administrar Mapa para representar jerarquías donde es bien conocida la estructura; las relaciones que se establecen entre las entidades que forman parte del mismo, nestado, nmunicipio y nparroquia es de uno a muchos y los cambios en dicha estructura son poco probables. Se utilizará también el patrón entidad-atributo-valor porque todas las entidades creadas se van a representar con sus atributos y los valores de los mismos lo que posibilitará que la aplicación resulte mucho más configurable a la hora de manipular la información. Además se empleará el patrón de llaves subrogadas porque cada entidad que se cree va a estar representada por una llave primaria única para facilitar que las mismas sean más fáciles de consultar por el identificador único que poseen. Estos dos últimos patrones serán utilizados en todo el diseño.

1.4. Normalización de las Bases de Datos

El concepto de normalización fue introducido por E.D.Codd y pensado para aplicarse a sistemas relacionales (11). La teoría de la normalización se ha desarrollado para obtener estructuras de datos eficientes y es el proceso durante el cual los esquemas de relación insatisfactorios se descomponen, repartiendo sus atributos entre esquemas de relación más pequeños que poseen propiedades deseables para crear relaciones lógicas apropiadas entre tablas de una BD.

Ventajas de la Normalización:

- ✓ Evitar anomalías en la actualización. (11)
- ✓ Mejorar la independencia de los datos, permitiendo realizar extensiones de la base de datos afectando muy poco, o nada, a los programas de aplicación existentes que acceden a la misma. (11)
- ✓ Mínima redundancia, la información no está duplicada innecesariamente. (12)
- ✓ Máximo rendimiento de las aplicaciones, sólo se trata aquella información que va a servir de utilidad a cada aplicación. (12)
- ✓ Menos oportunidades para generar incoherencias.
- ✓ Facilidad de uso y claridad, los datos están agrupados en tablas que identifican claramente una entidad o relación. Su representación es clara y sencilla de entender incluso para usuarios finales. (12)
- ✓ Flexibilidad y facilidad de gestión, la información que necesitan los usuarios se puede obtener de las tablas relacionales o relaciones mediante operaciones del álgebra y cálculo relacionales. (12)
- ✓ Precisión, las interrelaciones entre las tablas consiguen mantener información diferente relacionada con toda exactitud. (12)

Las formas normales son aplicables a tablas individuales, decir que una base de datos entera está en la forma normal n es decir que todas sus tablas están en la forma normal n. (13)

Las formas normales proveen el marco formal para analizar los esquemas de relación de la BD con sus claves, las dependencias funcionales entre sus atributos y una serie de pruebas que pueden efectuarse sobre los esquemas de relación individualmente. Además proporcionan los criterios para determinar el grado de vulnerabilidad de una tabla, mientras más alta sea la forma normal aplicable a una tabla, es

menos vulnerable a inconsistencias lógicas de varios tipos y anomalías. Hay que tener presente que normalizar demasiado puede conducir a tener una BD ineficiente y hacer al esquema demasiado complejo para trabajar porque a medida que progresa la normalización, también aumenta el número y la complejidad de las combinaciones necesarias para recuperar los datos y un número muy elevado de combinaciones relacionales complejas entre demasiadas tablas puede afectar al rendimiento.

El proceso de normalización involucra un conjunto de reglas para cada fase que se realizan en orden, cada regla está basada en la que le antecede.

1.4.1. Primera Forma Normal (1FN)

Una relación R está en 1FN si los valores que contienen los atributos son atómicos, no pueden existir atributos multivaluados, compuestos o derivados. Un dominio es atómico si se considera que los elementos del dominio son unidades indivisibles (12). Esta forma normal elimina los valores repetidos dentro de una BD.

1.4.2. Segunda Forma Normal (2FN)

Una relación R está en 2FN si está en 1FN y los atributos no llaves de R, son funcionales y completamente dependientes de la llave primaria. Una dependencia funcional es completamente funcional si al eliminar los atributos A de R la dependencia no es mantenida.

1.4.3. Tercera Forma Normal (3FN)

Una relación R está en 3FN si está en 2FN y cada atributo que no forma parte de ninguna clave, depende directamente y no transitivamente de la clave primaria (14) . Existe dependencia funcional transitiva entre dos atributos cuando un atributo que no pertenece a la clave primaria permite conocer el valor de otro atributo.

1.4.4. Forma Normal de Boyce-Codd (FNBC)

Una relación R está en FNBC si está en 3FN y para toda dependencia funcional válida en R, se cumple que X es superllave o clave, por tanto cada determinante, atributo que determina completamente a otro, es clave candidata.

1.4.5. Cuarta Forma Normal (4FN)

Una relación R está en 4NF si está en FNBC y no contiene dependencias multivaluadas no triviales. Una dependencia multivaluada A ->-> B es trivial si B es un subconjunto de A o si A U B = R. (15)

La 4FN se asegura de que las dependencias multivaluadas independientes estén correctas y eficientemente representadas en un diseño de base de datos. Una tabla con una dependencia multivaluada es una donde la existencia de dos o más relaciones independientes muchos a muchos causa redundancia y esta redundancia es suprimida por la cuarta forma normal. (16)

1.4.6. Quinta Forma Normal (5FN)

Una relación R está en 5FN si está en 4FN y si cada relación de dependencia se encuentra definida por las claves candidatas. La 5FN está designada para reducir redundancia en las BD relacionales que guardan hechos de valores multivaluados aislando semánticamente relaciones múltiples relacionadas.

1.4.7. Forma Normal de Dominio/Clave (DKNF)

Una relación R que está en DKNF es una forma normal que requiere que la base de datos contenga restricciones de dominios y de claves. Una restricción del dominio especifica los valores permitidos para un atributo dado, mientras que una restricción clave especifica los atributos que identifican únicamente una fila en una tabla dada. (17)

1.4.8. Sexta Forma Normal (6FN)

Una relación R que está en la 6FN se piensa para descomponer variables de la relación a los componentes irreducibles. Aunque esto puede ser relativamente poco importante para las variables no temporales de la relación, puede ser importante al ocuparse de las variables temporales o de otros datos del intervalo.

En general, las primeras tres formas normales proveen suficiente nivel de normalización para cubrir las necesidades de la mayoría de las bases de datos. Para el diseño de la base de datos del sistema se decidió normalizar hasta el nivel tres.

1.5. Sistema Gestor de Bases de Datos (SGBD)

Los Sistemas de Gestión de Bases de Datos son aplicaciones que facilitan a los usuarios los medios necesarios para el proceso de definir, construir, recuperar y manipular las BD, proporcionando un acceso controlado y eficiente a la misma, asegurando su confidencialidad, seguridad, independencia física y lógica de los datos. Se compone de un lenguaje de definición de datos (DDL), de un lenguaje de manipulación de datos (DML) y de un lenguaje de consulta.

Los sistemas jerárquico y de red constituyen la primera generación de los SGBD a partir de los años 1960; el primer SGBD comercial desarrollado basado en el modelo jerárquico fue el Information Management System (IMS) con la finalidad de resolver problemas de diseño aeroespacial y de producción, de igual manera se puede considerar a IDS (Integrated Data Store) de Bachman como el primer sistema de base de datos creado bajo el concepto del modelo de datos de red. Viendo la necesidad de mejorar la forma de gestionar la información a través de los SGBD se desarrollaron los Sistemas Gestores de Base de Datos Relacionales (SGBDR) basados en la tesis propuesta por el Dr.Edgar F. Codd en 1970 del Modelo de Datos Relacional el cual propone modelar el mundo real mediante relaciones que representan abstracciones de elementos de la realidad.

Entre los SGBD más comunes se encuentran: Oracle, Microsoft SQL Server y Borland Interbase que comercialmente son los más fuertes, sin embargo en el mundo del software libre, se aprecian opciones tan completas como: SQLite, Firebird, DB2 Express-C, MySQL y PostgreSQL.

El SGBD a utilizar para el desarrollo e implementación de la Base de Datos para el Sistema de Gestión de Información de las Coordinaciones Regionales es el PostgreSQL 8.4, debido a los beneficios que ofrece.

1.5.1. PostgreSQL

El origen de PostgreSQL se sitúa en el gestor de bases de datos POSTGRES desarrollado en la Universidad de Berkeley y que se abandonó en favor de PostgreSQL a partir de 1994 por Michael Stonebraker. Ya entonces, contaba con prestaciones que lo hacían único en el mercado y que otros gestores de bases de datos comerciales han ido añadiendo durante este tiempo. (18)

Éste sistema gestor de base de datos se encuentra bajo licencia BSD (Berkeley Software Distribution). Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre. (19)

DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia y arreglos. (20)

Altamente Extensible: PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario. (20)

Escalabilidad: Posee una gran escalabilidad, es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.

Herencia: Las tablas pueden ser configuradas para heredar características de una tabla padre. Los datos son compartidos entre las tablas padres e hijas. Las tuplas insertadas o eliminadas en la tabla hija serán insertadas o eliminadas en la tabla padre respectivamente. (20)

Multiplataforma: Puede funcionar en múltiples plataformas por lo que corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows.

Soporte SQL Comprensivo: PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92. (20)

Integridad Referencial: PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos. (20)

Cliente/Servidor: PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor para garantizar la estabilidad del sistema; un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

API Flexible: La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++ y Pike. (20)

Amplia Variedad de Tipos Nativos: Soporta distintos tipos de datos, además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, direcciones MAC, direcciones IP, cadenas de bits, texto de largo ilimitado, incorpora una estructura de datos array. Adicionalmente los usuarios pueden crear sus propios tipos de datos.

Lenguajes Procedurales: PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido. (20)

Control de Concurrencia Multi-Versión (MVCC): Es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios, permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos permitiendo una alta concurrencia. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases porque un lector nunca es bloqueado por un escritor. PostgreSQL es capaz de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

Write Ahead Logging (WAL): La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual se podrá restaurar la base de datos. (20)

Cumple completamente con las características atomicidad, consistencia, aislamiento y durabilidad (Atomicity, Consistency, Isolation and Durability: ACID) para realizar transacciones seguras: (20)

- ✓ **Atomicidad indivisible:** Es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

- ✓ **Consistencia:** Es la propiedad que asegura que sólo se empieza aquello que se puede acabar, por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.

- ✓ **Aislamiento:** Es la propiedad que asegura que una operación no puede afectar a otras, esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.

- ✓ **Durabilidad:** Es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

Documentación: Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.

Gran Comunidad de Usuarios y Desarrolladores: Comunidades muy activas, varias comunidades en castellano.

Soporte

- ✓ Soporta el uso de índices, reglas, disparadores y vistas.
- ✓ Claves ajenas también denominadas llaves ajenas o claves foráneas.
- ✓ Altamente adaptable a las necesidades del cliente.
- ✓ Consultas complejas.
- ✓ Permite trabajar con grandes volúmenes de datos.
- ✓ Almacenaje especial para tipos de datos grandes.
- ✓ Utilidades para limpieza de la BD.
- ✓ Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

Los Límites para el Almacenamiento en PostgreSQL son: (21)

- ✓ Máximo tamaño Base de Datos: Ilimitado.
- ✓ Máximo tamaño de Tabla: 32 TB.
- ✓ Máximo tamaño de Fila: 1.6 TB.
- ✓ Máximo de tamaño de Campo: 1 GB.
- ✓ Máximo número de filas por Tabla: Ilimitado.
- ✓ Máximo número de columnas por Tabla: 250 a 1600 (depende de los tipos usados).
- ✓ Máximo número de índices por Tabla: Ilimitado.

1.6. Herramienta para el Diseño de la Base de Datos

Las herramientas CASE (Ingeniería de Software Asistida por Computadora) tienen como principal ventaja la mejora de la calidad de los aspectos realizados durante todo el ciclo de vida de desarrollo del software, al facilitar ayuda en el desempeño de tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación

automática, documentación o detección de errores entre otras. Dentro de estas herramientas CASE se pueden destacar: Enterprise Architect, Visual Paradigm y Rational Rose.

La herramienta a utilizar en el diseño de la Base de Datos para el Sistema de Gestión de Información de las Coordinaciones Regionales es el Visual Paradigm for UML Enterprise Edition 6.4.

1.6.1. Visual Paradigm

Es una herramienta CASE que utiliza "UML": como lenguaje de modelado, soporta el ciclo de vida completo del desarrollo de software. Presenta dentro de sus características:

- ✓ Posee gran factibilidad de uso, es potente, fácil de instalar, utilizar, actualizar y es compatible entre ediciones.
- ✓ Permite diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación del proyecto con diferentes extensiones como PDF, .Doc.
- ✓ Ofrece soporte para el mapeo de objeto relacional (Object-Relational Mapping – ORM).
- ✓ Brinda facilidades para la generación de BD (transformación de diagramas de Entidad-Relación en tablas de BD).
- ✓ Posibilita ingeniería inversa de BD (desde SGBD existentes a diagramas de Entidad-Relación).
- ✓ Permite el uso de un lenguaje estándar común para todo el equipo de desarrollo.

1.7. Conclusiones

En este capítulo se hizo un estudio de los aspectos necesarios, relacionados con la Solución de Bases de Datos para el Sistema de Gestión de Información de las Coordinaciones Regionales, que servirán de ayuda, guía o punto de partida para el desarrollo de este trabajo. Se abordaron los conceptos y las características más importantes de las BD y del SGBD a utilizar. Para el diseño y la implementación de la BD se definieron algunos aspectos importantes como:

- ✓ El Gestor de Base de Datos a utilizar es: PostgreSQL.
- ✓ La herramienta que ayudará para el diseño es: el Visual Paradigm.

CAPÍTULO 2: DISEÑO DE LA BASE DE DATOS

2.1. Introducción

En este capítulo se describirá el diseño de BD propuesto como solución a las necesidades de almacenamiento de información en los diferentes módulos para el Sistema de Gestión de Información de las Coordinaciones Regionales. Para una mayor organización el capítulo se divide en fragmentos del negocio identificados para desarrollar el sistema, fragmentos que responden a un problema que surge a partir del actuar diario en las CRs.

En la figura 1 se muestran algunas combinaciones de relaciones entre entidades, pero es válido recordar que las relaciones de identificación y de no identificación pueden tener cualquier combinación de cardinalidad.

En el diseño se utilizó una notación para nombrar las tablas teniendo en cuenta el tipo de información que se almacena. Las tablas de nomencladores que son aquellas que guardan información predefinida en el sistema se identificarán con la letra “n” como prefijo del nombre. Las tablas de relaciones, las cuales se crean a partir de una relación entre dos o más tablas tendrán como prefijo una “r”. Las tablas de datos, las cuales guardan la información variable se identifican por su nombre. Los atributos que identifican las tablas tendrán id_ como prefijo del nombre de la entidad.

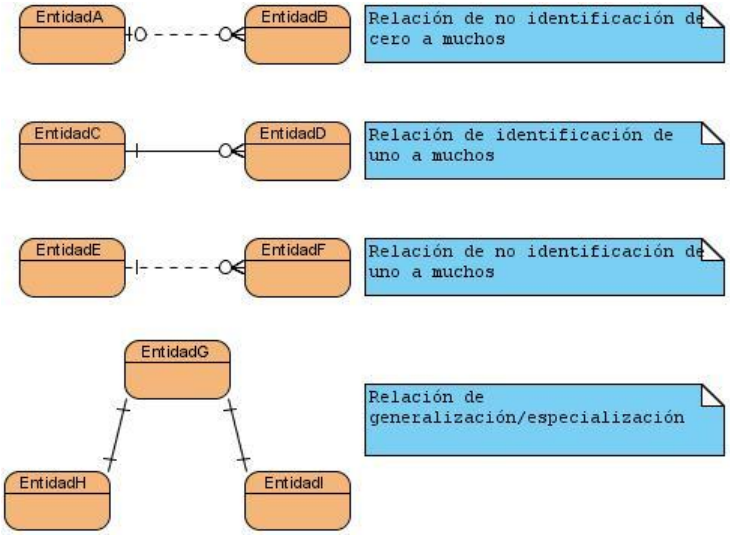

Figura 1. Relaciones entre Entidades.

