Universidad de las Ciencias Informáticas
Facultad 1

TRABAJO DE DIPLOMA PARA OPTAR POR EL
TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Diseño de las bases de datos para el proyecto Control de Acceso e Identificación de la UCI.

AUTOR: Wilber Abreu Gómez

TUTOR: Lic. Mónica del Carmen Muñoz

Ciudad de La Habana, 16 de junio del 2008
"Año 50 de la Revolución"

A mis padres, por su apoyo y confianza.

A mi hermano que siempre ha estado presente.

A mi novia Liset por su amor y comprensión.

A mi familia.

A mis padres.

A mi novia Liset por apoyarme y estar a mi lado en todo momento.

A mis amigos Adrian, Victor, Leonel, Roberto por compartir estos 5 años de estudio.

A mis compañeros de estudio.

A los que de una forma u otra aportaron su granito de arena en la realización de este trabajo.

A nuestro comandante en jefe Fidel Castro Ruz y a las Fuerzas Armadas Revolucionarias, forjadores de esta grandiosa Revolución cubana.

Wilber

Declaración de autoría.

Declaro que soy el único autor de este trabajo y autorizo a la Dirección de Informatización de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 15 días del mes de junio del año 2008.

_____ _____

Wilber Abreu Gómez Lic. Mónica del Carmen Muñoz.

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Diseño de las bases de datos para el proyecto Control de Acceso e Identificación de la UCI.

Autor: Wilber Abreu Gómez.

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero en Ciencias Informáticas; y propongo que se le otorgue al Trabajo de Diploma la calificación de ___.puntos.

___ Días del mes de __Junio___ del año 2008.

_____

Lic. Mónica del Carmen Muñoz.

Resumen.

La Universidad de las Ciencias Informáticas (UCI) cuenta con miles de personas y cuantiosos recursos materiales, donde el área de Seguridad y Protección unido al proyecto de Control de Acceso e Identificación juegan un papel muy importante en el cuidado de los activos de la misma.

En la actualidad existen algunos sistemas realizados por este proyecto, pero no cumplen con las expectativas de la UCI, por ejemplo, el Sistema de Control de Acceso de Personas y Vehículos.

La gestión de las fotos de las personas de la UCI es un proceso engorroso, brindando en ocasiones informaciones erróneas y es necesaria la implementación de un servicio Web que solucione este problema. Dicho servicio sustituirá además el Sistema de Gestión de Fotos que actualmente se encuentra en uso en las Oficinas de Seguridad y Protección.

Surge también la necesidad de crear un portal Web para la comunidad universitaria que permita la gestión de las informaciones y reportes referente a la seguridad en la UCI, entre otras funcionalidades.

Para dar cumplimiento a estas tareas se diseñarán las bases de datos necesarias que permitan el almacenamiento y la gestión de las informaciones de los sistemas antes mencionados.

Con los nuevos diseños de las bases de datos del Sistema de Control de Acceso de Personas, Vehículos y Visitantes (SCAPV), además del Portal de Seguridad y Protección y del Sistema de Gestión de Fotos se obtendrán nuevas facilidades para los especialistas del área Seguridad y Protección de la UCI, así como para las personas de la UCI en general.

Permitirán el control de algunos activos que antes no se controlaban. Este trabajo describe los pasos y las herramientas empleadas para la confección de los diseños de bases de datos de estos sistemas.

Palabras claves:
Bases de datos: Colección de datos interrelacionados.
Activos: Personas, medios, vehículos.

Índice.

Índice de figuras.

Índice de tablas.

Introducción.

El control de acceso es una de las principales formas de garantizar la seguridad de las instituciones a nivel mundial, para ello las empresas destinan cuantiosos recursos, entre ellos, personal de seguridad y otros medios.

La Universidad de las Ciencias Informáticas (UCI), es un centro de grandes dimensiones que cuenta con gran cantidad de recursos materiales y humanos. El control de estos activos es una tarea difícil, por esta razón existe un Sistema de Seguridad y Protección bien organizado, que no está ajeno al proceso de informatización del país y de la UCI.

La UCI cuenta con el proyecto productivo Control de Acceso e Identificación que tiene como objetivo automatizar todos los procesos de la Seguridad y Protección existentes en la Universidad. Dentro de los procesos a informatizar se tiene el Sistema de Control de Acceso de Personas, Vehículos y Visitantes (SCAPV), que aunque ya cuenta con una versión, esta tiene deficiencias y no cumple con los nuevos requerimientos entre ellos el Control de Acceso de Vehículos, Control de Acceso de Visitantes y el Control de Medios.

El Control de Acceso de Vehículos consiste en comprobar y registrar los accesos de entrada y salida de los vehículos en general, ya sea de un visitante o de la UCI. Ese registro incluye la hora y lugar por donde accedió entre otras funcionalidades.

El Control de Acceso de Visitantes es una tarea de suma importancia y que facilitará el proceso engorroso que se hacía hasta el momento. Los estudiantes podrán pedir autorización a los dirigentes correspondientes para se visitados, esta autorización se registra y en el momento de la visita por los diferentes puntos de acceso, sólo se comprueban los datos del visitante con los ingresados en el sistema.

El Control de Medios aunque no forma parte del nombre del sistema no deja de ser importante. En ocasiones es necesario sacar algún medio personal que ha entrado con anterioridad a la Universidad y este proceso se torna muy difícil en la toma de decisiones por

parte del personal de la seguridad, esto se debe a que no se conoce el origen del medio para comprobar que no está siendo extraído de forma indebida. Es necesario un sistema que permita conocer los medios que están autorizados a salir de la Universidad, así como su propietario.

La gestión de la información referente a las personas de la UCI es un proceso muy importante debido a la repercusión que tiene sobre todos los sistemas que funcionan con la misma. Actualmente se buscan alternativas para modificar o migrar los sistemas encargados de proveer dichas informaciones, debido a que no son multiplataforma y son funcionales únicamente en el Sistema Operativo Windows, entre estos se encuentra el Sistema de Gestión de Fotos, que posee varias versiones pero no cumplen con las expectativas de la UCI. Sus principales deficiencias son:

- No se ajustan a las necesidades del proceso que se realiza en la UCI.
- No es un sistema multiplataforma.
- Es una aplicación de escritorio, se requiere que sea mediante la Web, utilizando servicios Web y no mediante recursos compartidos.
- Las fotos no poseen incluida en ellas información del propietario de la misma, en caso de que ocurra una contingencia que altere el orden de los datos.

Esta última deficiencia es una de las más importantes debido a que en ocasiones la gestión de las fotos no ha sido confiable. Como consecuencia se debe realizar un nuevo sistema de gestión de fotos para la UCI y con ello un diseño de base de datos que satisfaga los nuevos requerimientos del mismo.

Otra tarea importante que tiene el Proyecto Control de Acceso e Identificación es la integración de todos los procesos automatizados en un Portal de Seguridad y Protección, donde la comunidad universitaria conozca las medidas e informaciones referentes a la Seguridad y Protección orientadas por este órgano. Además es necesario que los directivos de la UCI puedan obtener de una manera más fácil los reportes de accesos en determinado

momento. La Dirección de Informatización determinó la necesidad de crear un portal Web capaz de darle solución a estas y otras situaciones.

La decisión de diseñar y crear nuevas bases de datos surge por un análisis de las existentes en el Sistema de Control de Acceso de Personas y Vehículos, y la necesidad de otros sistemas en desarrollo que no poseen una base de datos donde guardar la información de una manera persistente, es el caso del nuevo Sistema de Gestión de Fotos y del Portal de Seguridad y Protección.

Lo expuesto anteriormente deja claro que no existe un correcto diseño de las bases de datos que satisfaga los requerimientos del Proyecto Control de Acceso e Identificación.

Para solucionar la situación relacionada con las bases de datos del Proyecto Control de Acceso e identificación, se determinó darle solución al siguiente **problema científico:**

- La inexistencia y deficiencias en las bases de datos del Proyecto Control de Acceso e Identificación de la UCI, que permitan el almacenamiento y la gestión de la información.

Para darle cumplimiento al problema científico antes mencionado, durante todo el trabajo se tratará de darle solución a las siguientes **preguntas científicas**.

- ¿Cómo lograr un almacenamiento de los datos del Sistema de Gestión de Fotos y del Portal de Seguridad y Protección?
- ¿Cómo erradicar las deficiencias en las base de datos del SCAPV?

Por tanto, el **objeto de estudio** es el Control de Acceso e Identificación de la UCI.

El **campo de acción** queda enmarcado específicamente en las bases de datos del proyecto Control de Acceso e Identificación de la UCI.

El **objetivo general** diseñar las bases de datos del proyecto Control de Acceso e Identificación.

Lo expuesto anteriormente permite plantear la siguiente **idea a defender:**

El diseño obtenido de las bases de datos, teniendo en cuanta que no deben tener las mismas deficiencias, permitirán el almacenamiento y gestión de la información de manera persistente.