2.2. Aspectos Generales

2.2.1. Nomencladores

Como parte del diseño uno de los elementos significativos fue la creación de los nomencladores que son tablas que se crean con el objetivo de almacenar los datos que son accedidos con mayor frecuencia por el sistema con el fin de alcanzar un mayor rendimiento, lo que contribuye a minimizar la probabilidad de que el usuario ingrese datos con errores y además lograr una mayor facilidad de trabajo para el usuario. Un nomenclador va a estar determinado por su nombre, descripción, si está activo y si es permanente, para poder trabajar con los nomencladores a nivel de programación tienen que estar activos y en el caso de permanente es que no puede ser eliminado el nomenclador.

2.2.2. Adjunto

Los adjuntos son documentos, fotos, videos, archivos de audio, etc. que su capacidad no excederá de 10mb y su objetivo para el sistema es servir de evidencia del trabajo en cada una de la CRs. Se adjuntará la información referente a los módulos Administrar Usuario, Recursos Humanos, Bienes Muebles, Recursos Materiales, Denuncia, Atención de Casos, Administrar Programas, Actividades, Consejos Comunales y Diagnóstico. Debido a esto se hizo necesario crear la entidad adjunto. De cada adjunto se almacenará el nombre, el adjunto, el tamaño, la fecha de registro y el tipo de contenido.

2.3. Módulo de Administrar Usuarios

2.3.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo responsable de la administración del sistema, que permitirá el acceso a la aplicación a los usuarios autorizados. La aplicación está basada en roles, los cuales representan un conjunto de privilegios sobre cada una de las funcionalidades del sistema según los permisos que tengan los usuarios con el objetivo de restringir las actividades que puedan realizar de acuerdo a su cargo.

El usuario es la persona que debe autenticarse para interactuar con el sistema con un nombre de usuario y una contraseña ,para almacenar estos datos se tiene la entidad usuario_sistema y una vez autenticado podrá realizar las acciones en dependencia del rol que tenga asignado como funcionario al insertarse en el sistema; los roles del sistema son: Secretario del CR, Coordinador Regional, Equipo Técnico Profesional (ETP), Administrador de la CR, Director de la Dirección de Programas y Coordinaciones Nacionales(DPCN), Supervisor, Administrador del Sistema y Administrador de Programas que dentro del

diseño van a estar representados por la entidad nrol. Para lograr las asignaciones de los roles a los diferentes usuarios se tiene en cuenta que un usuario puede tener más de un rol asignado y que un rol puede estar presente en más de un usuario, así surge la relación de muchos a muchos entre estas dos entidades dando lugar a la nueva tabla rusuario_sistema_rol. Un usuario del sistema si el rol que tiene asignado es de ETP puede tener asociados varios tipos de atención por lo que surge la necesidad de crear una relación entre la entidad usuario_sistema y el nomenclador ntipo_atencion, la misma se establece con una cardinalidad de muchos a muchos y genera una nueva entidad que guardará la relación establecida en la entidad retp_tipo_atencion. A su vez si el rol es de supervisora puede tener varios estados a los cuales atenderá, para darle solución a esta problemática se creó la relación entre usuario_sistema y el nomenclador nestado surgiendo la nueva entidad rsupervisora_estado.

Como se muestra en la figura 2, en el diseño se tiene la entidad funcionario que almacenará los datos generales del funcionario que desempeña funciones en el organismo del Estado representando al poder público y de los que se registra la fecha de ingreso al ministerio y a la coordinación, ubicación administrativa, profesión, función que ejecuta, cargo, correo, esta entidad hereda de la entidad funcionario_base que almacenará los datos más identificativos del funcionario. La relación de generalización/especialización entre estas entidades surge por la necesidad de crear un usuario para las CRs sin necesidad de que dicho funcionario esté registrado todavía por el módulo de Recursos Humanos en la BD, por tanto solo se registrarán sus datos principales como el nombre, apellidos, cédula y la CR a la cual pertenece, información que es necesaria a la hora de crear un usuario para el sistema. De esta forma surge la relación de funcionario_base con usuario_sistema donde se especifica que un funcionario tendrá un único usuario con el que podrá interactuar con el sistema.

Con este diseño se logra restringir el acceso a la aplicación a los usuarios autorizados y de acuerdo a los permisos que tengan realizar las acciones que le correspondan.

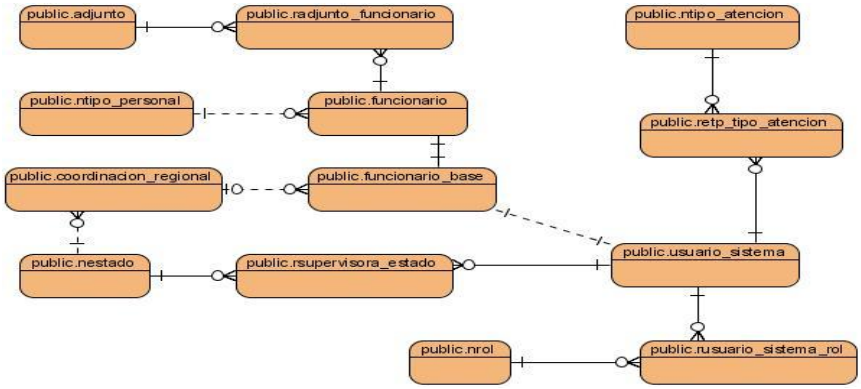

Figura 2. Módulo Administrar Usuario.

2.4. Módulo Recursos Humanos

2.4.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que permitirá gestionar la información de los Recursos Humanos de las CRs, donde se controlará el expediente de cada uno de los trabajadores con todos sus datos personales y de contacto.

En el diseño del módulo como se muestra en la figura 3 fue necesario crear la entidad periodo_vacacional que recogerá de cada funcionario el periodo vacacional con el que será estimulado y la entidad periodo_reposo donde se registrará la fecha de inicio y fin de ese periodo y el motivo de las ausencias. Se creó además la entidad control_asistencia donde se llevará el control con la hora de entrada y salida en la mañana y en la tarde del funcionario, en el caso de haber terminado su jornada laborar luego del horario establecido se le registrará como horas extras, si por algún motivo llegarán tarde al trabajo, la coordinadora le llenará unas observaciones y le justificará, pero en caso de ausencia injustificada se le registrará también.

Estas entidades se crean debido a que de un funcionario se necesita llevar su control de asistencia y a su vez un funcionario es el encargado de registrar dichos controles de asistencias al resto del personal de la CR. Para darle solución a la problemática se creó una relación de uno a muchos entre las entidades funcionario y control_asistencia que permitirá conocer los controles de asistencias personales de cada funcionario y otra relación de muchos a muchos entre funcionario_base y control_asistencia que posibilitará saber qué funcionario registró un control de asistencia. Además un funcionario es el encargado de llevar el registro del periodo vacacional y del periodo de reposo de los funcionarios de la CR, lo que da lugar a que surjan las relaciones entre funcionario con periodo_vacacional y periodo_reposo.

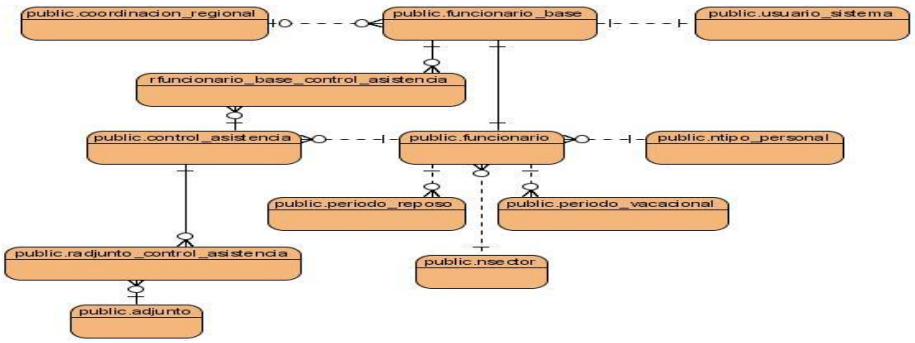

Figura 3. Módulo Recursos Humanos.

2.5. Módulo Bienes Muebles

2.5.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que permitirá gestionar los Bienes Muebles pertenecientes a las CRs, específicamente las incorporaciones y desincorporaciones. Los bienes muebles son aquellos que presentan una depreciación es decir los activos fijos tangibles de una CR y pueden estar dados por sillas, mesas, computadoras.

En el diseño de este módulo (véase figura 4) se hizo necesario crear una entidad bien mueble la cual se encargará de almacenar el número de inventario, nombre, localización, estado, además de una entidad operacion_bien_mueble en la cual se encontrarán los atributos referentes a las operaciones de tipo incorporación y desincorporación que se les puede realizar a un bien mueble por el funcionario encargado.

Debido a que a un bien mueble se le pueden realizar varios tipos de operaciones y una misma operación se le puede realizar a varios bienes muebles se genera una nueva entidad roperacion_bien_mueble pudiendo conocer qué tipo de operación se le hizo a un determinado bien mueble. Se necesita además conocer la CR a la cual pertenece un bien mueble de ahí la relación de bien_mueble con la entidad coordinación_regional.

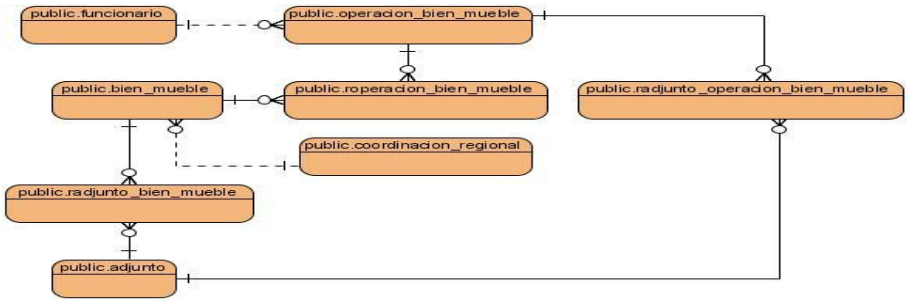

Figura 4. Módulo Bienes Muebles.

2.6. Módulo Recursos Materiales

2.6.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que permitirá gestionar la información asociada al control de los Recursos Materiales que se le entregan a las CRs y que pueden estar dados por pelotas, guantes, bates, entre otros, para la realización de las actividades en la comunidad. Se controlará un registro con los datos de los recursos, ingresos y retiros de los materiales.

Como se muestra en la figura 5, correspondiente al diseño de este módulo, se hizo necesario crear la entidad material la cual se encargará de almacenar el nombre del material, la cantidad en depósito, además de una entidad operación en la cual se encontrarán los atributos referentes a las operaciones de tipo inserción y retiro que se le puede realizar a un material por el funcionario encargado.

Debido a que a un bien mueble se le pueden realizar varios tipos de operaciones y una misma operación se le puede realizar a varios materiales se genera una nueva entidad roperacion_material pudiendo conocer qué tipo de operación se le realizó a un determinado material. Se necesita además conocer la CR a la cual pertenece el material de ahí que surja la relación de la entidad material con la entidad coordinación_regional.

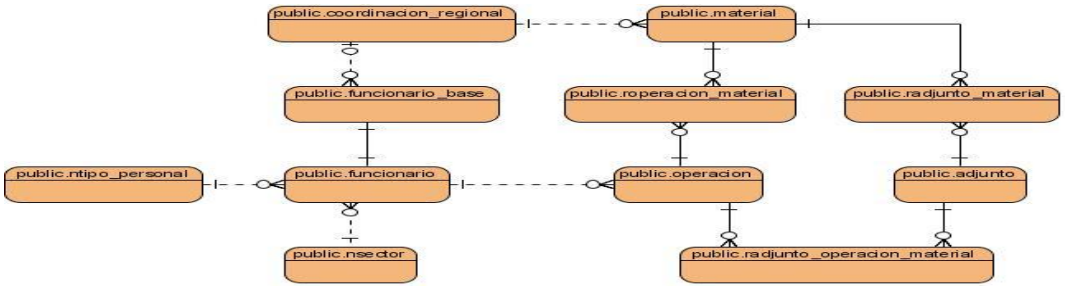

Figura 5. Módulo Recursos Materiales.

2.7. Módulo Denuncia

2.7.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que permitirá gestionar las denuncias recogidas por las CRs y las remisiones emitidas a las mismas. La denuncia consiste en la declaración de la ocurrencia de un hecho delictivo por parte de una persona que ha sido afectada directa o indirectamente por otra persona, pero las CRs no son las

encargadas de resolver las denuncias son solo enlaces entre la población y las instituciones responsables de darles solución.

En el diseño de este módulo (véase figura 6) se tiene como elemento significativo la denuncia y se hizo necesario crearla para almacenar toda la información relacionada con el hecho delictivo la hora, la fecha, lugar del hecho denunciado, tipo de denuncia: confidencial o abierta, coordinación regional a la que pertenece, además el origen de la denuncia, el tipo de clasificación del delito, el sector que integra y el tipo de actividad asociada que se obtienen a través de las relaciones con las entidades norigen_denuncia, nclasificacion_delito, nsector y ntipo_actividad respectivamente, de tipo nomenclador que estarán previamente definidas por el sistema.

Un tipo de actividad puede ser uno de los orígenes de la denuncia, esta puede ser seleccionada o no por el usuario, por esta razón la relación que existe entre estas entidades es opcional. Cuando se formula una denuncia, se pueden recoger o no los datos personales del entrevistado que es el que realiza la declaración, debido a esto la relación entre denuncia y ciudadano es opcional también, lo mismo puede ocurrir entre denuncia y los datos correspondientes a la clasificación del delito y el sector. En el caso de registrar los datos de la persona denunciante, se almacenarán en la entidad ciudadano.

Una vez realizada la denuncia el usuario del sistema encargado puede remitir al denunciante a las instituciones u organizaciones que le darán solución a la denuncia, por lo cual fue necesario crear la entidad remisión que almacenará la fecha, descripción, estado de la denuncia, el estatus de la remisión: registrada, remitida o procesada y organización a la cual va remitida de ahí que se establezca la relación entre la entidad remision y la entidad norganismo_institucion.

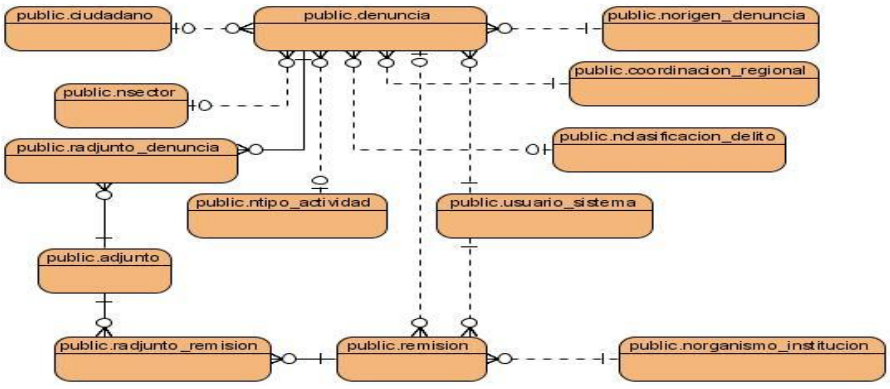

Figura 6. Módulo Denuncia.

2.8. Módulo Atención de Casos

2.8.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que permitirá gestionar los casos atendidos por el Equipo Técnico Profesional en las CRs tales como citas, consultas, remisiones del caso, expediente del ciudadano.

En la figura 7 se muestra el diseño de la BD correspondiente al módulo atención de casos. Para este módulo fue necesario crear la entidad caso con el objetivo de almacenar los datos correspondientes de los casos atendidos por el Equipo Técnico Profesional tales como el nombre del caso, la fecha de registro, la cantidad de casos, los motivos del caso, nombre de la institución por la que es referido, el status del caso: iniciado, seguimiento o cerrado y se encuentra relacionado con la tabla ciudadano que almacenará los datos referentes a la persona que se le atenderá el caso y con los nomencladores que permitirán obtener el origen y tipo de caso, la coordinación regional a la que pertenece y el tipo de actividad al que va dirigido el caso.

Un caso de acuerdo a su necesidad puede tener asociados varios tipos de atenciones y por tanto puede tener más de una remisión, debido a esto se hizo necesario establecer relación con cardinalidad de muchos a muchos entre caso y ntipo_atencion que genera la entidad rcaso_tipo_atencion en la cual se almacenará el responsable y de uno a muchos entre la entidad caso y la entidad caso_remision que se creó para almacenar los datos referentes a las remisiones que se le pueden realizar a un caso tales como la fecha, los objetivos, las observaciones, el servicio requerido, el funcionario que remite el caso, y las instituciones u organizaciones a la que va remitido, debido a esto se establece la relación entre caso_remision y la entidad ninstitucion_organismo.

Las consultas van a estar asociadas a un caso y de ellas se almacenará la fecha de la consulta, por ello se creó la entidad consulta y fue necesario relacionarla con la entidad caso. Un funcionario es el encargado de emitir observaciones de las consultas que se le realicen a un caso, estos datos se obtendrán mediante la entidad consulta_observacion que además almacena la fecha en que se realizó la observación, debido a esto fue necesario relacionar la entidad consulta_observacion y funcionario.

De las citas programadas se necesita almacenar la fecha, la hora de atención, tiempo de duración de la cita, la dirección, las observaciones que se le vayan a emitir, la asistencia a la cita, por tanto se ve la necesidad de crear la entidad cita para guardar dichos datos. Además las citas pueden tener asociados

varios tipos de atenciones tales como jurídico, social, criminológico, psicológico de ahí que surja la relación de cita con ntipo_atencion. La entidad cita fue necesario relacionarla con las entidades ciudadano y funcionario para obtener los datos correspondientes del ciudadano al que se le programó la cita y los funcionarios encargados de atenderlas según el tipo de atención.

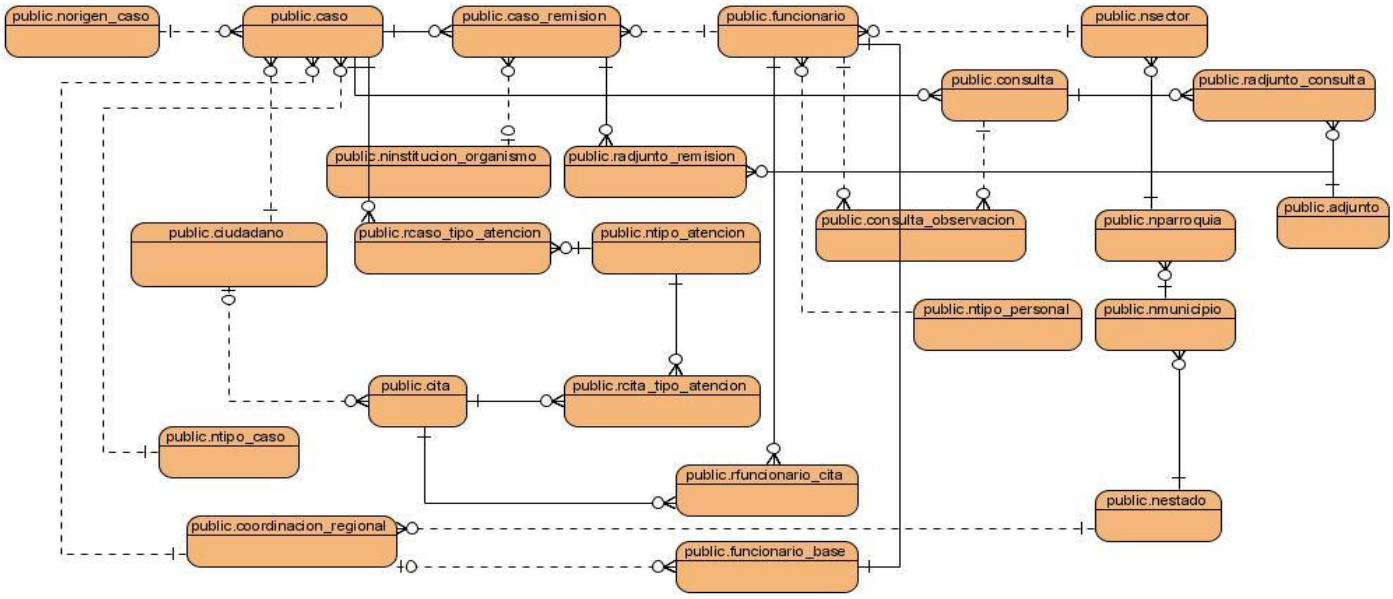

Figura 7. Módulo Atención de Casos.

2.9. Módulo Administrar Programas

2.9.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que gestionará la información correspondiente a los programas que la DGPD propone que las CRs ejecuten en cada uno de sus radios de acción según sus necesidades.

Como se muestra en el diseño (véase figura 8) se hizo necesario crear la entidad programa, de la cual se almacenará entre otros datos de interés el nombre, autores, fecha de creación, tiempo de ejecución, objetivos, debilidades y fortalezas.

Se gestionará también la información correspondiente a los proyectos de ahí la necesidad de crear la entidad proyecto para almacenar los datos referentes al nombre, autores, fecha de creación, tiempo de ejecución, objetivos, debilidades y fortalezas del proyecto.

La relación que se establece entre proyecto y programa es que un proyecto pertenece a un programa cuando los tipos de delitos a los que va dirigido el proyecto están asociados a una clasificación del delito y a su vez un programa estará dirigido a esa clasificación del delito, debido a esto surgen las relaciones entre las entidades proyecto_version y ntipo_delito, entre nclasificacion_delito y ntipo_delito y entre programa y nclasificacion_delito.

Un proyecto puede variar con el tiempo y de acuerdo con las necesidades que presente la comunidad puede ser modificado el tipo de delito al que va dirigido, cuando esto ocurre el proyecto cambia y constituye una nueva versión del proyecto, pero es necesario mantener almacenada la información correspondiente a la versión anterior y a la nueva de los proyectos para la generación de los informes. Para darle solución a esta problemática y lograr mantener con el tiempo los datos y tener un control más estricto de los cambios se vio la necesidad de crear la entidad proyecto_version que almacenará la fecha en que se realizó la modificación, de esta forma surge la relación entre proyecto y proyecto_version.

Un proyecto puede incluir realizar una serie de actividades en la comunidad de acuerdo a los tipos de delitos a los que va dirigido, de la misma forma una actividad tipo puede estar asociada a varios proyectos y a varios tipos de delitos, debido a esto se hizo necesario establecer las relaciones de muchos a muchos entre las entidades proyecto_version y actividad_tipo y entre actividad_tipo y ntipo_delito. De estas actividades tipo se conocen además las instituciones u organizaciones involucradas, los recursos necesarios para el cumplimiento de la actividad y los indicadores a medir en dicha actividad, para obtener esos datos la entidad actividad_tipo se encuentra relacionada con los nomencladores ntipo_institucion_organizacion, nrecurso y con indicador_version.

Los indicadores son utilizados para medir el impacto de las actividades, cuando a un indicador a medir en una actividad se le hacen modificaciones en el tipo de dato o en el atributo opción del indicador que puede ser de suma, promedio, moda, esto constituye una nueva versión de ese indicador, para lograr un mayor control sobre estas versiones y con el objetivo de que perduren los datos anteriores y actuales de los indicadores con el tiempo para la generación de reportes se creó la entidad indicador_version que almacenará la fecha de modificación y estará relacionada además con las entidades indicador_cadena, indicador_entero y con indicador_decimal para registrar los valores según el tipo de dato del indicador a medir.

La entidad ficha_resumen estará relacionada con programa y proyecto, la misma surge con el objetivo de mantener almacenado un resumen del programa o proyecto, la relación con esta entidad puede ser opcional debido a que no es obligatorio realizar el resumen.

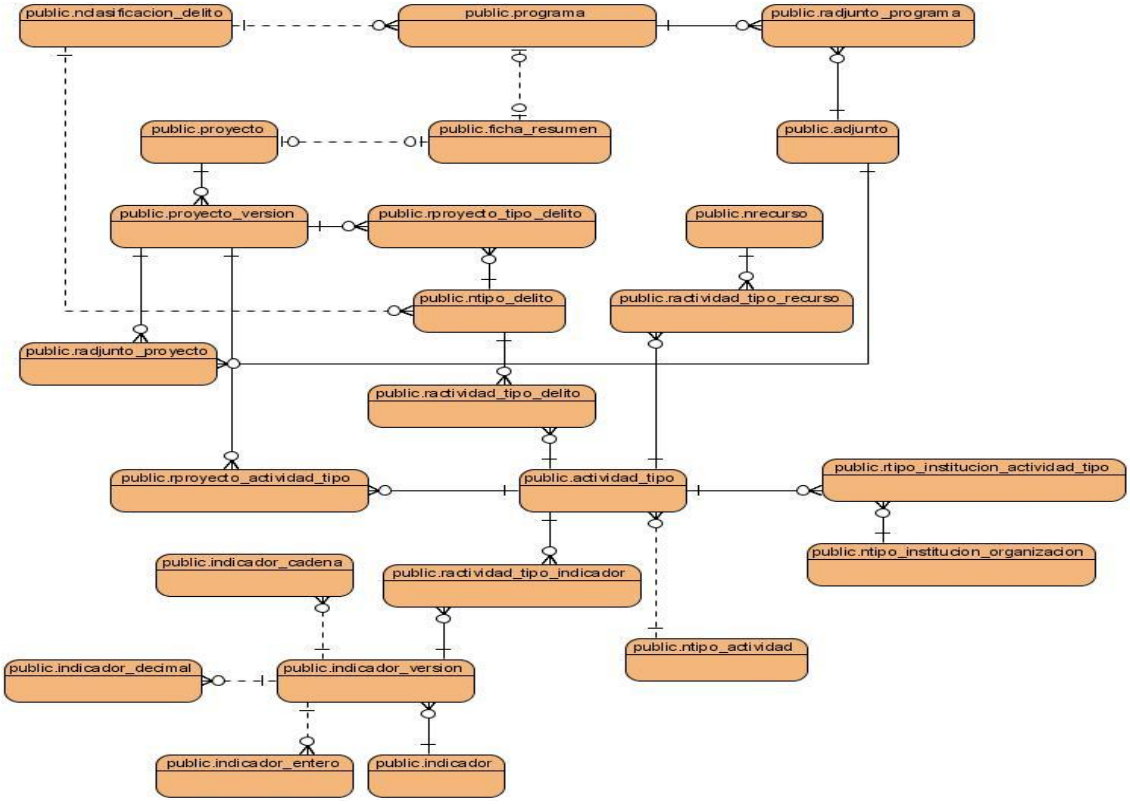

Figura 8. Módulo Administrar Programas.

2.10. Módulo Planificación de Actividades

2.10.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que gestionará el listado de las planificaciones de actividades anual, las planificaciones anuales de actividades por trimestre, la planificación anual de actividades por mes y la planificación anual de recursos. La directora de la DPCN es la persona encargada de realizar la planificación anual de actividades, la supervisora es la persona encargada de realizar la planificación anual de actividades por trimestre, el coordinador regional es el encargado de realizar la planificación anual de actividades por mes y la planificación de recursos.

Como se muestra en la figura 9, correspondiente al diseño de este módulo, para almacenar la información correspondiente a la planificación de actividades anual, trimestral, mensual y planificación de los recursos se hizo necesario crear las entidades planificacion_anual, planific_anual_trimestral, planific_anual_mensual y planif_anual_recurso que almacenan el estatus que puede ser cerrada o registrada, la fecha y las observaciones emitidas por el responsable de la planificación.

A un proyecto se le pueden realizar varios tipos de planificaciones anuales, mensuales, trimestrales y de recursos, a la vez una misma planificación puede estar vinculada a diferentes proyectos, debido a esto las relaciones que existen entre proyecto_version con las entidades planific_anual_trimestral, planific_anual_mensual y planif_anual_recurso presentan una cardinalidad de muchos a muchos. Estas planificaciones guardan relaciones entre ellas ya que de una planificación anual se realiza una planificación trimestral y a su vez de esta planificación trimestral se realiza una planificación mensual.

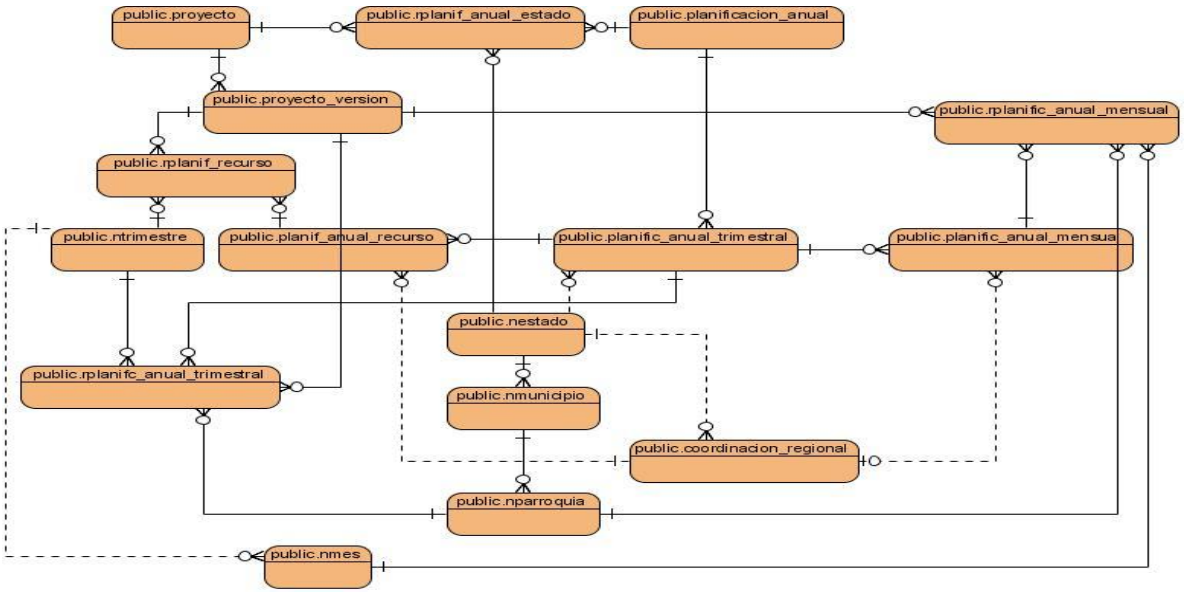

Figura 9. Módulo Planificación de Actividades.

2.11. Módulo Actividades

2.11.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que permitirá gestionar la información correspondiente a las actividades de las CRs dígase: actividades planificadas, actividades realizadas, informes del cumplimiento de actividades mensual, trimestral y anual, informe trimestral y anual de logros, por la directora o por la supervisora.

La DGPD propone una serie de programas acorde a las necesidades de cada región, grupo o problema a enfrentar que deben ejecutar en sus áreas de acción. Estos programas generan diversas actividades las cuales son orientadas a cada CR por la supervisora de la DGPD correspondiente a ese estado, cada región adapta las actividades según su realidad, por lo que las CRs deben rendir cuenta de su ejecución generando un informe que contiene todo lo realizado en el período y otras informaciones demostrando sus logros y desempeños.