Para darle cumplimiento al objetivo general se plantean lo siguientes **objetivos específicos:**

- Interpretar la arquitectura propuesta.
- Diseñar los Diagramas Entidad-Relación de las bases de datos.
- Validar los diseños realizados.
- Analizar la seguridad de las bases de datos.
- Probar las bases de datos con consultas y herramientas de carga intensiva.

Para cumplir con los objetivos trazados, se desarrollaron las siguientes **tareas de investigación:**

- Realizar un estudio a nivel mundial sobre otros sistemas similares y ver las principales tendencias en el uso de gestores de base de datos (software libre).
- Realizar un estudio de la base de datos con que trabaja actualmente el Sistema de Control de Acceso de Personas y Vehículos.
- Realizar un estudio sobre las tendencias en arquitectura de bases de datos para portales Web y para sistemas de gestión de fotos.
- Estudiar los requisitos funcionales y no funcionales.
- Estudiar los diagramas de clases persistentes obtenido a partir de los diagramas de clases del diseño.

Métodos científicos de investigación.

Métodos Teóricos:

- **Histórico lógico**: Se utiliza este método investigativo al estudiar como ha evolucionado y se han desarrollado las bases de datos desde su surgimiento hasta nuestros días, sus herramientas, formas de trabajos, modelos, etc.

- **Modelación**: Este método se usa para modelar las bases de datos para una mayor comprensión del trabajo que se realiza y los objetivos que se deben cumplir, dando respuesta a los requerimientos planteados con anterioridad.

Métodos Empíricos:

- **Entrevistas**: Se realizan en este trabajo con el objetivo de precisar el problema a resolver, teniendo en cuenta los requisitos que deben satisfacer las bases de datos a diseñar.
- **Observación científica:** En este trabajo se utiliza este método, específicamente el tipo externa incluida, debido a que hubo que estudiar el proceso de Control de Acceso e Identificación de la UCI, esta observación se realizo de manera consiente y orientada al cumplimiento del objetivo del trabajo.

Estructuración del contenido.

El contenido de este trabajo fue organizado por capítulos, los cuales a su vez están divididos por temáticas. El documento cuenta con tres capítulos de los cuales se describe de forma general su contenido:

Capítulo I

En este capítulo se realiza el estudio del estado del arte referente a las bases de datos que usan los sistemas de Control de Acceso. Se enuncian tecnologías, metodologías y herramientas a utilizar en el desarrollo del trabajo y se presentan características que posibilitaron su selección.

Capítulo II

En este capítulo se analiza el negocio sobre el cual se desarrolla el sistema y se realiza una descripción de los procesos dentro del mismo. Se hace referencia a la información sobre la cual se basa el desarrollo de la aplicación. Se recogen los requisitos tanto funcionales como no funcionales; se determina en diagrama de clases persistentes, se realiza el modelo de

datos o Diagrama de Entidad Relación (DER), que incluye también la descripción de sus tablas.

Capítulo III

En este capítulo se realiza la validación teórica y funcional del diseño de las bases de datos. Se analizan aspectos de integridad, seguridad, normalización de las bases de datos, redundancia de la información y trazabilidad de las acciones.

Capítulo 1

CAPÍTULO 1: Fundamentación Teórica.

1.1 Introducción.

En este capítulo se presenta una reseña del tema de las bases de datos en la actualidad, se realiza un estudio del estado del arte. Se especifican las herramientas que se usarán en el diseño de las bases de datos basadas en las tendencias, técnicas, tecnologías y metodologías usadas en el mundo. Para realizar esta selección se tendrá en cuenta la nueva arquitectura propuesta por la Dirección de Informatización de la UCI y las ventajas de la selección.

1.2 Introducción a las bases de datos.

El almacenamiento de datos ha evolucionado, desde los sistemas de archivos secuenciales hasta los sistemas de bases de datos que hoy se conocen.

En los archivos secuenciales para acceder a una posición se necesitaba recorrer el archivo entero. Como solución a este problema aparecieron los archivos indexados, donde el acceso podía ser aleatorio y a una posición deseada. Dada la gran complejidad que iban alcanzando los programas y el volumen de datos que generaban, se requería de un almacenamiento que garantizara un cierto número de condiciones y que permitiera operaciones complejas sin que se violaran estas restricciones. Además cada usuario que accediera a los datos debía tener su trabajo protegido de las operaciones que hicieran el resto de usuarios.

Así surgen las bases de datos, que constituyen una colección de datos interrelacionados que se puede utilizar por uno o más programas de aplicación, los cuales pueden variar en el tiempo.

Componentes de una Base de Datos:

- Hardware: Se refiere a los dispositivos de almacenamiento en donde reside la Base de Datos, así como a los dispositivos periféricos (unidad de control, canales de comunicación, etc.) necesarios para su uso.
- Software: Está constituido por un conjunto de programas que se conoce como Sistema Gestor de Base de Datos, el cual maneja todas las solicitudes formuladas por los usuarios de la misma.
- Usuarios: Personas que interactúan con la base de datos.

Existen tres clases de usuarios relacionados con una base de datos:

1. El programador de aplicaciones es el encargado de crear programas de aplicación que utilizan la base de datos.
2. El usuario final, quien accede a la base de datos por medio de un lenguaje de consulta o de programas de aplicación.
3. El administrador de la base de datos, quien se encarga del control general del Sistema Gestor de Base de Datos (SGBD).

1.3 Sistema Gestor de Base de Datos (SGBD).

Los sistemas de Gestión de Bases de Datos, son aplicaciones que permiten a los usuarios definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma. Los SGBD es la aplicación que interactúa con los usuarios de los programas de aplicación y la base de datos. [1]

Algunos de los SGBD más conocidos son: SQL, PostgreSQL, DB2, SLQ/DS, ORACLE, INGRES, INFORMIX, SYBASE, PARADOX, DBASE, ACCESS, FOXPRO, R, RM/T y RM/V2.

Entre los principales SGBD propietarios más potentes se encuentra Oracle y Microsoft SQL Server, de software libre tenemos opciones muy completas como MySQL y PostgreSQL, que

sobre este último se centrará este estudio; debido a las nueva arquitectura propuesta por la Dirección de Informatización insertada en la estrategia del país de migrar a software libre.

Objetivos de un SGBD.[1]

- Definir la base de datos mediante el lenguaje de definición de satos, el cual permite especificar la estructura, tipo de datos y las restricciones sobre los datos, almacenándolo todo en la base de datos.
- Separar la descripción y manipulación de la data, permitiendo un mayor entendimiento de los objetos, además de flexibilidad de consulta y actualización de los datos.
- Permitir la inserción, eliminación, actualización, consulta de los datos mediante el Lenguaje de Manejo de Datos, lo que permite resolver el problema que presentan los sistemas de archivos, donde hay que trabajar con un conjunto fijo de consultas o la necesidad de tener muchos programas de aplicaciones.
- Existen dos tipos de programas de manejo de datos, los cuales se diferencian por la forma en que acceden a los datos.
  - ✓ Lenguajes procedurales: manipulan la base de datos registro a registro y se deben especificar las operaciones a realizar para obtener los datos resultados.
  - ✓ Lenguajes no procedurales: manipulan la base de datos en conjuntos de registros y se especifican qué datos deben obtenerse como resultado sin plantear las forma de hacerlo. El lenguaje no procedural más utilizado es SQL (Structure Query Languaje) que se ha convertido en un estándar y el lenguaje por defecto de los SGBD relacionales.
- Proporcionar acceso controlado a la base de datos.
  - ✓ Seguridad: los usuarios no autorizados no pueden acceder a la base de datos.
  - ✓ Integridad: mantiene la integridad y consistencia de la base de datos.
  - ✓ Control de Recurrencia: permite el acceso compartido a la base de datos.
  - ✓ Control de Recuperación: restablece la base de datos después de producirse un fallo de software o hardware.
  - ✓ Diccionario de datos o Catálogo: contiene la descripción de los datos de la base de datos y es accesible por el usuario.

- Gestionar la estructura física de los datos y su almacenamiento, proporcionando eficiencia en las operaciones de la base de datos y el acceso al medio de almacenamiento.
- Proporcionar un mecanismo de vistas, que permita a cada usuario tener su propia vista o visión de la base de datos. El lenguaje de definición nos permite definir las vistas como subconjuntos de la base de datos, permitiendo:
  - ✓ Proporcionar un nivel de seguridad excluyendo datos para que no sean vistos por determinados usuarios.
  - ✓ Permiten que los usuarios vean los datos en el formato deseado.
  - ✓ Una vista representa una imagen consistente y permanente de la base de datos, aún cuando a la base de datos se le hagan cambios en su estructura.
- Eliminar la redundancia de datos, establecer una mínima duplicidad en los datos y minimizar el espacio en disco utilizado.
- Proveer interfaces procedimentales y no procedimentales, permitiendo la manipulación por usuarios interactivos y programadores.
- Independizar la estructura de la organización lógica de los datos (Independencia física).
- Independizar la descripción lógica de la Base de datos y las descripciones particulares de los diferentes puntos de vistas de los usuarios.
- Permitir una fácil administración de los datos.