En el diseño (véase figura 10) se encuentra como elemento significativo la entidad actividad que se creó con el objetivo de almacenar los datos correspondientes al nombre de la actividad, la fecha, hora de inicio y fin de la actividad, lugar y características de la población a la que va dirigida la actividad, el estado, municipio, parroquia, el nombre de la red de articulación a utilizar, el nombre y el tipo de institución u organización a abordar y estos últimos datos se obtendrán mediante las relaciones de muchos a muchos entre actividad con las entidades nredes_articulacion, ntipo_institucion_organizacion.

Una actividad puede ser planificada o realizada, debido a esto se tiene en el diseño una relación de generalización/especialización entre actividad y las entidades actividad_planificada y actividad_realizada, de esta manera podrán heredar los atributos de la entidad actividad que guarda los datos más generales además de almacenar sus propios atributos característicos que la identifican como planificadas o realizadas. Las actividades planificadas van a registrar además la justificación para el caso en que la actividad no se realice.

De las actividades realizadas se almacenarán las fortalezas, debilidades, los recursos, la accesibilidad al uso de los recursos requeridos para la ejecución de la actividad, el costo que trae consigo los recursos y si se encuentran financiados por la DGPD o por la gobernación, el nombre y el tipo de medio de comunicación empleado, de esta forma se ve la necesidad de relacionar la entidad actividad_realizada con nrecurso y nmedios_comunicacion. En las comunidades se hace necesario evaluar las actividades realizadas para conocer el grado de compromiso asumido por las comunidades, el nivel de comunicación con la comunidad, el interés de la comunidad por la actividad, el grado de confianza a la institución, la relación de la actividad con el objetivo alcanzar, estas encuestas se obtendrán mediante la relación de muchos a muchos entre actividad_realizada y el nomenclador nencuesta_comunidad. Una actividad realizada puede ser medida por diferentes tipos de indicadores con valores enteros, cadenas y decimales

de esta forma surge la relación entre indicador_version y actividad_realizada generando la nueva entidad rindicador_valor que a su vez tendría asociadas con cardinalidad de uno a muchos las entidades valor_indicador_entero, valor_indicador_cadena y valor_indicador_decimal para registrar los valores según el tipo de dato del indicador a insertar.

Para controlar los diferentes informes correspondientes al cumplimiento de las actividades según la etapa determinada anual, mensual o trimestral se hizo necesario crear las entidades informe_anual_logro, informe_cump_anual, informe_cump_mensual, informe_cump_trimestral e informe_trimestral_logro y relacionarlas con las entidades con nmes y ntrimestre. De los informes se pueden emitir observaciones sobre el cumplimiento del plan por el usuario encargado, debido a esto se establece la relación con usuario_sistema. La entidad observacion_informe es la tabla que almacenará los datos sobre las observaciones que emitan la directora nacional y la supervisora de los informes de cumplimento mensual, producto de esto mantiene una relación con cardinalidad de muchos a muchos con la tabla informe_cump_mensual.

Para el diseño se creó además la entidad datos_cualitativos que corresponde a los datos cualitativos que se emiten con respecto al informe anual y trimestral de logros y de los que se almacenan los impactos, logros alcanzados y obstáculos generándose de esta manera las nuevas entidades rdatos_cualitativos_informe_anual_logro y rdatos_cualitativos_informe_trimestral_logro respectivamente. Por la necesidad de emitir los datos cualitativos de los proyectos se creó la relación de muchos a muchos entre la entidad proyecto y datos_cualitativos que generó la entidad rdatos_cualit_proyecto donde se podrán guardar las fortalezas y debilidades que influyen sobre los proyectos en cuestión.

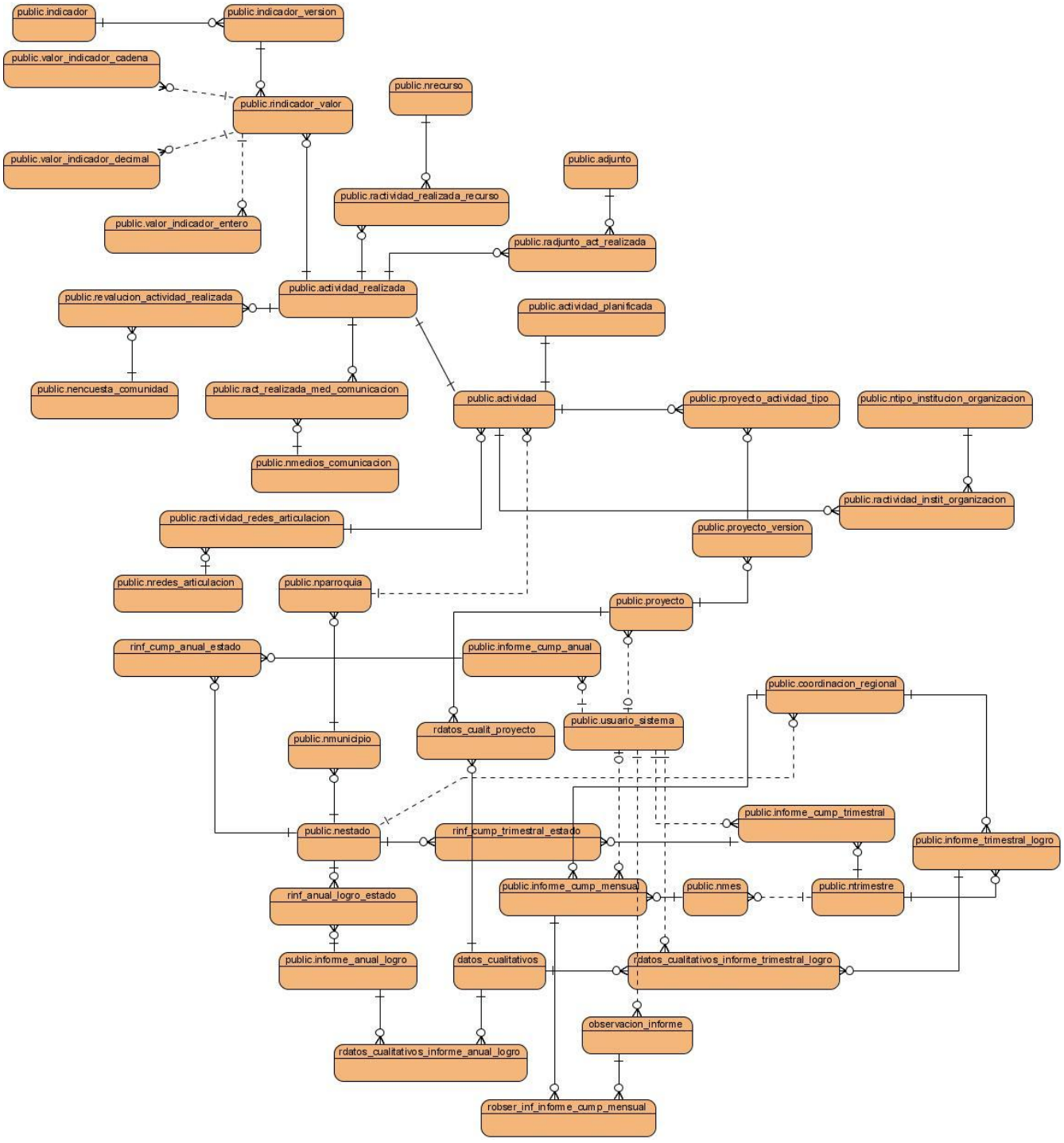

Figura 10. Módulo Actividades.

2.12. Módulo Consejo Comunales

2.12.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo destinado a la gestión de la información relacionada con los consejos comunales que será registrada en un expediente de la CR con los datos asociados a las capacitaciones brindadas, visitas domiciliarias y otros datos de interés.

En el diseño de este módulo (véase figura 11) se hizo necesario crear la entidad consejo_comunal la cual se encargará de almacenar el nombre, la fecha de abordaje, fecha de fundación, la población, el número de familias y el sector que tiene asociado. Los CCs cuentan con un grupo de personas que sirven de enlace para el trabajo entre el CC y la CR por lo que se necesitó crear la entidad miembro_persona_enlace y relacionarla con la entidad consejo_comunal. Una persona de enlace es un ciudadano por esto la relación que existe entre la entidad ciudadano y miembro_persona_enlace es una relación de generalización/especialización donde la entidad miembro_persona_enlace hereda de la entidad ciudadano que almacena los datos generales de una persona el nombre, apellidos, cédula, teléfono, sexo, dirección, profesión, correo y estado civil. La relación entre estas entidades surge por la necesidad de crear una persona con los datos característicos que la identifican como persona de enlace, por tanto solo se registrarán los datos que permitirán controlar si servirán de miembro o persona de enlace, si son voceros o no y el comité al que pertenecen, información necesaria a la hora de conocer si una persona es miembro o persona de enlace.

La entidad demanda_social se necesitó relacionarla con consejo_comunal porque almacenará la información correspondiente a las demandas sociales realizadas en los CCs tales como el nombre y descripción de la demanda, la fecha de registro, las personas involucradas, la persona que recogió la información y el funcionario receptor de la demanda que debe ser usuario del sistema para poder registrarla.

Referente a los CCs se pueden emitir observaciones, debido a esto se crea la entidad observacion y se establece la relación con consejo_comunal. De estas observaciones se almacenará la información relacionada con la fecha de realización, el nombre correspondiente a la observación, la descripción detallada, el nombre de la persona que recogió la observación, el funcionario receptor de la información y el CC al que pertenece.

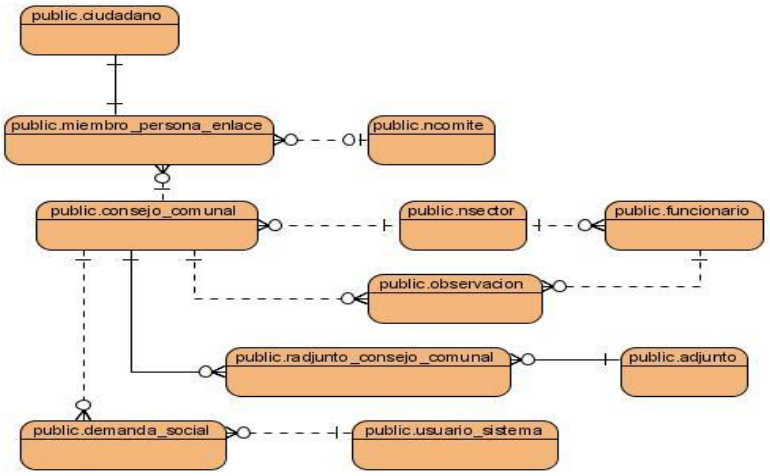

Figura 11. Módulo Consejos Comunales.

2.13. Módulo Diagnóstico

2.13.1. Descripción del Módulo y Elementos Representativos del Diseño

Módulo que permitirá gestionar los diagnósticos realizados a los CCs abordados por las CRs, así como sus estadísticas. Los diagnósticos tienen el objetivo de conocer los principales problemas económicos, políticos y sociales que presenta la comunidad. En el aspecto político se necesita gestionar los diversos comités, instituciones públicas, las diferentes misiones que operan en la comunidad. En el aspecto social es necesario conocer los diferentes servicios que se prestan en la población, el nivel de educación escolar predominante, los problemas de deserción escolar existente, el tipo de construcción predominante en las viviendas para conocer la calidad de vida existente. También se lleva a cabo un estudio de las diferentes actividades que se realizan tanto culturales como deportivas recreativas, los tipos de delitos que más se manifiestan, los lugares donde ocurren con frecuencia, las estrategias para minimizar la delincuencia, las instituciones y organismos que están estrechamente vinculados con la prevención del delito. En el orden económico se necesita conocer el nivel de ingreso promedio y de desempleo que existe en las familias, los principales establecimientos comerciales que tienen un mayor impacto económico y social en la comunidad, los tipos de proyectos que se realizan en el CC y los comités encargados de su cumplimiento.

En la figura 12 se evidencia el diseño del módulo Diagnóstico. Este modelo permite definir las situaciones antes planteadas con el uso principalmente de nomencladores lo que brinda mayor flexibilidad a la hora de gestionar la información que se necesita recopilar de los CCs.

Como parte del diseño se tiene la entidad diagnóstico que se creó con el objetivo de almacenar la información que se requiere gestionar de los CCs en las diferentes esferas, para ello fue necesario relacionarla con las distintas entidades que permitirán obtener dicha información como nmision, ncomite, nistitucion_denuncia, ndesempleo_comunidad, nestrategia, nestablecimiento, nnivel_ingreso_promedio, entre otras. En el diseño casi en su totalidad se presentan relaciones con cardinalidad de muchos a muchos entre diagnostico y estas entidades.

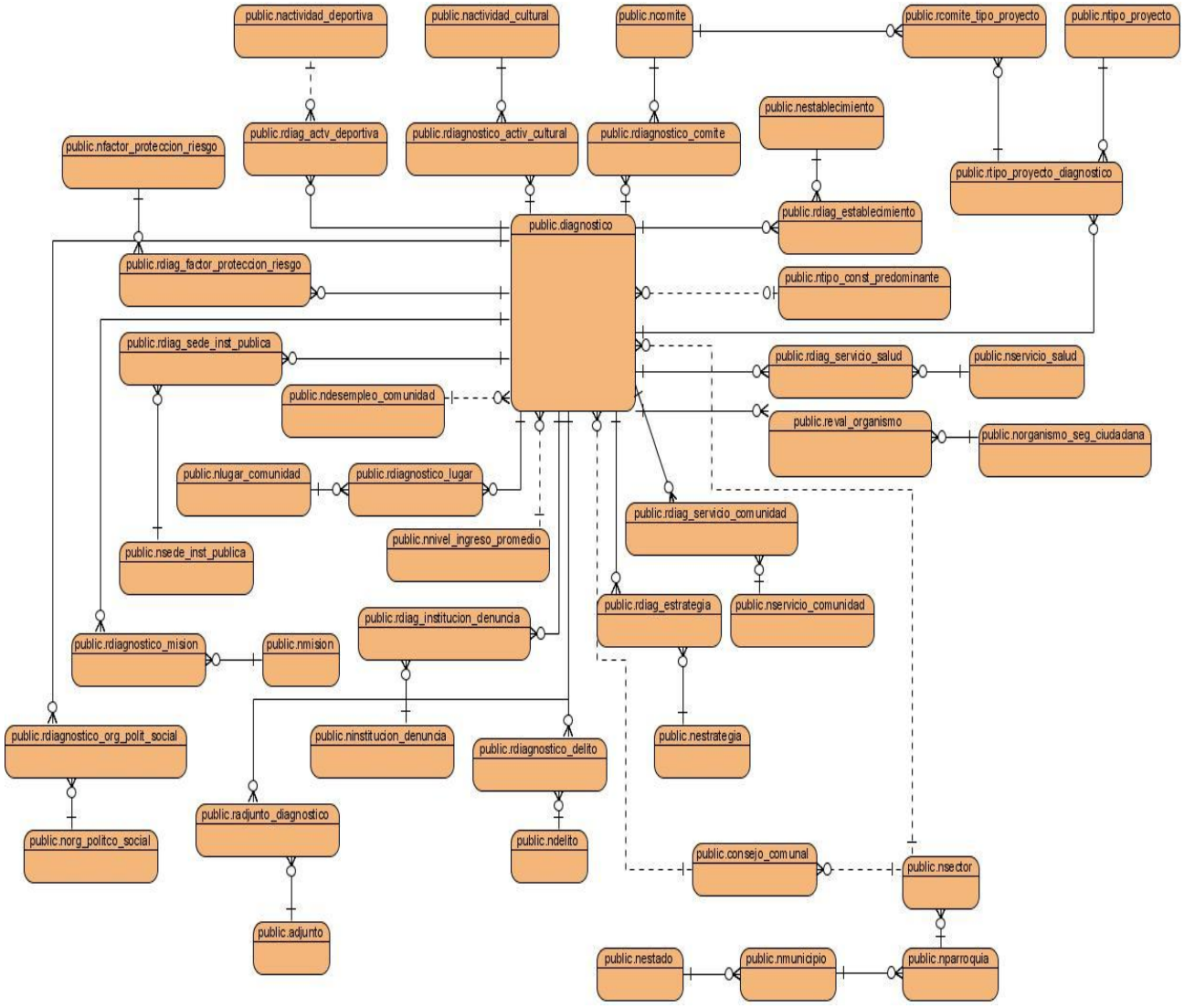

Figura 12. Módulo Diagnóstico.

2.14. Módulo Administrar Mapa

2.14.1. Descripción del Módulo y Elementos Representativos del Diseño

Este módulo permitirá gestionar las imágenes de mapa del país, estados y municipios que se visualizarán en la aplicación mostrando la cantidad de CCs abordados por las CRs.

Entre los elementos significativos en el diseño de la BD de este módulo (véase figura 13) se encuentran las relaciones entre las entidades nestado, nmunicipio y nparroquia con cardinalidad de uno a muchos y fue necesario establecer dichas relaciones debido a que un estado puede tener varios municipios, al igual que un municipio puede tener asociadas varias parroquias. Se hizo necesario además crear las entidades parroquia_imagen, municipio_imagen y estado_imagen para poder almacenar la imagen del CC que se desea mostrar y las coordenadas de los puntos que representan los estados, municipios y parroquias en dichas imágenes. Como cada una de estas entidades estará relacionada con la parroquia, municipio y estado correspondiente las relaciones entre ellas serán de uno a uno porque para cada parroquia, municipio y estado existirá una sola imagen.

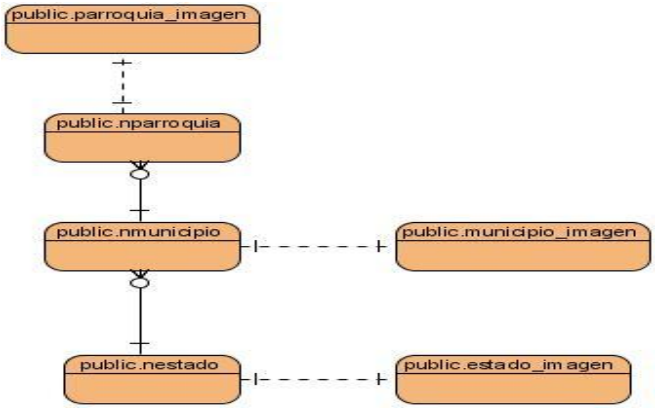

Figura 13. Módulo Administrar Mapas.

2.15. Conclusiones

En este capítulo se realizó una descripción del funcionamiento de los procesos del negocio a modelar en el sistema, lo que posibilitó realizar el diseño de BD que se utilizará para gestionar la información generada en los mismos. Se describieron los elementos significativos del diseño propuesto para cada uno de los módulos reflejando de ellos las entidades más relevantes para el Sistema de Gestión de Información de las Coordinaciones Regionales con sus interrelaciones y propiedades. Este diseño brinda mucha flexibilidad y ventajas desde el punto de vista de la aplicación debido a la creación y el uso de

nomencladores que facilitarán el trabajo en el sistema. La asignación de roles y la forma de autenticación definida en el diseño permite que sólo los usuarios autorizados tengan acceso sobre cada una de las funcionalidades del sistema según los privilegios que posean. El diseño propuesto permite darle solución a los problemas referentes a la recopilación de información de las actividades que se realizan en las CRs de cada estado y en la DGPD con los asuntos relacionados al manejo de los CCs, generación de los reportes necesarios para el trabajo diario e información a la población de los proyectos que se llevan a cabo por la Dirección.

CAPÍTULO 3: COMPLEMENTO DEL DISEÑO DE LA BASE DE DATOS

3.1. Introducción

En el capítulo se muestra la distribución de almacenamiento lógica y física de la BD para el sistema, el tema de la seguridad enfocados en los permisos que deben tener los usuarios que se conecten a la BD y las tareas programadas sobre la BD que garantizan algunos requisitos para el Sistema de Gestión de Información de las Coordinaciones Regionales. Se describen las reglas básicas de salvas y recuperación definidas en el servidor para la disponibilidad de la información en momentos de fallas. Esta parte del trabajo muestra algunos elementos de la solución que complementan el diseño de la BD.

3.2. Estructura de Almacenamiento de la Base de Datos

3.2.1. Espacios de tablas (Tablespaces)

Una BD se puede encontrar dividida en una o más unidades lógicas llamadas tablespaces, que son utilizados para separar la información en grupos y así simplificar la administración de los datos.

Los tablespaces en PostgreSQL permiten al Administrador de Base de Datos definir localizaciones alternativas en el sistema de ficheros donde los archivos que representan objetos en la base de datos pueden ser almacenados. Una vez creado, un tablespace puede ser referenciado por su nombre a la hora de crear objetos de la base de datos. (22)

Mediante el uso de espacios de tablas el administrador puede controlar la estructura del disco de instalación de PostgreSQL. Esto es útil por lo menos en dos formas: (22)

- ✓ Si la partición o volumen en el que el grupo se ha inicializado se queda sin espacio y no puede extenderse, un espacio de tablespace se puede crear en una partición diferente y se utiliza hasta que el sistema puede ser reconfigurado.
- ✓ Permitir a un administrador aplicar el uso de objetos de base de datos para optimizar el rendimiento. Por ejemplo, un índice que es muy usado puede ser colocado en un medio de almacenamiento más rápido, como un costoso dispositivo de estado sólido. Al mismo tiempo, una tabla para almacenar datos que rara vez se utiliza o los resultados no son críticos podrían estar almacenados en un sistema de disco menos costoso o más lento.

Existen varias razones que justifican este modo de organización de las tablas en espacios de tablas: (23)

- ✓ Permiten distribuir a nivel físico los distintos objetos de las aplicaciones.
- ✓ Un espacio de tablas puede quedarse offline debido a un fallo de disco, permitiendo que el SGBD continúe funcionando con el resto.

3.2.1.1. Tablespaces Definidos

Las tablas, índices y bases de datos completas pueden ser asignados a espacios de tablas en particular. Para el Sistema de Gestión de Información de las Coordinaciones Regionales de Prevención del Delito se proponen 3 espacios de tablas. Esto brindará ventajas para:

- ✓ La disponibilidad de la información del sistema porque de ocurrir un error o dañarse un tablespace se puede continuar accediendo al resto de la información sin problemas.
- ✓ La velocidad de respuesta porque al estar la información en dispositivos diferentes las operaciones de entrada/salida sobre los datos almacenados en diferentes tablespaces se realizarán con recursos físicos diferentes y la cantidad de información será menos en cada tablespace algo que también brindará mayor velocidad de ejecución.
- ✓ La seguridad de la información porque al tener la información organizada por tablespaces mejora las operaciones de salvas (backup) y recuperación de los datos, porque estas pueden ser realizadas a nivel de tablespaces sin tener que afectar a otras.

Teniendo en cuenta las ventajas de tener la solución almacenada en tablespaces a continuación se presentan los tablespaces definidos y una breve descripción de las características de los datos a almacenar en ellos y su uso en la aplicación.

- ✓ **ts__nomencladores:** Tablespace creado con el propósito de almacenar los datos que son accedidos con mayor frecuencia por el sistema, con el fin de lograr un mayor rendimiento, en este caso se tendrá almacenada la información correspondiente a los nomencladores.

- ✓ **ts _indices:** Tablespace creado para almacenar los índices. Si bien los índices pertenecen lógicamente a una tabla, estos son estructuras físicas con crecimiento independiente de los objetos sobre los cuales actúan. Unas de las ventajas que tiene la creación de un tablespace para los índices es que si el mismo sale de línea como producto de la falta de capacidad en disco, la información de las tablas puede seguir siendo consultada.

- ✓ **ts_datos:** Tablespace utilizado para el almacenamiento del resto de la información que se recoge en el sistema.

3.2.1.2. Creación y Manipulación de los Tablespaces.

Antes de crear una BD, se pueden crear los tablespaces donde se guardarán los datos de la misma, si bien se pueden utilizar los tablespaces por defecto de PostgreSQL (*pg_default* y *pg_global*) es recomendable crear tablespaces separados.

Para definir un tablespace se utiliza en comando CREATE TABLESPACE y la ubicación debe ser un directorio existente y vacío, posteriormente, todos los objetos creados dentro de las tablas se almacenan en los archivos bajo este directorio. Para poder crear los tablespaces en una determinada localización es imprescindible que las ubicaciones asignadas pertenezcan al usuario administrador del sistema que se creó en la instalación de PostgreSQL, por eso es necesario tener ciertos privilegios en la BD, cualquier usuario no debe tener permiso a realizar estas operaciones porque pueden alterar la configuración de almacenamiento que se tiene definida para la BD del sistema.

La sentencia para la creación de un tablespace en PostgreSQL quedaría de la siguiente forma:

```
create tablespace <nombre_tablespace> owner <usuario> location <direccion_fisica>;
```

Para remover un tablespace se utiliza en comando DROP TABLESPACE y para determinar el conjunto de tablespaces existentes, se consulta en catálogo del sistema *pg_tablespace*, por ejemplo:

```
SELECT spcname FROM pg_tablespace;
```

Los tablespaces definidos quedarían especificados de la manera siguiente:

```
create tablespace "ts_nomencladores" owner "postgre" location 'mnt/ prevencion1 /nomencladores';
create tablespace " ts _indices" owner "postgre" location 'mnt/ prevencion2 /indices';
create tablespace "ts_datos" owner "postgre" location 'mnt/ prevencion3 /resto';
```

De esta forma quedan conformados los espacios de tablas que se proponen para el Sistema de Gestión de Información de las Coordinaciones Regionales utilizados para especificar ubicaciones físicas alternativas para el almacenamiento de los objetos que contiene la BD.

3.2.2. Usuarios de Base de Datos

Para acceder a los datos en una BD PostgreSQL se debe tener acceso a una cuenta en esa BD y cada cuenta debe tener una palabra clave asociada. Las claves son fijadas cuando se crea un usuario y pueden ser alterados por el Administrador de la Base de Datos o por el usuario mismo. Las contraseñas de usuarios son almacenadas en texto plano en la tabla de sistema *pg_shadow*, pero sólo los superusuarios de PostgreSQL tienen permiso para ver la tabla. Los usuarios para el Sistema de Gestión de Información de las Coordinaciones Regionales son creados con dos propósitos, para acceder desde el sistema y para administrar la BD. En esta sección se abordarán los principales elementos para la creación de los usuarios utilizados en el acceso y administración de los datos del sistema.

3.2.2.1. Creación de Usuarios

El objetivo de la creación de usuarios es establecer una cuenta segura y útil, que tenga los privilegios adecuados y los valores por defecto apropiados. En PostgreSQL se puede especificar todo lo necesario para crear una cuenta con el comando CREATE USER. Esto es muy importante para el sistema ya que se puede gestionar los niveles de acceso a la BD en dependencia del objetivo con que se acceda, ya sea para administrar la información o para modificarla, en el caso del acceso por parte del sistema. Los parámetros que se deben tener en cuenta para crear un usuario son:

```
create user name [[with] option [,...]]
```

Donde: Las opciones (option) pueden ser las siguientes:

```
superuser | nosuperuser
createdb | nocreatedb
createrole | nocreaterole
createuser | nocreateuser
inherit | noinherit
login | nologin
connection limit connlimit
[encrypted | unencrypted] password ´password´
valid until ´timestamp´
in role rolename [, ...]
in group rolename [, ...]
role rolename [, ...]
admin rolename [, ...]
user rolename [, ...]
sysid uid
```

En la BD del sistema se crearon dos usuarios: *preven* y *prevencion_sistema*. El primero con el objetivo de administrar la BD y el segundo para acceder a la información desde el sistema, cada uno de estos con los

privilegios específicos para realizar la tarea por la que fueron creados. Para crear estos usuarios se utiliza la sentencia siguiente:

```
CREATE USER nombre_usuario
[WITH [SYSID uid]
[PASSWORD 'password']]
[CREATEDB | NOCREATEDB]
[CREATEUSER | NOCREATEUSER]
[IN GROUP groupname [,...]]
[VALID UNTIL 'abstime']
```

3.2.3. Asignación de Privilegios a los Usuarios

Los privilegios de sistema son asignados a los usuarios con el fin de determinar los permisos sobre los recursos y objetos del servidor de BD. Estos deben ser asignados cuidadosamente, teniendo presente el fin con el que se crea el usuario, asignándole solamente los privilegios necesarios para realizar su función en la BD. Los privilegios pueden ser asignados sobre objetos (tablas, índices, vistas, funciones) de la BD.

Los privilegios pueden agruparse en roles de forma tal que al asignarle un rol a un usuario éste tendrá el conjunto de privilegios agrupados en el rol lo que facilita el trabajo del Administrador de la Base de Datos ya que de lo contrario tendría que asignarles a los usuarios privilegios por privilegios. A continuación se muestran los roles y privilegios que son utilizados para los usuarios de la BD del sistema:

Rol dba: Todos los privilegios del sistema con la opción with admin option.

- ✓ Superusuario.
- ✓ Creación de objetos de BD.
- ✓ Creación de roles.
- ✓ Modificación del catálogo directamente.

3.2.4. Creación de Roles

Los privilegios se pueden agrupar en roles, de forma tal que se puedan identificar diferentes tipos de usuarios. El SGBD tiene por defecto un conjunto de roles, con los cuales identifica a un grupo determinado de usuarios, pero es posible que algunos de estos no se adapten a determinadas características de un usuario, en estos casos se pueden crear roles que identifiquen a grupos de usuarios en la BD.

Para la BD del sistema se tienen dos usuarios; *preven* que se crea con el fin de que administre la BD, es decir que tendría el rol de dba, y el usuario *prevencion_sistema* el cual permite que el sistema acceda a la información, tendría el rol *sistema_rol_prevencion*. Todas las acciones desde el sistema sobre los datos

se harán por medio de consultas o vistas, es por esto que el usuario *prevencion_sistema* debe tener los permisos de conectarse a la BD y además tener permisos para poder insertar, actualizar y eliminar datos de las tablas, así como permiso de selección de vistas y uso de secuencias. Los privilegios se asignan y eliminan mediante las sentencias GRANT y REVOKE. PostgreSQL define los siguientes tipos de operaciones sobre las que se pueden dar privilegios: Select, Insert, Update, Delete, Rule, References, Trigger, Create, Temporary, Execute, Usage, y All privileges. En el caso del rol *prevencion_sistema* se le dará privilegios sobre las operaciones de Select, Insert, Update, Delete, Usage o Trigger en dependencia del tipo de objeto de BD.

A continuación se muestra la sentencia para crear el rol *sistema_rol_prevencion*.

```
CREATE ROLE "sistema_rol_prevencion" LOGIN CONNECTION LIMIT 10 PASSWORD '******';
GRANT INSERT, UPDATE, DELETE ON "public"."nombre de objeto" TO "
sistema_rol_prevencion ";
```

3.2.4.1. Asignación de Roles.

Luego de creados los roles estos se deben asignar a los usuarios de la BD. A continuación se muestra como asignarle los roles a los usuarios de la BD:

```
CREATE ROLE "preven " SUPERUSER CREATEDB LOGIN dba;
CREATE ROLE " prevencion_sistema " SUPERUSER CREATEDB LOGIN sistema_rol_prevencion;
```

3.2.5. Creación de Esquemas

Una BD engloba un conjunto de esquemas y estos organizan lógicamente los objetos de la BD pero no tienen nada que ver con el almacenamiento físico de ellos. Una BD puede tener uno o más esquemas que pueden contener tablas, índices, procedimientos, vistas, pero además otros tipos de objetos incluyendo tipos definidos por el usuario, funciones y operadores.