1.3.1 Arquitectura de los Sistemas Gestores de Bases de Datos.

Existen tres características importantes inherentes a los sistemas de bases de datos: la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios y el uso de un catálogo para almacenar el esquema de la base de datos. En 1975, el comité ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los sistemas de bases de datos, que resulta muy útil a la hora de conseguir estas tres características.

El objetivo de la arquitectura de tres niveles es separar los programas de aplicación de la base de datos física. En esta arquitectura, el esquema de una base de datos se define en tres niveles de abstracción distintos:

- En el *nivel interno* se describe la estructura física de la base de datos mediante un esquema interno. Este esquema se especifica mediante un modelo físico y describe todos los detalles para el almacenamiento de la base de datos, así como los métodos de acceso.
- En el *nivel conceptual* se describe la estructura de toda la base de datos para una comunidad de usuarios, mediante un esquema conceptual. Este esquema oculta los detalles de las estructuras de almacenamiento y se concentra en describir entidades, atributos, relaciones, operaciones de los usuarios y restricciones.
- En el *nivel externo* se describen varios esquemas externos o vistas de usuario. Cada esquema externo describe la parte de la base de datos que interesa a un grupo de usuarios determinado, oculta a ese grupo el resto de la base de datos.[1]

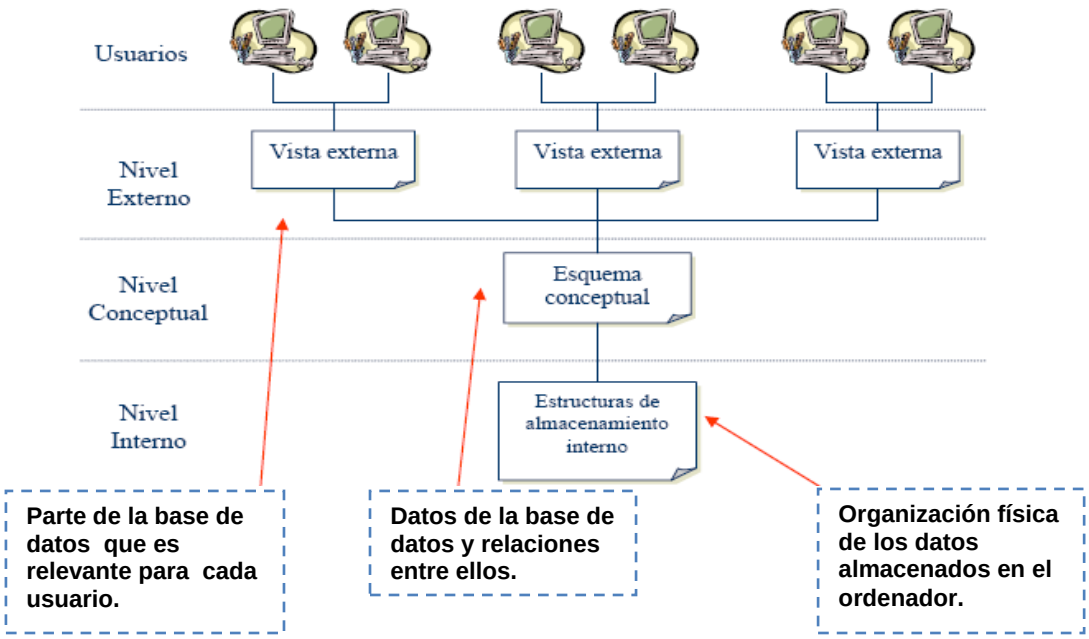

Figura 1. 1 Arquitectura de un sistema gestor de base de datos.

1.3.2 PostgreSQL.

Es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley.

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

Tiene soporte completo para llaves foráneas, joins, vistas, subconsultas (incluyendo subconsultas en la clausula FROM), triggers, y procedimientos almacenados (en varios lenguajes). Incluye la mayoría de los tipos de datos de los estándares SQL92 y SQL99 (**INTEGER**, **NUMERIC**, **BOOLEAN**, **CHAR**, **VARCHAR**, **DATE**, **INTERVAL**, **TIMESTAMP**, entre otros). También soporta almacenamiento de objetos grandes (imágenes, sonido y video), así como sus propias interfaces de programación para C/C++, Java.
Es Open Source, bajo la licencia BSD, es multiplataforma, y su distribución es gratis. Los usuarios pueden mejorar las fallas que estos gestores puedan tener, debido a que se les permite arreglar el código fuente a su conveniencia. Es uno de los pocos gestores de bases de datos que permite la creación de nuevos tipos de datos. Este gestor posee una integridad referencial muy buena permitiendo una mayor seguridad en la base de datos.

Características:

- **DBMS Objeto-Relacional:** Aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son las consultas SQL declarativas, el control de concurrencia multiversión, el soporte multiusuario, las transacciones, optimización de consultas, la herencia, los arreglos entre otros.

- **Cliente/Servidor:** Usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.
- **Altamente extensible:** Soporta los tipos de datos base, así como: fechas, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. Además operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario.
- **Soporte SQL Comprensivo:** Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.
- **Integridad Referencial:** Es utilizada para garantizar la validez de los datos de la base de datos.
- **Lenguajes Procedurales:** Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Además tiene habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.
- **MVCC (Multi-Versión Concurrency Control) Control de Concurrencia multiversión:** Es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios, es decir, permite la lectura sin que sea bloqueada por los usuarios que están actualizando registros.
- **Write Ahead Logging (WAL):** Esta característica 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 caso de que no existan las condiciones para la conexión con la base de datos, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos desde el punto en que se quedó.[2]

1.3.2.1 SQL Manager 2005 for PostgreSQL.

PGManager es una herramienta gráfica fácil de utilizar para la administración del servidor de base de datos PostgreSQL. Permite obtener la documentación sobre el diseño de la base de datos y tiene un grupo de herramientas importantes para la importación y exportación de datos. Posibilita asignar los derechos de usuarios de una forma sencilla y rápida.

1.3.3 SQLite.

Es una pequeña librería programada en lenguaje C que implementa un completo motor de base de datos multiplataforma que no precisa configuración. SQLite se encuentra en el dominio público. Es muy rápido y la ventaja fundamental es que permite utilizar un amplio subconjunto del lenguaje estándar SQL. Se destaca también por su versatilidad. Su potencia se basa fundamentalmente en la simplicidad, lo que a cambio lo hace una solución a descartar en entornos de tráfico muy elevado y/o alto acceso concurrente a datos. Encapsula toda la base de datos en un único fichero. Sus desarrolladores destacan, que su principal característica, es su completo soporte para tablas e índices en un único archivo por base de datos, soporte transaccional, rapidez (unas 2 veces más veloz que MySQL y PostgreSQL, escaso tamaño (unas 25 mil líneas de código C) y su completa portabilidad. [3]

SQLite, no permite múltiples usuarios accediendo en modo escritura a la base de datos, debido que el mecanismo de bloqueo que utiliza es muy "basto": bloquea toda la base de datos. Así esta librería está especialmente indicada cuando se requiera de una gran rapidez en las consultas y nos baste que sólo un único usuario pueda realizar modificaciones.[3]

1.3.4 Db4o.

Se trata de una base de datos que está específicamente diseñada para proporcionar persistencia a los programas orientados a objetos.

Persistencia de objetos es la capacidad de guardar los objetos en un sistema con el fin de que existan incluso después de que la aplicación que los creó deje de usarlos. [4]

Características:[5]

- **Consumo mínimo de recursos**: Db4o está diseñado para ser embebido en clientes u otros componentes de software, de manera totalmente invisible para el usuario final. Es por eso que viene como una librería fácil de incorporar, con un tamaño que ronda los 400 Kb. Como el motor corre en el mismo proceso de la aplicación, el usuario cuenta con control completo sobre la administración de memoria, y puede realizar procesos de profiling y debugging del desempeño sobre todo el sistema. Si la aplicación está corriendo, la base también lo está, sin excepción. Y aún más importante, Db4o es extremadamente flexible a la hora de actualizar una base existente con un modelo de objetos que ha cambiado. Siempre asume que no hay un administrador de base de datos y por lo tanto permite a la aplicación cambiar del modelo viejo al modelo nuevo de modo transparente.
- **Alto rendimiento**: El rendimiento de Db4o es equivalente al de los mejores sistemas de bases de datos tradicionales.
- **Fácil implementación**: Sólo hay que agregar la librería única de Db4o (.jar o .dll) al entorno de desarrollo, abrir el archivo de base de datos y almacenar cualquier objeto (sin importar su complejidad) con una sola línea de código.
- **Portabilidad**: Corre de manera embebida y nativa en plataformas orientadas a objetos. Se pueden desarrollar aplicaciones para desplegar en varias plataformas (por ejemplo, en PDA) o en combinaciones heterogéneas de clientes Windows y servidores Java.
- **Confiabilidad**: Finalmente, Db4o soporta todas las propiedades ACID. Múltiples usuarios simultáneos de una base Db4o son efectivamente aislados, y sus operaciones son serializadas de forma transparente por la librería. Las transacciones se terminan con los métodos commit() y rollback() de la clase ObjectContainer, en caso de que el sistema detenga el servicio durante una actualización de la base de datos, cuando el ObjectContainer de Db4o es reabierto, se completan de forma correcta todas las transacciones interrumpidas.