El esquema es propiedad de un usuario de la BD y para poder crear o acceder a los objetos de un esquema se realiza mediante un nombre compuesto por el nombre del esquema y el nombre del objeto separado por un punto, quedando la notación *esquema.objeto*.

En PostgreSQL cada BD se crea con un esquema especial nombrado "*public*". Por defecto, todos los objetos que se creen en la BD irán para ese esquema. La sentencia de creación de esquemas en PostgreSQL tiene la siguiente estructura:

```
CREATE SCHEMA schemaname [AUTHORIZATION username] [schema element [...]];
```

Donde:

- ✓ **schemaname:** Nombre del esquema que será creado, si el nombre es omitido, será utilizado el nombre del usuario como nombre del esquema.
- ✓ **username:** Nombre del usuario que será propietario del esquema, si se omite, el propietario del esquema es el usuario que ejecuta la sentencia.
- ✓ **schema_element [...]:** Permite sentencias DDL dentro de la creación del esquema.

La BD del sistema contará con tres esquemas utilizados para organizar de manera más comprensible y lógica la información que se necesita gestionar. Dentro de los esquemas se tiene el esquema por defecto *public* donde se tendrán todas las tablas y los objetos de la BD que utilizan cada uno de los módulos que conforman el sistema. Se tienen además los esquemas *administracion_programas* y *actividades* donde se tendrán sólo los objetos que serán utilizados por los módulos Administrar Programas y Actividades respectivamente para los cuales se tuvo la necesidad de crearlos debido a que estos son módulos complejos que gestionan numerosos elementos correspondientes a los programas, proyectos, el manejo y control de sus versiones por lo que fue necesario separarlos hacia nuevos esquemas para tener de forma más accesible y organizada los objetos, específicamente las funciones, vistas y disparadores que necesitan estos módulos.

De esta forma quedan conformados los esquemas que se proponen para el Sistema de Gestión de Información de las Coordinaciones Regionales utilizados para organizar lógicamente los objetos de base de datos relacionados:

```
CREATE SCHEMA "administracion_programas" AUTHORIZATION "postgres";
CREATE SCHEMA "actividades" AUTHORIZATION "postgres ";
```

3.2.6. Tareas Programadas

Las tareas programadas o trabajos (jobs) de la BD son instrucciones que se ejecutan sin ser invocadas por los usuarios. Las tareas son acciones configuradas para que se ejecuten en instantes de tiempos definidos por el Administrador de la Base de Datos. En este caso se utilizará cron, que es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a

intervalos regulares. La tarea programada que se creó se encargará de eliminar de la BD todos los adjuntos que quedaron colgados en el sistema al final del día. El trabajo con los adjuntos consta de dos momentos, un primer momento en el que se inserta un adjunto, donde se garantiza que los adjuntos no estarán cargados en memoria, logrando un mayor rendimiento desde el punto de vista de la aplicación y un segundo momento en el que se asocian los datos correspondientes del módulo al adjunto. Un adjunto colgado es aquel adjunto que se insertó en la BD y que no se relaciona con ninguno de los módulos.

3.2.6.1. Configuración de la Tarea.

La configuración de las tareas programas con cron es un procedimiento sencillo, consiste en localizar en su sistema UNIX el archivo *crontab* del usuario que se encargará de ejecutar dicha tarea en el intervalo de tiempo seleccionado, en la mayoría de estos sistemas se puede localizar este archivo en el directorio /HOME de dicho usuario. Una vez localizado se especifica la hora, minuto, mes, año y día en que se desea ejecutar dicha tarea, especificando además el archivo que contiene el script a ejecutar en este caso sería *delete_adjunto.sh* y guardando los logs generados en el archivo *delete_adjunto.log.* En la figura 14 se muestra la manera de incluir una tarea en el archivo *crontab*.

```
 GNU nano 2.0.9                    Archivo: /tmp/crontab.aX9CUp/crontab

SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
HOME=/home/servidor


# m h  dom mon dow   command
  30  13    * * * /home/servidor/delete_adjunto.sh > /home/servidor/logs/delete_adjunto.log

¬
```

Figura 14. Forma de Incluir una Tarea en el Archivo Crontab.

3.3. Creación de las Políticas de Salvas (backup) y Recuperación (recovery) de la BD

La información siempre está expuesta a daños diversos originados por infecciones del sistema, fallos de hardware (cortes de corriente y picos de tensión, excesos de temperatura y daños en los dispositivos), apagados incorrectos del equipo, accidente o problemas motivados por algún software que podrían provocar la pérdida de datos.

Para conseguir un funcionamiento seguro de la BD y una pronta recuperación ante cualquier tipo de fallas se necesita planear una estrategia de copias de seguridad y de recuperación.

3.3.1. Copias de Seguridad (backup)

Una copia de seguridad, también denominada copia de respaldo o backup, es una copia adicional que se le realiza al contenido en este caso de la BD con la finalidad de disponer de la información en situaciones de pérdida o daños en el equipo y la garantía de poder recuperarlos a partir de un estado no utilizable de la misma. La periodicidad de estos backups, se debe establecer teniendo en cuenta la importancia de la información que se maneja que será en función del tamaño de los datos y de la frecuencia con que son modificados, lo cual marcará de esta forma la estrategia de copias de seguridad.

Generalmente, si la estrategia de backup se basa en la copia de los ficheros de datos y en el archivado de los ficheros de log (WAL), se han de tener copias de ambos. Si se pierde uno de los ficheros de log se dice que se tiene un agujero en la secuencia de ficheros, esto invalida el backup, pero permite a la BD ser llevada hasta el principio del agujero realizando una recuperación incompleta.

Es necesario comprender algunas reglas que determinan la situación de los ficheros y otras consideraciones que afectarán al esquema de backup: (23)

- ✓ Es recomendable archivar los ficheros de log en disco, y luego copiarlos a cinta, pero siempre en un disco diferente del que soporta los ficheros de datos.
- ✓ Los ficheros copias no deben estar en el mismo dispositivo que los originales. No siempre hay que pasar las copias a cinta, ya que si se dejan en disco se acelera la recuperación. Además, si se copian las copias a cinta y se mantienen en el disco, se puede sobrevivir a diversos fallos de dispositivo.

3.3.1.1. Copias de Seguridad de Ficheros del SO

Es el método más sencillo, pero el más ineficaz, se trata de realizar una copia de todos los ficheros de un clúster, por ejemplo: (24)

```
$ su - postgres
$ cd /tmp/backup
$ tar cvfz copia.tar.gz $PGDATA
```

Utilizar este método trae consigo algunas desventajas: (24)

- ✓ La base de datos debe estar parada.
- ✓ No se pueden recuperar partes del clúster (bases de datos, esquemas, tablas, etc.)

- ✓ La recuperación consiste en borrar todo el clúster y descomprimir el fichero de copia de seguridad, con lo que se pierden los datos que se hayan modificado desde la última copia de la base de datos.

3.3.1.2. Volcado SQL Usando las Herramientas PostgreSQL

Los volcados de este tipo se realizan usando las herramientas que proporciona PostgreSQL. Estos volcados son muy flexibles y de gran utilidad, permiten hacer copias de seguridad de toda la base de datos o de partes de ella, y luego, dada una copia de seguridad, permiten restaurar lo que quiera.

Además, estas herramientas sirven para la transmisión de datos entre bases de datos, las herramientas que se pueden utilizar: (24)

- ✓ **pg_dump:** Vuelca una base de datos o parte de ella a un fichero, bien en texto plano o en un formato propio de PostgreSQL. Se puede recuperar cualquier objeto que esté en el fichero aisladamente. El servidor debe estar en marcha.

- ✓ **pg_dumpall:** Vuelca un clúster completo

- ✓ **pg_restore:** Recupera los objetos volcados en una copia de seguridad que no se realizó en texto plano sino en un formato propio de PostgreSQL.

- ✓ **psql:** Se usa para recuperar los volcados en texto plano.

Utilizar este método trae consigo algunas desventajas: (23)

- ✓ En grandes bases de datos el proceso de volcado hacia un fichero puede demorar.
- ✓ Dependiendo del tamaño de la base de datos, el proceso puede generar un fichero de mayor tamaño que el que soporta el sistema de ficheros.
- ✓ Se pierde toda la información sobre las transacciones que se están realizando en el momento de la realización del backup.

3.3.1.3. Volcado en Línea y Recuperación PITR

Un sistema ejecutando PostgreSQL produce una larga secuencia indefinida de los registros de WAL. El registro describe cada cambio realizado a los archivos de datos de la base de datos. Este registro existe principalmente para propósitos de seguridad si el sistema se bloquea, la base de datos se puede restaurar

a la coherencia de reproducir las entradas del registro realizados desde el último punto de control. El sistema se divide físicamente en los archivos de esta secuencia de segmentos WAL, que normalmente son de 16 MB cada uno y recicla los ficheros de log que no van a ser necesitados renombrándolos a números superiores dentro de la secuencia. Estos ficheros tienen un nombre único y se generan en el subdirectorio *pg_xlog* que se encuentra en el directorio de datos ($PGDATA) usado por PostgreSQL. El número de ficheros WAL contenidos en *pg_xlog* dependerá del valor asignado al parámetro *checkpoint_segments* en el fichero de configuración *postgresql.conf.*

Para activar el archivado, en el fichero *postgresql.conf* se debe indicar el comando de copia para preservar los ficheros de log en el parámetro *archive_command*

El procedimiento para realizar una copia de seguridad en línea sería:

- ✓ Asegurarse de que el archivado WAL está habilitado y en funcionamiento.
- ✓ Antes de empezar y desde una consola SQL hay que ejecutar:
  ```
  select pg_start_backup('nombre_copia');
  ```
- ✓ Con el servidor en marcha, hacemos la copia desde el sistema operativo, no hace falta parar el servidor, por ejemplo: $ tar –cvf backup_nombre_copia.tar $PGDATA.
- ✓ Cuando acaba, desde la consola SQL, marcamos el final, ejecutando:
  ```
  select pg_stop_backup();
  ```
- ✓ Se crea así un fichero de marca en el directorio $PGDATA/pg_xlog/archive_status y copia los logs que se reciclan en donde se haya indicado en *archive_command*.

Este método tiene como principal ventaja que almacena la información de las transacciones confirmadas cuando se está realizando el proceso de salva.

Utilizar este método trae consigo algunas desventajas: (23)
Como este método salva todos los ficheros de log, más el directorio de datos del servidor PostgreSQL, tiene como principal desventaja el tamaño que puede alcanzar el backup.

3.3.1.4. Definición de las Políticas de Salvas para la Base de Datos del Sistema

Para realizar las copias de seguridad de la BD para el Sistema de Gestión de Información de las Coordinaciones Regionales se propone el Volcado en Línea que es la técnica generalmente utilizada en

situaciones donde se requiere una alta fiabilidad, además este brinda la posibilidad de que las copias se efectúen con la BD funcionando por lo que no hay que detener la misma, facilita también almacenar toda la información sobre las transacciones que se están realizando en el momento de la realización del backup lo que contribuye a que ningún dato se pierda, permite salvar todos los ficheros de log más el directorio de datos del servidor incluyendo los archivos de configuración del posgreSQL. El backup a utilizar es un backup físico de tipo en caliente debido a que los datos se copian físicamente los ficheros de la BD mientras la misma se encuentra abierta y en funcionamiento. Se propone además realizar un backup completo de la BD los fines de semana. Los pasos para la preparación y realización en PostgreSQL quedarían definidos de la siguiente manera:

Se establece el parámetro de configuración *archive_mode* a on, y se crea un comando para la opción archive_command que realice el archivado de los archivos de log. Por ejemplo:

```
archive_command = 'test! -f /var/lib/pgsql/backup_in_progress || cp -i %p
/var/lib/pgsql/archive/%f < /dev/null'
```

Donde:

„%p" representa el nombre del fichero de log con la ruta absoluta y „%f" sin la ruta.

Este comando realizará el archivado cuando exista el fichero /var/lib/pgsql/backup_in_progress que funciona como "archivo interruptor", si no existe este archivo, silenciosamente retornará cero como código de salida (permitiendo a PostgreSQL reciclar el fichero de log). Luego de esta preparación se puede realizar el backup usando un script como el siguiente:

```
touch /var/lib/pgsql/backup_in_progress
psql -c "select pg_start_backup('hot_backup');"
tar -cf /var/lib/pgsql/backup.tar /var/lib/pgsql/data/
psql -c "select pg_stop_backup();"
sleep 20
rm /var/lib/pgsql/backup_in_progress
tar -rf /var/lib/pgsql/backup.tar /var/lib/pgsql/archive/
```

Primeramente es creado el fichero /var/lib/pgsql/backup_in_progress, permitiendo el archivado de todos los ficheros de log que se puedan crear en el tiempo que dure el proceso de creación del backup, luego de crearse el backup se borra el archivo interruptor y por último los archivos de log archivados se añaden al backup.

3.3.2. Recuperación de la Base de Datos (recovery)

Los datos deben ser repuestos, recurriendo entonces a la información almacenada en la copia de seguridad. La recuperación de los datos deberá ser efectuada rápidamente y de forma eficiente, para que los servicios no se encuentren inactivos por mucho tiempo. La prioridad de la reposición de los datos debe ser establecida conforme a las necesidades de la organización.

Para realizar una recuperación frente a los distintos problemas que puedan provocar la caída del servidor utilizando un archivo backup con la última copia del mismo se pueden seguir los siguientes pasos:

- ✓ Se detiene el servicio del PostgreSQL, si está en ejecución.
- ✓ Borrar todo los ficheros y subdirectorios existentes bajo el directorio de datos del clúster y de la carpeta raíz de los tablespaces.
- ✓ Restaurar los ficheros de la base de datos desde el último archivo de salva.
- ✓ Crear un fichero de recuperación recovery.conf en el directorio de datos del clúster, en este fichero hay una serie de parámetros que pueden ayudar a recuperar hasta el momento o la transacción que se quiera y es lo que se conoce como recuperación Point-in-time (recuperación a un punto del tiempo):
    - **restore_command:** Lo que ejecuta esta variable es lo que PostgreSQL ejecutará antes de empezar la recuperación. Por ejemplo:
      ```
      restore_command = 'cp /mnt/server/archivedir/%f %p'
      ```
    - **recovery_target_time:** Hasta qué momento.
    - **recovery_target_xid:** Hasta una transacción determinada.
    - **recovery_target_inclusive:** Si los dos casos anteriores son inclusivos o no.
- ✓ Iniciar el servidor, el servidor arrancará en modo recuperación y procederá a leer los ficheros de log archivados que necesita. Una vez completado el proceso de recuperación, el servidor renombrará el fichero recovery.conf a recovery.done (para prevenir que accidentalmente vuelva a entrar en modo recuperación si vuelve a caer) y comenzará las operaciones normales sobre la base de datos.
- ✓ Inspeccionar el contenido de la BD para asegurarse que se haya restaurado lo que se desea.

Es necesario efectuar pruebas sistemáticas de las salvas que se hayan realizado para verificar la integridad de las mismas y prevenir cualquier incidencia sobre estos ficheros.

De esta forma quedan conformadas las políticas de salvas y recuperación de la BD que se proponen para el Sistema de Gestión de Información de las Coordinaciones Regionales.

3.4. Conclusiones

En este capítulo se definieron algunas de las actividades a realizar para llevar a cabo la gestión y configuración en el servidor de la BD para el Sistema de Gestión de Información de las Coordinaciones Regionales de forma tal que permita obtener un sistema estable con una buena integración y una alta disponibilidad. Se determinaron elementos importantes como la distribución física y lógica de almacenamiento, la seguridad establecida en el servidor basada en los permisos que deben tener los usuarios que se conecten a la BD, las tareas programadas definidas en el servidor, además las políticas establecidas para alcanzar un funcionamiento seguro de la BD mediante las copias de seguridad y así lograr una pronta recuperación en caso de fallas.