Este motor de base de datos posee tres modos de configuración: [5]

- Modo embebido.
- Modo servidor.
- Modo servidor embebido.

1.4 Modelos de Bases de Datos.

Un modelo de datos es básicamente una “descripción” de un contenedor de datos donde se guarda la información, así como de los métodos para almacenar y recuperar información de esos contenedores.[6]

1.4.1 Bases de Datos Objeto Relacionales.

Una Base de Datos Objeto Relacional (BDOR) es una base de datos que desde el modelo relacional evoluciona hacia una base de datos más extensa y compleja, incorporando para obtener este fin, conceptos del modelo orientado a objetos. Un Sistema de Gestión Objeto-Relacional (SGBDOR) contiene dos tecnologías: la tecnología relacional y la tecnología de objetos.[6]

El modelo Objeto Relacional trata de resolver desde diferentes ángulos el problema de almacenamiento de objetos en base de datos. Extiende capacidades del modelo relacional para permitir que los objetos sean almacenados en columnas de una base de datos relacional. Este modelo es a menudo referido como un DBMS híbrido.

Con las Bases de Datos Objeto-Relacional, se pueden crear nuevos tipos de datos, que permiten gestionar aplicaciones más complejas con una gran riqueza de dominios. Tienen la posibilidad de incluir el chequeo de las reglas de integridad referencial a través de los disparadores, entre otras características. Uno de los gestores de Bases de Datos Objeto Relacional que más se utiliza en la actualidad es PostgreSQL.

1.4.2 Bases de Datos Orientada a Objetos. (BDOO).

Las bases de datos orientadas a objetos (BDOO), almacenan y recuperan objetos que poseen estado y comportamiento. Las clases utilizadas en un determinado lenguaje de programación orientado a objetos son las mismas clases que serán utilizadas en una BDOO; de tal manera, que no es necesaria una transformación del modelo de objetos para ser utilizado por un SGBDOO. De forma contraria, el modelo relacional requiere abstraerse lo suficiente como para adaptar los objetos del mundo real a tablas.[7]

El modelo de datos orientado a objetos es similar al paradigma orientado a objetos, excepto que los objetos son persistentes, es decir, que ellos continúan existiendo después que la ejecución del programa ha concluido y los objetos en una base de datos de objetos se almacena exactamente como ellos fueron creados en la aplicación.

En una BDOO, las entidades de aplicación son las clases, las instancias de entidad son objetos creados desde las clases, y las relaciones se mantienen por medio de inclusión lógica.

Características principales de las Bases de Datos Orientada a Objetos:[5]

- Las bases de datos orientadas a objetos no sólo permiten transparentemente trabajar en un entorno de programación basado en objetos sino que soportan la tecnología de objetos.
- Agregación: Objetos que están compuestos por otros.
- Encapsulamiento: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia: Los objetos heredan atributos y comportamientos de su objeto padre.
- Polimorfismo: Permite a los objetos responder de forma distinta a un mismo mensaje.
- Integridad de datos:
  - ✓ La integridad estructural: asegura que los contenidos son consistentes con el esquema de la base de datos. La integridad referencial requiere relaciones bidireccionales entre objetos para asegurar que no hay referencias a objetos eliminados.

- ✓ Integridad lógica: asegura que las propiedades lógicas de los datos son correctas. El valor es correcto para los datos de forma consistentes y el acceso concurrente no provoca que se registren valores inválidos.

- ❖ Seguridad: posee encriptación, normalmente diferentes métodos y niveles de autenticación para acceder a la base de acuerdo al producto.
- ❖ Tolerancia a fallos: En el caso de que ocurran problemas con el hardware o software se protege el procesamiento transaccional.
- ❖ Acceso a datos: Normalmente realiza utilizando un iterador para recorrer los datos como si los objetos estuvieran en colecciones o diccionarios. Los objetos son cargados por demanda, y no permanecen en memoria.

En la siguiente figura se ilustra cómo se combinan los conceptos de tecnología de objetos y bases de datos para conformar una base de datos de objetos.

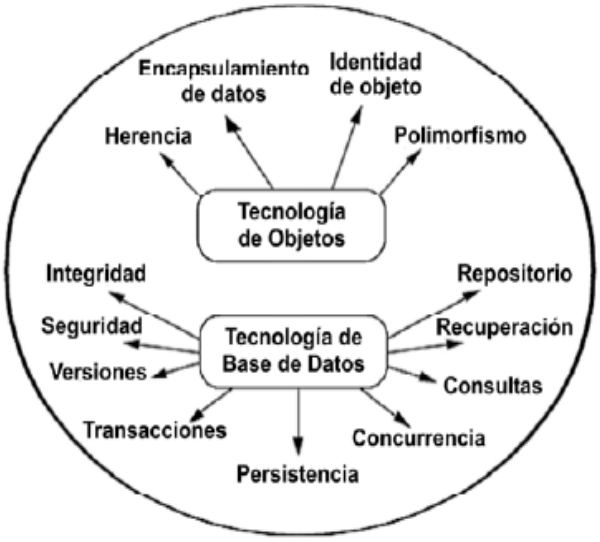

Figura 1. 2 Tecnología de objetos y bases de datos

1.5 Tendencias actuales y futuras.

El uso de las bases de datos ha aumentado notablemente en los últimos años como plataforma para el desarrollo de los sistemas, es por eso que aumentan las demandas de

nuevas operaciones y así surgen nuevas técnicas y herramientas que le den soluciones a las necesidades planteadas por los usuarios.

Algunas de las tendencias actuales y futuras de las bases de datos son:

- La explotación efectiva de la información dará ventaja competitiva a las organizaciones.
- En el futuro la mayoría de las organizaciones cambiarán la forma convencional de manejo de la información al uso de base de datos debido a las ventajas que ofrecen.
- El uso de las bases de datos facilita y soporta en gran medida a los sistemas de información para la toma de decisiones.
- Los lenguajes de consulta (SQL) permitirán el uso del lenguaje natural para solicitar información de la base de datos, haciendo más rápido y fácil su manejo.[8]

1.5.1 Aplicaciones con DMBS embebidos.

Las aplicaciones con DMBS embebido incluyen una base de datos en dispositivos móviles parcialmente conectados a un caché de objetos en una aplicación que requiere tiempos de respuesta muy rápidos. En el caso particular de DB4o es muy utilizado para aplicaciones que interactúan con dispositivos móviles, sistemas de tiempo real.

1.5.2 Bases de datos desarrolladas para sistemas similares de control de acceso.

En la Universidad de las Ciencias Informáticas.

- Sistema Control de Acceso v1: se realizó en el proyecto Control de Acceso e Identificación. En esta versión se usó como SGBD SQL Server 2000 que no cumple con la nueva arquitectura de la DI.
- Sistema de Control de Acceso de personas a los laboratorios (UCILab), que actualmente está funcionando en algunos laboratorios de proyectos de la Universidad de las Ciencias Informáticas. Usa como gestor de base de datos MySQL.

- Sistema Control de Proyecto: es usado para el control de las personas pertenecientes a los proyectos productivos y usa como gestor de base de datos PostgreSQL.
- Sistema de Control de Acceso al Comedor: Este sistema controla el acceso de las personas a los comedores de la UCI. Inicialmente se usó como SGBD SQL Server y recientemente se migró al SGBD PostgreSQL debido a las ventajas que trae el uso del mismo y a la nueva arquitectura planteada por la DI para los sistemas de la intranet universitaria. La base de datos de este sistema se analizó detalladamente debido a la gran similitud que posee este con respecto al SCAPV. El estudio de esta base de datos aportó algunas técnicas a seguir en el nuevo diseño de la base de datos del SCAPV.

1.6 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software.

1.6.1 Rational Unified Process (RUP).

Utilizaremos como metodología de desarrollo de software a RUP (Rational Unified Process) para facilitar el desarrollo del sistema.[9]

El Proceso Unificado es un proceso de desarrollo de software que contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software (Figura 1.3). Más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas, para diferentes áreas de aplicación, tipos de organizaciones, niveles de actitud y tamaños de proyecto. Está basado en componentes, lo cuál quiere decir que el sistema software en construcción está formado por componentes, software interconectado a través de interfaces bien definidas. Utiliza el Lenguaje Unificado de Modelado (Unified Modeling Lenguaje, UML) para preparar todos los esquemas de un sistema software. Garantiza la elaboración de todas las fases de un producto de software orientado a objetos.

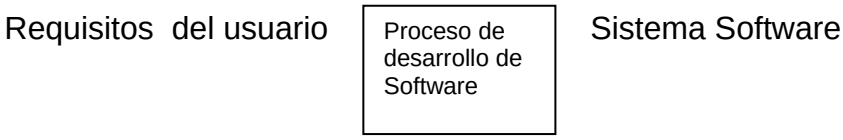

Figura 1. 3 Un proceso de desarrollo de software.

1.6.2 Características del RUP.

- *Dirigido por los Casos de Uso:* Los casos de usos no son sólo una herramienta para especificar los requisitos de un sistema; también guían su diseño, implementación y prueba, es decir el proceso de desarrollo. Estos además guían la arquitectura del sistema y esta a su vez influye en la selección de los casos de uso.
- *Centrado en la arquitectura:* La arquitectura involucra los aspectos estáticos y dinámicos más significativos del sistema. Los casos de uso deben encajar en la arquitectura cuando se llevan a cabo y la arquitectura debe permitir el desarrollo de todos los Casos de Uso requeridos, actualmente y en el futuro. Esto provoca que tanto arquitectura como Casos de Uso deban evolucionar en paralelo durante todo el proceso de desarrollo de software. El Modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas UML.
- *Iterativo e incremental:* Se propone que el proyecto se divida en mini proyectos, en cada uno de estos la arquitectura y los casos de uso van logrando un equilibrio. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Según RUP el Diseñador Principal de Bases de Datos es el responsable de definir los detalles de diseño de la Base de Datos y para ello crea el Modelo de Datos, garantizando su integridad y consistencia. Este artefacto describe las representaciones lógicas y físicas de los datos persistentes usados por la aplicación.

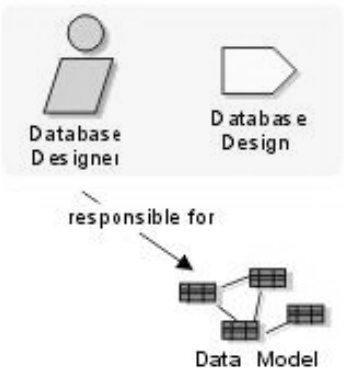

Figura 1. 4 Artefacto construido por el diseñador de la Base de Datos.

1.6.3 UML.

UML (Unified Modeling Lenguaje) es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. Se utiliza para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema y con este fin se utiliza en el trabajo. [10]

1.7 Herramientas CASE.

Las herramientas CASE (Computer Aided Software Engineering) son herramientas de software que automatizan tareas particulares del ciclo de vida del software. Son la base para el proceso de análisis y desarrollo de software. Brindan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (Investigación preliminar, Análisis, Diseño, Implementación e Instalación).[11]

1.7.1 Visual Paradigm.

Los sistemas que se construyen hoy son cada vez más complejos, las herramientas de modelado con UML ofrecen beneficios para todos los involucrados en un proyecto.

Visual Paradigm es una herramienta CASE que utiliza “UML” como lenguaje de modelación y se utilizará para dar soporte al modelado visual de los principales flujos de trabajo en el desarrollo del software según RUP. Posibilita representar sistemas complejos y desarrollar la solución de software correcta más rápida y económicamente, además mejora la comunicación entre los miembros del equipo usando un lenguaje grafico entendible por todos. Incluye un conjunto de herramientas de ingeniería inversa y generación de código que facilitan el camino hasta el producto final.

- Entorno de creación de diagramas para UML 2.0
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Es posible integrarlo a los principales IDEs.
- Es multiplataforma.

1.7.2 DB Visual Architect.

Esta herramienta está especializada en la ingeniería del software de bases de datos. Permite gestionar proyectos muy complejos con gran sencillez. Incluye herramientas muy interesantes para ingeniería inversa de bases de datos. Se puede integrar con Eclipse, Microsoft Visual Studio, Borland JBuilder, NetBeans, IntelliJ IDEA, BEA Weblogic workshop, entre otros.

1.8 Conclusiones.

Actualmente existen diversas herramientas con que se pudiera dar solución al problema planteado. Para la selección de las mismas se tuvo en cuenta fundamentalmente la arquitectura propuesta por la DI de la UCI. Esta arquitectura propone PostgreSQL como gestor de base de datos, aunque se puede sugerir otra variante, propone además SQLite y db4o como base de datos embebida.

Después de un análisis de las características del SGBD PostgreSQL, se concluye con que permite darle solución a parte del problema planteado en este trabajo.

Para las bases de datos embebidas del sistema de Control de Acceso de Personas, Vehículos y Visitantes, que estarán ubicadas localmente en los puntos de acceso de la UCI, se usará db4o, para esta selección se tuvieron en cuenta algunos aspectos:

- Db4o es una base de datos de orientada a objetos, lo que permite mayor rapidez en las transacciones.
- Elimina la incompatibilidad entre los tipos de estructuras de datos que se transfieren, se conoce como **inadaptación de impedancia**.
- Es más cercano a la realidad que un modelo funcional.
- Es más fácil de mantener y reutilizar.

Finalmente se concluye con la selección de las metodologías y herramientas más factibles para darle cumplimiento a los objetivos planteados.

- Se escoge el modelo de base de datos Objeto-Relacional para la base de datos remota del SCAPV, del Portal de Seguridad y Protección y para el Sistema de Gestión de Fotos.
- Se escogió el modelo de base de datos Orientado a Objetos para la base de datos local del SCAPV.

- Se escogió como metodología de desarrollo RUP y como lenguaje de modelado UML para la realización del diagrama de clases persistentes.
- Para el diseño de las bases de datos se seleccionó Visual Paradigm, específicamente la herramienta DB Visual Architect, teniendo en cuenta principalmente que la UCI posee la licencia del Visual Paradigm, y además cumple con los requisitos necesarios para darle cumplimiento a los objetivos del presente trabajo.
- Los SGBD seleccionados:
- PostgreSQL y DB4o.
- Para la administración de las bases de datos en PostgreSQL se escoge EMS PostgreSQL Manager 3 y para la generación de datos de prueba EMS Data Generador for PostgreSQL.

Capítulo 2

CAPÍTULO 2: Descripción y Análisis de la Solución Propuesta.

2.1 Introducción.

En este capítulo se describe la solución propuesta para cada sistema que compone el proyecto Control de Acceso e Identificación, donde se tuvo en cuenta los requisitos funcionales y no funcionales, en este capítulo fue de gran importancia el estudio de los diagramas de clases persistentes que obtuvieron los analistas a partir de los diagramas de clases de diseño. Esto permitió la realización de los diagramas de entidad-relación de las respectivas bases de datos.

2.2 Requisitos funcionales y no funcionales.

Definir las necesidades que posee el sistema es un proceso complejo, y determinar los requisitos es un paso muy importante para satisfacer a los usuarios finales y clientes.

En este proceso se especifican y validan los servicios que el sistema debe proporcionar, así como las restricciones que posee. Esta fase es de gran importancia, sin una buena ingeniería de requisitos no es posible realizar un sistema que cumpla con los requerimientos.

Requisitos funcionales: Los requisitos funcionales representan las funcionalidades del sistema, es decir, son capacidades o condiciones que el sistema debe cumplir.

Requisitos no funcionales: Los requisitos no funcionales representan aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica, es decir, son propiedades o cualidades que el producto debe tener, estas se dividen por categorías, por ejemplo: facilidad de uso, fiabilidad, eficiencia, portabilidad, hardware, software, seguridad, etcétera.

2.4 Sistema Control de Acceso de personas, vehículos y visitantes.

2.4.1 Requisitos funcionales.

- **RF-1 Autenticar usuario.**
  RF-1.1 Verificar Usuario y contraseña.

- **RF-2 Gestionar acceso puntual de personas de la UCI.**
  RF-2.1 Verificar la validez de las credenciales de personas de la UCI.
  RF-2.2 Registrar las entradas de personas de la UCI.
  RF-2.3 Registrar las salidas de personas de la UCI.
  RF-2.4 Eliminar las entradas de personas.
  RF-2.5 Eliminar las salidas de personas.

- **RF-3 Gestionar accesos de vehículos.**
  RF-3.1 Verificar la validez de los vehículos de forma puntual.
  RF-3.2 Registrar las entradas de los vehículos.
  RF-3.3 Registrar las salidas de los vehículos.
  RF-3.4 Eliminar las entradas de vehículos.
  RF-3.5 Eliminar las salidas de vehículos.

- **RF-4 Gestionar accesos de visitantes.**
  RF-4.1 Registrar las entradas de visitantes.
  RF-4.2 Registrar las salidas de visitantes.
  RF-4.3 Eliminar las entradas de visitantes.
  RF-4.4 Eliminar las salidas de visitantes.