CAPÍTULO 4: IMPLEMENTACION DE LA BASE DE DATOS

4.1. Introducción

En este capítulo se incluyen los elementos principales que se tuvieron en cuenta para implementar los diferentes objetos de la BD. Se contemplan además los elementos de implementación de objetos de BD necesarios para el diseño propuesto en el capítulo 2.

4.2. Creación de los Objetos de la Base de Datos

4.2.1. Secuencias

Una secuencia es un objeto que se utiliza para generar números enteros únicos, apropiada sólo para las tablas que utilizan columnas numéricas sencillas como claves. Cuando se inserta en una aplicación una nueva fila en una tabla, la misma solicita una secuencia para proporcionar el siguiente valor disponible como clave primaria de la nueva fila. Las secuencias se basan en la aritmética bigint, en PostgreSQL el alcance o el número máximo que se puede generar es 9223372036854775807.

Debido a las particularidades del Sistema de Gestión de Información de las Coordinaciones Regionales que contará con una BD montada en la Dirección General de Prevención del Delito y otras montadas por cada una de las CRs de Venezuela es necesario que cada una de estas CRs presenten identificadores de objetos de BD únicos, eliminando así el riesgo de duplicidad de identificadores a la hora de que los datos sean replicados a la Dirección General. Para ello se definió que los identificadores de cada CR iban a ser representados por un número de 64 bits, de ellos los primeros 52 bit servirían para representar dichos identificadores y los 12 bit restantes identificarían la CR. El rango de identificadores para cada una de las CRs se define por la multiplicación siguiente:

Donde:

#CR sería un número entero comenzando por 1 hasta 4096 ( ).

Sentencia de Creación de Secuencias

```
CREATE [TEMPORARY | TEMP] SEQUENCE [esquema.] secuencia
[INCREMENT [BY] entero]
[MINVALUE entero | NO MINVALUE]
[MAXVALUE entero | NO MAXVALUE]
[START [WITH] entero]
[CACHE entero]
[[NO] CYCLE]
```

```
[OWNED BY {table.column | NONE}]
```

Donde:

- ✓ **Increment:** Especifica que el valor se agrega al valor de secuencia actual para crear un nuevo valor. Un valor positivo hará una secuencia ascendente, uno negativo una secuencia descendente. El valor por defecto es 1.

- ✓ **Minvalue:** Determina el valor mínimo que la secuencia puede generar.

- ✓ **Maxvalue:** Determina el valor máximo que la secuencia puede generar.

- ✓ **Start:** Representa el valor por donde se comienza a generar la secuencia.

- ✓ **Cache:** Especifica cuántos números de secuencia deben ser asignados previamente y almacenados en la memoria para un acceso más rápido. El valor mínimo es 1 (sólo un valor se puede generar a la vez), y esta es la opción por defecto.

- ✓ **[[NO] CYCLE]:** La opción *CYCLE* permite que si se alcanza el límite *maxvalue* o *minvalue* por una secuencia ascendente o descendente, el siguiente número generado será el *maxvalue* o *minvalue*, respectivamente. Si se especifica *NO CYCLE*, todas las llamadas a nextval después que la secuencia ha alcanzado su valor máximo devolverán un error.

- ✓ **[OWNED BY {table.column | NONE}]:** La opción *OWNED BY* permite que la secuencia de estar asociado con una columna de la tabla especificada debe tener el mismo propietario y estar en el mismo esquema que la secuencia. *OWNED BY NONE* es el valor predeterminado, especifica que no hay tal asociación.

Un ejemplo de una de las secuencias implementadas para la CR1 sería:

```
CREATE SEQUENCE bien_mueble_id_bien_mueble_seq
INCREMENT 1
MINVALUE 4503599627370496
MAXVALUE 9007199254740992
START 4503599627370520
CACHE 1;
ALTER TABLE bien_mueble_id_bien_mueble_seq OWNER TO postgres;
```

4.2.2. Triggers o Disparadores

Un trigger es un procedimiento que se ejecuta de forma inmediata cuando ocurre un evento, estos eventos pueden ser inserción, actualización o eliminación de datos de una tabla. Un disparador nunca se llama directamente, en cambio cuando una aplicación o usuario intenta insertar, actualizar o eliminar una fila en una tabla, la acción definida en el disparador se ejecuta automáticamente. Los disparadores permiten simplificar la codificación de aplicaciones de acceso a las BD logrando mayor consistencia y control centralizado.

En PostgreSQL para implementar un trigger se debe realizar primero una función que retorne un disparador, dentro de esta se programarán las acciones que se desean ejecutar y posteriormente se creará el disparador, en el cual se especificará que se debe ejecutar la función anteriormente implementada.

Sentencia de Creación de Triggers

```
CREATE TRIGGER <trigger name>
<BEFORE|AFTER>
<INSERT|DELETE|UPDATE>
ON <relation name>
FOR EACH <ROW|STATEMENT>
EXECUTE PROCEDURE <procedure name> (<function args>);
```

Donde:

- ✓ **<BEFORE|AFTER>:** Determina si la función debe ser llamada antes o después del evento.
- ✓ **<INSERT|DELETE|UPDATE>:** Determina en que eventos será llamada la función.
- ✓ **<relation name>:** Determina la tabla afectada por el evento.
- ✓ **FOR EACH:** Determina si el trigger se ejecutará para cada fila afectada o bien antes o después de que la secuencia se haya completado.
- ✓ **<procedure name>:** Es la función llamada.

En la BD para el sistema fue necesario implementar varios disparadores que se encargarán de llevar el control de las versiones de los indicadores y programas creados en el módulo de Administración de Programas. El objetivo principal de estos disparadores es generar un identificador entero que especificará

la versión del programa o identificador creado. Además de encargarse de actualizar las actividades tipo en la nueva versión del indicador que se acaba de insertar por el sistema.

Se implementaron los disparadores *generate_version on indicador_version*, *generate_version on proyecto_version*, y *update_record_act_tipo* en el esquema administracion_programas.

De igual forma con el objetivo de facilitar como determinar el estado de un ciudadano que puede ser iniciado, cerrado o en seguimiento para el Módulo de Atención de Casos, se implementó el disparador *status_ciudadano _on_update on caso* que se encargará de determinar el estado del ciudadano a partir del estado de los casos asociados al mismo.

A continuación un ejemplo de creación de uno de los disparadores que componen la solución de BD del sistema:

Definición de la función:

```
CREATE OR REPLACE FUNCTION
"administracion_programas"."func_trigger_gen_version_proyecto"()
RETURNS trigger AS
$body$
DECLARE
 var_proyecto_version INTEGER;
BEGIN
 NEW.id_proyecto_version:= 0;

 SELECT max(proyecto_version.id_proyecto_version)
 INTO var_proyecto_version  FROM proyecto_version
 WHERE proyecto_version.id_proyecto = NEW.id_proyecto;

 IF var_proyecto_version >= 0
 THEN NEW.id_proyecto_version:= var_proyecto_version + 1;
 END IF;

 RETURN NEW;
END;
$body$
LANGUAGE 'plpgsql' VOLATILE CALLED ON NULL INPUT SECURITY INVOKER;
```

Definición del trigger:

```
CREATE TRIGGER "generate_version" BEFORE INSERT
ON "public"."proyecto_version" FOR EACH ROW
EXECUTE PROCEDURE "administracion_programas"."func_trigger_gen_version_proyecto"();
```

4.2.3. Vistas

Las vistas se pueden ver como una tabla virtual debido a que no tienen datos almacenados propios, distinguibles y físicamente almacenados, en su lugar el sistema almacena la definición de la vista (las reglas para acceder a las tablas base físicamente almacenadas para materializar la vista) en algún lugar de los catálogos del sistema. Las vistas se utilizan para simplificar la visión del usuario sobre un conjunto de tablas, haciendo transparente para él la forma de obtención de los datos.

Sentencia de Creación de Vistas

```
CREATE VIEW view_name
AS select_stmt
```

Donde:

select_stmt: Es una instrucción *select* válida que no se ejecuta cuando se crea la vista, simplemente se almacena en los catálogos del sistema y se ejecuta cada vez que se realiza una consulta contra la vista.

Fue necesario implementar vistas en la BD para el sistema con el objetivo de mejorar el proceso de visualizar los listados por la aplicación, específicamente para los Módulos de Administración de Programas y Actividades.

4.2.4. Índices

Lograr que una consulta trabaje es una cosa, pero obtener una consulta que trabaje lo más rápidamente es otra muy diferente debido a que el orden de ejecución interno de una sentencia puede afectar gravemente a la eficiencia del SGBD, por lo que se hace necesario que este lleve a cabo una optimización antes de su ejecución. La manera que se utilizará para optimizar la BD y así garantizar un mejor tiempo de respuesta en la BD del sistema es haciendo uso de los índices.

Un índice se crea sobre una o varias columnas de una misma tabla, de esta manera, cuando se solicita recuperar datos de ella mediante alguna condición de búsqueda (cláusula where de la sentencia), ésta se puede acelerar si se dispone de algún índice sobre las columnas objetivo de una manera más rápida y eficiente.

El uso de los índices ayudará a que las consultas, las actualizaciones sean más rápidas, lo que contribuye a lograr mejoras notables y optimizar eficazmente la búsqueda de datos al ser usados para encontrar rápidamente los registros que tengan un determinado valor en alguna de sus columnas.

Rendimiento de los índices: (25)

Se trata de agilizar el acceso a los datos:

- ✓ **En las búsquedas:** Cuando en la condición de la instrucción interviene alguna columna que es utilizada en algún índice, el sistema agiliza la búsqueda de la información cambiando a rastreo indexado sobre este índice.
- ✓ **En las ordenaciones:** Cuando la instrucción exige ordenar la información, el sistema comprueba si existe algún índice que le permite devolver la información ya ordenada.

Sentencia de Creación de Índices

```
CREATE [UNIQUE] INDEX nombre_indice
ON tabla
[USING nombre_acceso]
(columna [ nombre_operador] [, ...] )
```

A continuación un ejemplo de la creación de uno de los índices que componen la solución de BD del sistema:

```
CREATE INDEX "Ref12402" ON "public"."actividad"
USING btree ("id_parroquia");
```

4.3. Implementación de Consultas a la Base de Datos

En las BD el método para acceder a los datos es mediante las consultas, de esta forma se puede modificar, borrar, mostrar y agregar datos utilizando el lenguaje de consultas SQL. Para el Sistema de Gestión de Información de las Coordinaciones Regionales fue necesario hacer un fuerte uso de las consultas utilizando funciones como la Crosstab e implementando una nueva como la función agregada mode para lograr visualizar los informes de cumplimientos mensuales, trimestrales y anuales así como los informes de logros y las estadísticas de los diagnósticos de forma más rápida contribuyendo así al rendimiento.

4.3.1. Función Crosstab

La función crosstab es equivalente en PostgreSQL a las consultas de referencias cruzadas, incluye varias funciones que devuelven varias filas. Toma un parámetro de texto que es una consulta SQL que produce

la fuente de conjunto de datos, esta declaración debe devolver una columna row_name, una columna de categoría, y una columna de valores. Si se tiene la siguiente forma:

```
row_name  cat  value
----------+-------+-------
 row1   cat1  val1
 row1   cat2  val2
 row2   cat1  val5
 row2   cat2  val6
```

La función crosstab se declara para devolver un registro setof, los nombres reales y los tipos de las columnas de salida deben ser definidos en la cláusula FROM de la instrucción de llamada SELECT.

```
SELECT * FROM crosstab ('...') AS ct (row_name text, category_1 text, category_2 text);
```
(26)

En este ejemplo se puede producir lo siguiente:

```
    <== value columns ==>
 row_name  category_1  category_2
----------+------------+------------
 row1    val1     val2
 row2    val5     val6
```

En la práctica la consulta SQL siempre debe especificar ORDER BY 1,2 para asegurar que las filas de entrada están debidamente ordenadas, es decir, los valores con la misma row_name se reúnen y se ordenan correctamente dentro de la fila. (26)

Las consultas implementadas en la BD se realizaron con el propósito de manipular, extraer y presentar la información con los diferentes elementos que se necesitaba mostrar por el sistema, para ello en ocasiones se utilizaron algunas funciones como la función Crosstab o Función de Referencia Cruzada con el objetivo de facilitar la visualización de los datos al permitir resumir las tablas pudiendo mostrar la información de las columnas en filas. Esta función se utilizó fundamentalmente en el módulo de Diagnóstico para visualizar las estadísticas de los diferentes diagnósticos realizados.

4.3.2. Funciones Agregadas

Las funciones agregadas son operaciones que permiten realizar cálculos estadísticos y que se pueden aplicar a las columnas de una relación para obtener una cantidad única, así se puede determinar el número de líneas por tabla o por grupo que cumple una condición, calcular la suma y la media de columnas numéricas o determinar el valor máximo o mínimo de una columna.

4.3.2.1. Las Funciones Básicas de Agregación:

Además de utilizar las funciones básicas de agregación predefinidas como: SUM, COUNT, AVG, MAX, MIN se implementó una nueva función agregada en la BD con el nombre de mode. Esta función se implementó debido a que PostgreSQL no la tiene implementada y era necesario para calcular la moda de los indicadores correspondientes al módulo de Actividades con el objetivo de utilizar estos resultados para mostrar los valores estadísticos que se necesitaban conocer en la generación de los diferentes informes como es el caso del informe trimestral de logros. A continuación en la siguiente figura 15 se muestra como quedaría implementada la función agregada mode.

```
CREATE OR REPLACE FUNCTION "public"."_final_mode" ("pg_catalog"."anyarray")
  RETURNS text AS
$body$
DECLARE
  rec_result record;
  result text;
BEGIN
     result := '';
      FOR rec_result IN
                        SELECT a
                                FROM unnest($1) a
                                GROUP BY 1
                                HAVING count(1) =
                                        (SELECT max(query.cant) FROM (SELECT b,
                                         count(b) as cant FROM unnest($1) b
                                          GROUP BY 1 HAVING count(1) > 1 ORDER by
                                           1) as query)
                                ORDER BY COUNT(1) DESC,
                                        1

   Loop

      result:= result ||'-'|| rec_result.a::text;

  END Loop;

   RETURN result;

END;
$body$
LANGUAGE 'plpgsql' VOLATILE CALLED ON NULL INPUT SECURITY INVOKER;
```

Figura 15. Creación de la Función Mode.

```
CREATE AGGREGATE "public"."mode" ( BASETYPE = "anyelement",
SFUNC = "array_append", STYPE = "anyarray",
FINALFUNC = "public"."_final_mode", INITCOND = "{}");
```

4.4. Conclusiones

En este capítulo se describieron los objetos implementados necesarios que dan solución al diseño propuesto en el capítulo 2 y que permitirán manipular la información dentro del SGBD. Para ello se implementaron las vistas, secuencias, índices, triggers, consultas y funciones que posibilitarán presentar la información que se necesita mostrar por el sistema.

CONCLUSIONES

Al finalizar el presente trabajo de diploma “Solución de Base de Datos para el Sistema de Gestión de Información de las Coordinaciones Regionales”, se ha dado cumplimiento al objetivo general y los objetivos específicos trazados al inicio del mismo. Se hizo un estudio de los procesos que serán automatizados por el sistema con el objetivo de poder realizar el diseño de la BD del sistema. El desarrollo del trabajo permitió además, adquirir conocimientos acerca de la administración del SGBD PostgreSQL definido en la arquitectura del Sistema de Gestión de Información de las Coordinaciones Regionales para manipular los datos.

De manera general se obtuvieron los siguientes resultados:

- ✓ Se obtuvo el diseño de la BD de los diferentes módulos del sistema que se utilizará para gestionar la información generada en los mismos.
- ✓ Se obtuvo la distribución lógica y física de almacenamiento de la información en la BD.
- ✓ Se implementaron los diferentes objetos de la BD que permitirán manipular la información dentro del SGBD PostgreSQL.
- ✓ Se realizó el esquema de seguridad del servidor de BD basados en los privilegios que tienen los usuarios creados.
- ✓ Se creó un plan de políticas de salvas y restauración del sistema para casos de fallas.