- **RF-5 Verificar PassBack.**
  RF- 5.1 Registrar motivo de PassBack.

- **RF-6 Mostrar Visitantes Autorizados.**

- **RF-7 Registrar Medios.**
- **RF-8 Mostrar Avisos.**
- **RF-9. Configurar el sistema.**
  RF-9.1 Configurar tiempos de sincronización de las BD.
  RF-9.2 Visualizar datos de configuración del sistema.
- **RF-10. Actualizar Lector.**
- **RF-11. Actualizar Base de Datos Local.**

2.4.2 Requisitos no funcionales.

- **RNF-1 Requerimientos de apariencia o interfaz externa:** El sistema debe tener un ambiente amigable y entendible para los usuarios finales, de forma tal que no les sea muy complicado utilizar el software.
- **RNF-2 Requerimientos de usabilidad:** El sistema podrá ser usado por personas autorizadas, la navegabilidad no debe ser muy compleja, todas las funcionalidades deben ser rápidamente accesibles por el usuario.
- **RNF-3 Requerimientos de rendimiento:** El tiempo de respuesta de una petición al servidor deber ser rápido ya que el sistema operara con grandes cantidades de datos.
- **RNF-4 Requerimientos de soporte:** Se le debe dar mantenimiento periódicamente a los servidores de bases de datos controlando la integridad de la información.
- **RNF-5 Requerimientos de portabilidad:** Debe tener facilidad para adaptarlo a diferentes ambientes. Independencia de la plataforma.
- **RNF-6 Requerimientos de seguridad y privacidad:**
  - ✓ Identificar al usuario antes de que pueda realizar cualquier acción sobre la configuración del sistema.
  - ✓ Garantizar que la información sea vista sólo por personas autorizadas.

- ✓ Verificación sobre acciones irreversibles (eliminaciones).
- ✓ Garantía de que el sistema funcione correctamente aun cuando no haya conectividad.

- ❖ **RNF-7 Requerimientos de confiabilidad:** Se debe chequear la integridad de los datos.

- ❖ **RNF-8 Requerimientos Legales:**

Debe cumplir con:
Decreto Ley – 186 / 98.
2da Sección, Artículos 5, 8, 11, Inciso D.
Reglamento del Decreto Ley 186 (Resolución No. 2 / 2001 del Ministerio del Interior).
Artículo 1. Capítulo IV (Artículos 3, 4, 90).
El Plan de Seguridad y Protección del Centro.

- ❖ **RNF-9 Requerimientos de Ayuda y documentación en línea:** Se debe brindar un sistema de ayuda que explique las diferentes funcionalidades con que cuenta el sistema, además los manuales de usuario.

- ❖ **RNF-10 Restricciones en el diseño y la implementación:** Para el análisis y el diseño del sistema debe ser utilizada la metodología RUP, usando el lenguaje de modelado UML y como herramienta para llevarlo a cabo el Visual Paradigm.

- ❖ **RNF-11 Requerimientos de software:** Se debe disponer en el servidor con Windows XP, Windows 2000 Server o 2000 Advanced Server. Se utilizará como lenguaje de programación: Java y como gestor de Base de Datos: PostgreSQL.

- ❖ **RNF-12 Requerimientos de hardware:** Requiere estar instalada en una PC Pentium, 256 Mb de RAM (como mínimo), Micro 2.0 o superior, 280 Mb de disco duro, una tarjeta de red de 100Mbps y el procesador de 133 MHz o superior.15:44.

2.5 Portal de Seguridad y Protección.

2.5.1 Requisitos funcionales.

- **RF-1 Autenticar.**

  RF-1.1 Permitir la autenticación de usuarios por el dominio.

- **RF-2 Administrar información.**

  RF-2.1 Publicar una nueva información.

  RF-2.2 Modificar información.

  RF-2.3 Eliminar información.

- **RF-3 Administrar enlaces a otros sitios.**

  RF-3.1 Añadir un nuevo vínculo.

  RF-3.2 Modificar un vínculo.

  RF-3.3 Eliminar un vínculo.

- **RF-4 Mostrar reportes.**

  RF-4.1 Mostrar reporte del parte de acceso de personas.

  RF-4.2 Mostrar reporte del parte de acceso de recursos.

  RF-4.3 Mostrar reporte del parte de acceso a vehículos.

  RF-4.4 Mostrar reporte del parte de las estadísticas de accesos para el Sistema Informativo.

  RF-4.5 Mostrar reporte del historial de credencial.

  RF-4.6 Mostrar reporte del historial de credenciales de un grupo.

- **RF-5 Administrar documentos.**

  RF-5.1 Añadir un documento nuevo.

  RF-5.2 Eliminar un documento.

  RF-5.3 Visualizar un documento.

  RF-5.4 Descargar un documento.

- **RF-6 Administrar Plan de Guardia.**
  RF-6.1 Crear un plan de guardia.
  RF-6.2 Modificar un plan de guardia.
  RF-6.3 Permitir la búsqueda en la planificación de la guardia.

2.5.2 Requisitos no funcionales.

- **RNF-1 Requerimientos de apariencia o interfaz externa.** Interfaz con colores poco llamativos, que permita mostrar el tipo de información que gestiona el sistema y el área de la UCI con la que se relaciona (Oficina de Seguridad y Protección). Interfaz con un diseño sencillo, que permita acelerar la velocidad de respuesta hacia el usuario debido a la gran cantidad de información a gestionar. Interfaz debe mostrar información relacionada solamente con las funcionalidades del sistema para no facilitar distracción en el trabajo que se desarrolle.

- **RNF-2 Requerimientos de usabilidad**: Cada usuario del sistema tendrá acceso según su rol. Documentar bien la aplicación y proporcionar materiales de ayuda para hacer mejor uso de los servicios que este ofrece.

- **RNF-3 Requerimientos de rendimiento:** Estará implementado sobre una tecnología Web, facilitando su uso a través de la red. El sistema deberá ser capaz de gestionar toda la información y dar respuesta en el menor tiempo posible.

- **RNF-4 Requerimientos de seguridad:** Únicamente tendrán acceso al sistema aquellos usuarios que tengan permiso establecidos por el Especialista de Seguridad y Protección. La autenticación se realizará sobre el dominio UCI.

Chequeo de seguridad sobre las acciones tales como verificación de borrado, de modificado, etc.
Se establecerán los servidores tanto de aplicación como de base de datos en lugares de restringido acceso.

- **RNF-5 Requerimientos de hardware:** Las PC de los clientes deberán tener 40 GB o superior, microprocesador superior a 1.0 GHz de velocidad, 256 MB mínimo de RAM.

- **RNF-6 Requerimientos de software:** Sistema operativo Windows XP o superior, PostgreSQL, Apache. Gestor de base de datos PostgreSQL

- **RNF-7 Restricciones en el diseño e implementación**: Para organizar el análisis y el diseño del sistema, debe ser utilizada la metodología RUP, usando el lenguaje de modelación UML y como herramienta para llevarlo a cabo el Visual Paradigm. Para separar la interfaz del negocio y el acceso a los datos debe utilizarse la arquitectura de tres capas.

2.6 Sistema de Gestión de Fotos.

2.6.1 Requisitos funcionales.

- **RF-1: Verificar que el usuario tenga acceso al servicio solicitado.**

- **RF-2: Verificar exactitud de los datos y fotos recibidas.**
  RF-2.1. Verificar datos de huellas digitales.
  RF-2.2. Verificar datos de recurso.
  RF-2.3. Verificar datos del personal.

- **RF-3: Darle formato a la imagen.**

- **RF-4: Darle resolución a la imagen.**

- **RF-5: Borrar datos EXIF de las fotos.**

- **RF-6: Introducir datos no visibles en las fotos**

- **RF-7: Crear el repositorio**
  RF-7.1: Principal.

RF-7.2: Seguridad.

- **RF-8: Guardar foto en el repositorio atendiendo a su clasificación.**
  RF- 8.1Principal.
  RF-8.2 Seguridad.
- **RF-9: Permitir historial de fotos.**
- **RF-10: Devolver la foto atendiendo a la clasificación del pedido.**
  RF-10.1: Principal.
  RF-10.2: Seguridad.
- **RF-11: Devolver datos no visibles de las fotos.**
- **RF-12: Gestionar Usuarios.**

2.6.2 Requisitos no funcionales.

- **RNF-1 Requisito de Usabilidad:** El tiempo de entrenamiento requerido para que los usuarios sean productivos operando el sistema, es 2 días y para los avanzados es 1 día. El sistema debe ser de fácil manejo para los usuarios que tengan niveles básicos sobre la computación o hayan trabajado con la Web. Debe tener una opción de ayuda sobre las principales operaciones que se realizan y sus iconos respectivos para lograr un menor tiempo de aprendizaje.
- **RNF-2 Requisito de Fiabilidad:** Debe garantizarse el resguardo de la información, de modo que esté duplicada, así como la grabación periódica de la base de datos, de forma tal que se posibilite la reinstalación del sistema y los datos, en caso de algún problema presentado durante la explotación del mismo. El sistema debe estar disponible las 24 horas del día todos los días de la semana. La herramienta de implementación a utilizar debe tener soporte para recuperación ante fallos y errores.
- **RNF-1 Requisito de Eficiencia:** Tiempo de respuesta por transacción 1.09 segundos. Rendimiento: 1000 transacciones por segundos, cantidad de datos que pueden ser transferidos. Utilización de recursos 18 % memoria y 20 % en disco duro.

- **RNF-1 Requisito de Soporte:** Para el servidor de aplicaciones se requiere que esté instalado un intérprete de ficheros PHP, con las últimas actualizaciones del lenguaje. Para el servidor de base de datos se requiere que esté instalado el gestor de base de datos PostgreSQL. Para el cliente se requiere que esté instalado un navegador que interprete JavaScript y versiones HTML 3.0 o superior.
- **RNF-1 Interfaces de usuario:** La interfaz a implementar debe ser sencilla, para que los usuarios que no son expertos en la rama de la informática no necesiten tanto tiempo de adiestramiento. Por el uso diario y constante que tendrá el software, la interfaz debe ser agradable, que favorezca el estado de ánimo del cliente y que combine correctamente los colores, tipo de letra y tamaño, los iconos estén en correspondencia con lo que representan. Deben utilizarse plantillas con un mismo estilo.

2.7 Patrones de diseño de bases de datos.

Los patrones de diseño proveen soluciones reutilizables a problemas de diseño conocidos. Un patrón no define código y no es específico a determinado dominio de programación, sino que por el contrario brinda una solución genérica para determinados problemas de diseño. Asimismo, conllevan una terminología común que puede ser utilizada para documentar de forma sencilla y sistemática diseños propios.[12]

Así como los patrones de diseño documentan soluciones a problemas de diseño comunes, los patrones para persistencia tienen un rol similar en el campo en cuestión. Concretamente se presentan a continuación una serie patrones que son fundamentales a la hora de diseñar una base de datos.[13]

- Representar objetos como tablas.
- Representar relaciones como tablas.
- Patrones de Separación.
- Patrones de Entrada/Salida.
- Patrones de Concurrencia.

En este trabajo es necesario aplicar estos patrones con el fin de diseñar las bases de datos de una manera más fácil. A continuación se ejemplifican los patrones que se usarán:

- Nombre: Representar objetos como tablas.

Problema: ¿Cómo representar un objeto en un esquema de base de datos relacional?
En este trabajo se le da solución a este problema definiendo una tabla para cada clase de objetos persistente. Y los atributos de la clase, que son tipos primitivos, serán las columnas de las tablas.

- Nombre: Representación de relaciones como tablas.

Problema: ¿Cómo representar una relación en un esquema de base de datos relacional?
Para las relaciones uno a uno, o uno a muchos, se colocará la clave foránea en la tabla de cardinalidad uno, para representar la relación entre objetos, o también se podrá crear una tabla asociativa para registrar los identificadores de cada unos de los objetos de la relación.
Para las relaciones muchos a mucho se creará una tabla asociativa para registrar los identificadores de cada uno de los objetos de la relación.

- Nombre: Patrones de separación.

Problema: ¿Cómo aislar los componentes de acceso a datos de otras partes de la aplicación?
Para lograr esta separación entre las capas de persistencia de las diferentes aplicaciones y los demás componentes se usarán algunos patrones como son, DAO y MVC. Este aislamiento brinda flexibilidad a la hora de seleccionar un modelo de datos de soporte, así como en los cambios en la forma de acceso a datos de los sistemas.

❖ Nombre: Patrones de entrada/salida.

Problema: ¿Cómo lograr la mayor eficacia en cuanto a la comunicación entre los objetos de dominio relacional y los datos relacionales? El problema en cuestión, denominado inadaptación de impedancia entre objetos y datos relacionales, es el resultado de las diferencias existentes entre el paradigma de orientación a objetos y el paradigma relacional.

En este trabajo se usará como ORM (Figura 2.2), Propel e Hibernate.

❖ Nombre: Patrones de concurrencia.

Problema: ¿Qué sucede cuando múltiples usuarios realizan operaciones concurrentes que implican datos comunes?

Este aspecto se tuvo en cuenta, debido a que PostgreSQL y Db4o tienen la capacidad de procesamiento transaccional que soporta concurrencia. En el caso de Db4o, sus transacciones permiten la concurrencia y recuperación de datos, mientras que PostgreSQL permite la concurrencia mediante la tecnología MVCC.

2.8 Nomenclatura.

Los nomencladores son importantes para el diseño de una base de datos. Un correcto uso de los mismos permitirá que el diseño realizado sea entendible por otras personas que deseen darle mantenimiento a la base de datos. En ocasiones se desintegran los equipos de desarrollo y la base de datos al igual que el sistema debe posibilitar el mantenimiento por un equipo de soporte.

En este trabajo se hacen uso de los nomencladores definidos por la DI de la UCI.

Las bases de datos del SCAPV, Portal de Seguridad y Protección seguirán las siguientes reglas:

❖ A los nombres de las bases de datos se le antepondrá “UCI_”. Ej. UCI_Acceso.

- A los nombre de las tablas se le antepondrá “Tb_” y cada palabra estará separada por “_”. Ej. Tb_hAcceso_Persona.
- Se antepondrá una “n” para las tablas nomencladoras. Ej. Tb_nCausa_Acceso.
- Se antepondrá una “d “para las tablas de datos. Ej. Tb_dPersona.
- Se antepondrá una “r” para las tablas que representen relaciones. Ej. Tb_rMedio_Portador.
- Se antepondrá una “h “para las tablas de historial. Ej. Tb_hAcceso_Persona.
- Para el carnet de identidad se usará CI.
- Para los campos que sean booleanos se antepondrá “Es_”. Ej. Es_Permitido.
- Para lo campos identificadores se antepondrá “Id_”. Ej. Id_Categoria.
- Los campos usarán “_” para separar las palabras de un campo, irán además con letra mayúscula en el principio. Ej. Nombre_Completo.
- Las tablas nomencladoras tendrán un campo descripción, en el que se especifica el significado del nomenclador en cuestión.

Todos los campos y tablas almacenarán una breve descripción de la información que almacena, con el objetivo de tener un mayor entendimiento de la estructura de las bases de datos.

En la base de datos del Sistema de Gestión de Fotos la nomenclatura a usar será la propuesta por Symfony aunque habrá algunos casos en los que se definirá según convenga.

2.9 Estrategia de integración de los módulos o sistemas.

El proyecto Control de Accesos e Identificación está estructurado en cinco sistemas. Los cuales permitirán aumentar la seguridad y la protección que la UCI requiere.

Las bases de datos diseñadas en este trabajo permitirán una mayor integración entre los sistemas que la conforman. Por ejemplo, el Portal de Seguridad y Protección obtendrá datos y reportes de los sistemas que conforman la estructura de la Figura 2.1, de la misma manera el sistema de Acreditación y el de Incidencias estarán vinculados a la base de datos del Sistema de Gestión de Fotos para obtener las fotos de las personas de la UCI.

Figura 2. 1 Estructura del proyecto Control de Acceso e Identificación.

2.9 Descripción y fundamentación de la arquitectura.

2.9.1 SCAPV.

Para este sistema se diseñaron dos bases de datos, una que tendrá una ubicación remota y la otra local.

La base de datos remota contiene veintinueve tablas, las que permitirán que se tengan almacenados los datos referentes a las personas, vehículos y visitantes autorizados, así como los accesos de los mismos al centro. Se guardarán además los datos de los medios autorizados a salir del centro.

Este sistema usará una arquitectura de tres capas, donde se delimitan las responsabilidades de cada funcionalidad. Las clases de acceso a datos se diseñarán siguiendo el patrón (DAO).

Para el mapeo objeto-relacional entre la aplicación Java y el SGBD PostgreSQL se usará Hibernate. A continuación se justifican las herramientas y lenguajes propuestos.

- **Java** es un lenguaje de programación orientado a objetos de alto nivel, gran rendimiento, sencillo, gran nivel de seguridad, multiplataforma y contiene las herramientas necesarias

para desarrollar cualquier tipo de aplicación, Web, de escritorio, para dispositivos móviles entre otras.

- **Hibernate** es un potente framework de mapeo objeto-relacional y servicio de consultas para Java. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos.

Mapeo Objeto-Relacional (ORM):

Permite mapear los objetos a registro en la base de datos. Consiste en la conversión de cada clase en una tabla, cada objeto en registro (fila) de la base de datos y cada atributo en una columna de la misma.