Con el diseño de BD propuesto se solucionaron las problemáticas presentadas por los diferentes módulos del Sistema de Gestión de Información de las Coordinaciones Regionales garantizando el almacenamiento, disponibilidad y seguridad de la información con el gestor de base de datos utilizado para manipular los datos. De esta manera, se le da solución al problema científico planteado en la investigación del presente trabajo.

RECOMENDACIONES

Este trabajo da solución al problema planteado y satisface los objetivos trazados al inicio de la investigación, pero hay elementos que se le pueden incorporar en un futuro para que su eficiencia sea aún mayor y no se vea enmarcado en determinados aspectos. Debido a esto al concluir este trabajo y en aras de darle continuidad al mismo y con el objetivo de obtener una solución más completa de la Base de Datos a continuación se presentan un conjunto de ideas y recomendaciones que se deberían tener en cuenta para una posterior versión de la misma:

- ✓ Realizar un estudio de los procesos de negocio para refinar los diseños de los módulos Actividades, Atención de Casos y Administrar Programas.
- ✓ Realizar un estudio del control de versiones para mejorar y optimizar su funcionalidad, para futuras versiones del sistema.

REFERENCIAS BIBLIOGRÁFICAS

1. Historia De Las Bases De Datos. [En línea] [Citado el: 14 de noviembre de 2010.] http://www.buenastareas.com/ensayos/Historia-De-Las-Bases-De-Datos/861977.html.

2. Base de Datos. [En línea] [Citado el: 14 de noviembre de 2010.] http://es.wikipedia.org/wiki/Base_de_datos.

3. **BURBANO PROAÑO, DIEGO JAVIER.** ANALISIS COMPARATIVO DE BASES DE DATOS DE CODIGO ABIERTO VS CODIGO CERRADO. [En línea] [Citado el: 14 de noviembre de 2010.] http://www.willydev.net/InsiteCreation/v1.0/willycrawler/2008.05.12.articulo.comparacion%20bases%20de%20datos%20open%20y%20propietarias.pdf.

4. **Alberca Manzaneque, Alejandro y Díaz Tendero, Jesús Galvez.** Modelos Avanzados de Bases de Datos. Universidad de Castilla- La Mancha : s.n.

5. Base de Datos. [En línea] [Citado el: 14 de noviembre de 2010.] http://www.monografias.com/trabajos55/base-de-datos/base-de-datos2.shtml.

6. Base de Datos. [En línea] [Citado el: 16 de noviembre de 2010.] http://www.icedehesa.edu.mx/Base%20de%20Datos.pdf.

7. **Gil, Fidel, Albrigo, Javier y Do Rosario, Javier.** *Sistemas de Gestión de Base de Datos SGBD/DBMS.* 2005.

8. **Costal Costa, Dolors.** Introducción al diseño de bases de datos. [En línea] [Citado el: 10 de enero de 2011.] http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02150.pdf.

9. Developer Works. [En línea] [Citado el: 10 de enero de 2011.] http://www.ibm.com/developerworks/ssa/data/library/techarticle/dm-0708chang/index.html?ca=drs-.

10. Patrones de diseño de bases de datos. [En línea] [Citado el: 31 de mayo de 2011.] http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=37165.

11. **Mato García, Rosa María.** *Sistemas de Bases de Datos.*

12. Normalización en ¨Bases de Datos¨. [En línea] [Citado el: 19 de noviembre de 2010.] http://www.scribd.com/doc/16311050/normalizacion.

13. Forma normal (base de datos). [En línea] [Citado el: 19 de noviembre de 2010.] http://es.wikipedia.org/wiki/Forma_normal_%28base_de_datos%29.

14. **Ross, Elieser Bello.** *Sistema de Gestión Policial (SIGEPOL). Diseño de Base de Datos.* **Mosquera González, Prof. Dr. Antonio.** Normalización Avanzada del Modelo Relacional.

[En línea] [Citado el: 19 de noviembre de 2010.] http://gva1.dec.usc.es/~antonio/docencia/2006basdat/teoria/T10_NAMR.pdf.

15. Cuarta forma normal. [En línea] [Citado el: 19 de noviembre de 2010.] http://es.wikipedia.org/wiki/Cuarta_forma_normal.

16. Forma normal de dominio/clave. [En línea] [Citado el: 19 de noviembre de 2010.] http://es.wikipedia.org/wiki/Forma_normal_de_dominio/clave.

17. **Gibert Ginestà, Marc y Pérez Mora, Oscar.** Bases de datos en PostgreSQL. [En línea] [Citado el: 26 de noviembre de 2010.] http://www.scribd.com/doc/54814954/1/Introduccion.

18. **Quiñones, Ernesto.** Introducción a PostgreSQL. [En línea] [Citado el: 26 de noviembre de 2010.] http://postgresql.org.pe/articles/introduccion_a_postgresql.pdf.

19. PostgreSQL. [En línea] [Citado el: 26 de noviembre de 2010.] http://www.iessanvicente.com/colaboraciones/postgreSQL.pdf.

20. Comunidad de PostgreSQL en español. Sobre PostgreSQL. [En línea] [Citado el: 26 de noviembre de 2010.] http://www.postgresql.org.es/sobre_postgresql.

21. Tablespaces. [En línea] [Citado el: 25 de enero de 2011.] http://www.postgresql.org/docs/8.4/static/manage-ag-tablespaces.html.

22. **Bonne Solís, Daymel.** *Migración del diseño y administración de la base de datos del Sistema de Gestión de Emergencias y Seguridad Ciudadana (171).*

23. **Paredes, Jose y Alvarez, Meidy.** Clase de Administración-PostgreSql. [En línea] 2010. [Citado el: 11 de enero de 2011.] http://www.codigolibre.org/index.php?option=com_rokdownloads&view=file&Itemid=126&id=107:dba-postgresql.

24. **Alarcón Medina, José Manuel.** Administración SGBD PostgreSQL. [En línea] Octubre / Noviembre 2006. [Citado el: 11 de febrero de 2011.] http://www.scribd.com/doc/32185176/Manual-de-PostgreSQL-Server.

25. Tablefunc. [En línea] [Citado el: 15 de febrero de 2011.] http://www.postgresql.org/docs/current/static/tablefunc.html.

BIBLIOGRAFÍA

1. Base de Datos. [En línea] [Citado el: 13 de noviembre de 2010.] http://www.basesdedatos.org/.

2. ¿Qué son las bases de datos? [En línea] [Citado el: 13 de noviembre de 2010.] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos.

3. Base de Datos. [En línea] [Citado el: 14 de noviembre de 2010.] http://www.monografias.com/trabajos12/basdat/basdat.shtml.

4. Base de Datos. [En línea] [Citado el: 14 de noviembre de 2010.] http://www.monografias.com/trabajos55/base-de-datos/base-de-datos2.shtml.

5. Modelo entidad-relación. [En línea] [Citado el: 16 de noviembre de 2010.] http://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n.

6. **BURBANO PROAÑO, DIEGO JAVIER.** ANALISIS COMPARATIVO DE BASES DE DATOS DE CODIGO ABIERTO VS CODIGO CERRADO. [En línea] [Citado el: 14 de noviembre de 2010.] http://www.willydev.net/InsiteCreation/v1.0/willycrawler/2008.05.12.articulo.comparacion%20bases%20de%20datos%20open%20y%20propietarias.pdf.

7. Base de Datos. [En línea] [Citado el: 16 de noviembre de 2010.] http://www.icedehesa.edu.mx/Base%20de%20Datos.pdf.

8. LOS DBMS(Sistemas Administradores de Bases de Datos). [En línea] [Citado el: 17 de noviembre de 2010.] http://www.unalmed.edu.co/~mstabare/Dbms.htm.

9. Alberca Manzaneque, Alejandro y Díaz Tendero, Jesús Galvez. Modelos Avanzados de Bases de Datos. Universidad de Castilla- La Mancha : s.n

10. Carlín Salgado, Silvia Eloisa y Moreno Rodríguez, Rosendo. VALORIZACIÓN DE LAS BASES DE DATOS DEDUCTIVAS Y DE LAS BASES DE DATOS ACTIVAS.

11. Gil, Fidel, Albrigo, Javier y Do Rosario, Javier. Sistemas de Gestión de Base de Datos SGBD/DBMS. 2005.

12. Mato García, Rosa María. Sistemas de Bases de Datos.

13. Normalización en ¨Bases de Datos¨. [En línea] [Citado el: 19 de noviembre de 2010.] http://es.scribd.com/doc/16311050/normalizacion.

14. Forma normal (base de datos). [En línea] [Citado el: 19 de noviembre de 2010.] http://es.wikipedia.org/wiki/Forma_normal_%28base_de_datos%29.

15. Ross, Elieser Bello. Sistema de Gestión Policial (SIGEPOL). Diseño de Base de Datos.

16. Mosquera González, Prof. Dr. Antonio. Normalización Avanzada del Modelo Relacional. [En línea] [Citado el: 19 de noviembre de 2010.] http://gva1.dec.usc.es/~antonio/docencia/2006basdat/teoria/T10_NAMR.pdf.

17. Cuarta forma normal. [En línea] [Citado el: 19 de noviembre de 2010.] http://es.wikipedia.org/wiki/Cuarta_forma_normal.

18. PostgreSQL. [En línea] [Citado el: 26 de noviembre de 2010.] http://www.iessanvicente.com/colaboraciones/postgreSQL.pdf.

19. Comunidad de PostgreSQL en español. Sobre PostgreSQL. [En línea] [Citado el: 26 de noviembre de 2010.] http://www.postgresql.org.es/sobre_postgresql.

20. Sierra, María. Ingeniería del Software Trabajando con Visual Paradigm for UML. [En línea] [Citado el: 2 de diciembre de 2010.] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

21. Herramientas CASE para el proceso de desarrollo de Software. [En línea] [Citado el: 2 de diciembre de 2010.] http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml.

22. Visual Paradigm for UML. [En línea] [Citado el: 2 de diciembre de 2010.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

23. Costal Costa, Dolors. Introducción al diseño de bases de datos. [En línea] [Citado el: 10 de enero de 2011.] http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02150.pdf.

24. Developer Works. [En línea] [Citado el: 10 de enero de 2011.] http://www.ibm.com/developerworks/ssa/data/library/techarticle/dm-0708chang/index.html?ca=drs-.

25. Tablespaces. [En línea] [Citado el: 25 de enero de 2011.] http://www.postgresql.org/docs/8.3/static/manage-ag-tablespaces.html.

26. Bonne Solís, Daymel. Migración del diseño y administración de la base de datos del Sistema de Gestión de Emergencias y Seguridad Ciudadana (171).

27. Paredes, Jose y Alvarez, Meidy. Clase de Administración-PostgreSql. [En línea] 2010. [Citado el: 11 de enero de 2011.] http://www.codigolibre.org/index.php?option=com_rokdownloads&view=file&Itemid=126&id=107:dba-postgresql.

28. Backup and Restore .[En línea] [Citado el: 5 de febrero de 2011.] http://www.postgresql.org/docs/8.4/static/backup.html.

29. Continuous Archiving and Point-In-Time Recovery (PITR). [En línea] [Citado el: 5 de febrero de 2011.] http://www.postgresql.org/docs/8.4/static/continuous-archiving.html

30. Alarcón Medina, José Manuel. Administración SGBD PostgreSQL. [En línea] Octubre / Noviembre 2006. [Citado el: 11 de febrero de 2011.] http://www.scribd.com/doc/32185176/Manual-de-PostgreSQL-Server.

31. Tablefunc. [En línea] [Citado el: 15 de febrero de 2011.] http://www.postgresql.org/docs/current/static/tablefunc.html.

32. CREATE SCHEMA. [En línea] [Citado el: 20 de enero de 2011.] http://www.postgresql.org/docs/8.4/static/sql-createschema.html.

33. Sequence Manipulation Functions. [En línea] [Citado el: 2 de febrero de 2011.] http://www.postgresql.org/docs/8.4/static/functions-sequence.html.

34. CREATE SEQUENCE. [En línea] [Citado el: 2 de febrero de 2011.] http://www.postgresql.org/docs/8.4/static/sql-createsequence.html.

35. Lockhart, Thomas. Guia del Programador de PostgreSQL. [En línea] [Citado el: 25 de febrero de 2011.] http://www.microalcarria.com/descargas/documentos/Linux/BBDD/PostgreSQL/castellano/programmer.pdf11.

36. GARAVITO, JULIO. MANUAL BÁSICO DE POSTGRESQL. [En línea] [Citado el: 25 de febrero de 2011.] http://laboratorio.is.escuelaing.edu.co/labinfo/doc/Manual_Basico_de_PostgreSQL.pdf.

37. Patrones de diseño de bases de datos [En línea] [Citado el: 31 de mayo de 2011.] http://eva.uci.cu/mod/resource/view.php?inpopup=true&id=37165

GLOSARIO

DGPD: Dirección General de Prevención del Delito.

MIPPRIJ: Ministerio del Poder Popular para Relaciones Interiores y Justicia.

CRs: Coordinaciones Regionales.

CR: Coordinación Regional: Es una oficina adscrita a la Dirección General de Prevención del Delito, conformada por un equipo de profesionales y técnicos que se encargan de la atención e implementación de los programas de prevención del delito.

CCs: Consejos Comunales.

CC: Consejo Comunal: En el marco constitucional de la democracia participativa y protagónica, son instancias de participación, articulación e integración entre las diversas organizaciones comunitarias, grupos sociales y los ciudadanos y ciudadanas, que permiten al pueblo organizado ejercer directamente la gestión de las políticas públicas y proyectos orientados a responder a las necesidades y aspiraciones de las comunidades en la construcción de una sociedad de equidad y justicia social.

BD: Base de Datos.

SGBD: Sistema Gestor de Bases de Datos.

SGBDR: Sistema Gestor de Base de Datos Relacional.

SO: Sistema Operativo.

UML: Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

CASE: Computer Aided Software Engineering o Ingeniería de Software Asistida por Computadora.

SQL: Structured Query Language o Lenguaje Estructurado de Consultas.

Campo: Unidad menor de información sobre un objeto almacenada en la BD que representa una propiedad de un objeto.

Tupla: Ocurrencia de una colección identificable de campos asociados que representan un objeto con sus propiedades.

API: Interfaz de Programación de Aplicaciones: Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Licencia GPL: Licencia Pública General (inglés: General Public License o GPL) otorga al usuario la libertad de compartir el software licenciado bajo ella y realizar cambios en él. Fue creada para mantener la

libertad del software y evitar que alguien quisiera apropiarse de la autoría intelectual de un determinado programa.

Licencia BSD: La Licencia de Distribución de Software de Berkeley (inglés: Berkeley Software Distribution o BSD) no impone ninguna restricción a los desarrolladores de software en lo referente a la utilización posterior del código en derivados y licencias de estos programas, permitiendo a los programadores utilizar, modificar y distribuir a terceros el código fuente y el código binario del programa de software original con o sin modificaciones.

DDL: El lenguaje de definición de datos está orientado a la definición, descripción y mantenimiento de la estructura de la base de datos.

DML: El lenguaje de manipulación de datos y el lenguaje de consulta sirve para obtener, insertar, eliminar y modificar los datos de la base de datos.

Clúster: Es una colección de bases de datos que están administradas por una sola instancia del servidor. Un clúster de base de datos consiste en crear los directorios donde se almacenarán las bases de datos, la generación de las tablas compartidas del catálogo y crear las bases de datos template1 y postgres.

ETP: Equipo Técnico Profesional.

DPCN: Dirección de Programas y Coordinaciones Nacionales.

RH: Recursos Humanos.

RDBMS: Es un Sistema Gestor de Bases de Datos Relacionales.