En la siguiente figura se muestra el proceso de mapeo objeto-relacional.

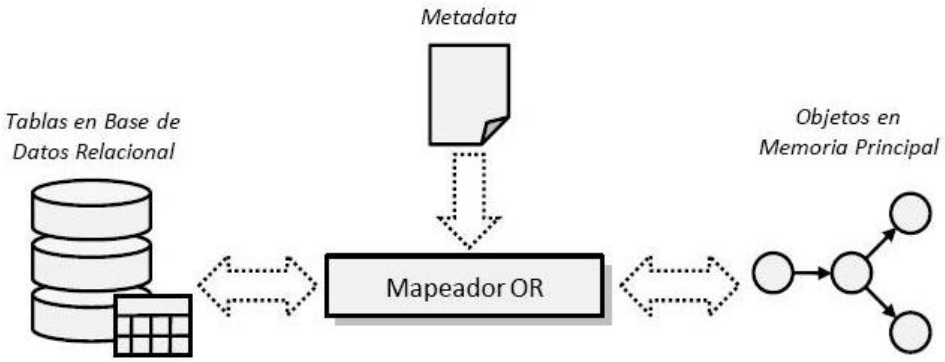

Figura 2. 2 Esquema de un Mapeo Objeto Relacional (ORM).

Es necesario que ambas bases de datos, tanto la local como la remota, sincronicen sus informaciones, esta tarea se realizará mediante Db4o Replication System (dRS) de manera bidireccional.

En la siguiente figura se representa la arquitectura propuesta para las bases de datos del SCAPV.

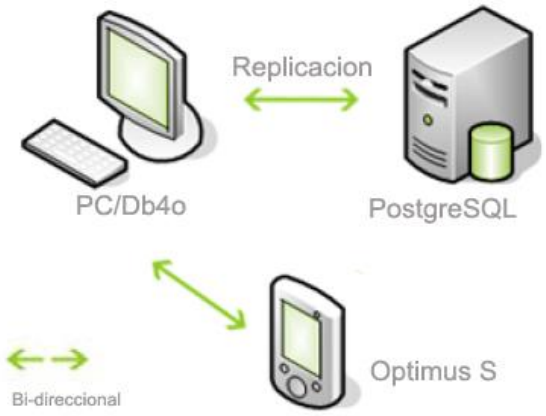

Figura 2. 3 Comunicación entre las bases de datos.

2.9.2 Portal de Seguridad y Protección.

La base de datos propuesta para el Portal de Seguridad y Protección contiene catorce tablas, las que permitirán que se almacenen los datos referentes a los avisos, documentos, informaciones, enlaces y reportes, así como las trazas de auditoría, para una mayor seguridad. Este portal estará vinculado a los servicios Web del sistema de Acreditación y del SCAPV del proyecto Control de Acceso e Identificación, esta comunicación se establecerá a través del protocolo SOAP. Se usará una arquitectura de tres capas.

2.9.3 Sistema de Gestión de Fotos.

La base de datos diseñada para este sistema cuenta con cinco tablas, las que serán utilizadas por dos módulos del sistema: el de Administración y el Servicio Web. El sistema se estructurará según el patrón de arquitectura MVC. Se usará PHP 5 y Symfony 1.0.16 ambos orientados a objetos.

Debido a que la base de datos usará el modelo objeto-relacional, para acceder de forma efectiva a esta base de datos desde un contexto orientado a objetos, es necesaria una interfaz

que traduzca la lógica de los objetos a la lógica relacional. Este sistema usará Propel como ORM (Figura 2.2).

A continuación se justifican las herramientas y lenguajes propuestos.

- **Symfony** es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Symfony está basado en un patrón clásico del diseño Web conocido como arquitectura MVC.
- **PHP 5** incluye los conceptos de clase, objeto, método, herencia y muchos otros propios de la programación orientada a objetos**.**

2.10 Análisis de optimización de consultas (Querys).

La optimización de las consultas es una consideración importante en un sistema de base de datos, ya que la diferencia en tiempo de ejecución entre una buena y una mala estrategia puede ser sustancial.

Existen algunos aspectos a tener en cuenta para obtener un mejor rendimiento en las consultas realizadas, ellos son:

- La cláusula **DISTINCT** es costosa en realizar, debido a que involucra en la mayoría de las veces un ordenamiento para eliminar datos duplicados, no se debe hacer un gran uso de ella, sólo cuando se requiera.
- Es preferible no usar la cláusula **HAVING** si la condición deseada se puede expresar en la cláusula **WHERE**.
- Las condiciones de **JOIN** se pueden evaluar más eficientemente contra un índice primario. Y en general, en términos de rendimiento es preferible evaluar una condición de igualdad numérica que una condición de igualdad sobre cadenas (strings) de caracteres.
- El orden de los **JOIN** en las consultas influye negativamente en el rendimiento de la misma, así como la aplicación que la ejecuta.
- Usar preferiblemente campos que formen la llave primaria dentro de la cláusula **WHERE**.

PostgreSQL posee la función **VACUUM** que posibilita la limpieza, organización de la base de datos y muestra además los reportes de la dispersión de los datos en cada tabla.

Es necesario ejecutar esta funcionalidad periódicamente para así aumentar la velocidad de respuesta ante las consultas del usuario.

Otra funcionalidad que posee PostgreSQL es el comando **EXPLAIN.** Muestra el plan de ejecución de una determinada consulta, que es la manera en que serán escaneadas las tablas involucradas, representa además el costo estimado de ejecución de la consulta.

Tanto el **VACUUM** como el **EXPLAIN** son comandos que se deben tener en cuenta a la hora de realizar las consultas en las bases de datos diseñadas. Estos son una gran ayuda para poder elegir entre las consultas la más eficiente. Ambos comando se usaron para optimizar las consultas realizadas.

El sistema de control de acceso posee historiales de los accesos a la UCI. Con el transcurso del tiempo esto imposibilita la rapidez de respuestas de algunas consultas debido a la gran cantidad de información almacenada en este historial.

Con el objetivo de optimizar el tiempo de respuesta de estas consultas se crearon los campos Estado_Ult_Acceso y Fecha_Ult_Acceso en las tablas Tb_dPersona y Tb_dAuto_Uci.

Es más rápido comprobar el acceso de una persona en aproximadamente 19 000 tuplas de la tabla Tb_dPersona, que comprobarlo en millones de tuplas de las tablas Tb_hAcceso_Persona y Tb_hAcceso. Además el uso de "**join**" demoraría aún más el tiempo de respuesta de la consulta.
A continuación se ejemplifica el análisis anterior, apoyándose del comando **EXPLAIN**.

Se desea saber la fecha y el estado del último acceso de una determinada persona.

Primer caso:

Tiempo de respuesta de la consulta en 19000 tuplas de la tabla Tb_dPersona: 0.09 segundos.

```
SELECT
public.tb_dpersona.estado_ult_acceso,
public.tb_dpersona.fecha_ult_acceso
FROM   public.tb_dpersona
WHERE   (public.tb_dpersona.id_expediente = 'E10575')
```

Segundo caso:

Tiempo de respuesta en 50000 tuplas de la tabla Tb_hAcceso_Persona: 0.17 segundos.

```
SELECT
public.tb_hacceso.fecha_acceso,
public.tb_hacceso.estado
FROM  public.tb_hacceso_persona
INNER JOIN public.tb_hacceso ON
(public.tb_hacceso_persona.id_acceso=public.tb_hacceso.id_acceso)
WHERE  (public.tb_hacceso_persona.id_expediente = 'E10575') ORDER BY   fecha_acceso DESC
LIMIT 1
```

El tiempo de respuesta de la segunda consulta aumentará con el tiempo, mientras que el de la primera se mantendrá rondando ese valor.

A continuación se hacen uso de las funciones "**Date_Part**", **"LIKE** "y **BETWEEN**, con el objetivo de optimizar las consultas.

Se desea conocer la cantidad de acceso en una fecha determinada.

Tiempo de respuesta en 50000 tuplas de la tabla Tb_hAcceso: 0.30 segundos.

```
SELECT count (*)
FROM public.tb_hacceso
WHERE
 Date_Part ('Year', public.tb_hacceso.fecha_acceso) = 2008
AND Date_Part ('Month', public.tb_hacceso.fecha_acceso) =5;
```

Tiempo de respuesta en 50000 tuplas de la tabla Tb_hAcceso: 0.25 segundos.

```
SELECT
count (*)
FROM public.tb_hacceso
WHERE   (public.tb_hacceso.fecha_acceso LIKE '2008-06%')
```

Tiempo de respuesta en 50000 tuplas de la tabla Tb_hAcceso: 0.17 segundos.

```
SELECT
count (*)
 FROM tb_hacceso WHERE tb_hacceso.fecha_acceso BETWEEN '2008-06-01 0:00:00’ AND '2008-06-30 23:59:59'
```

Si se usa funciones como Date_Part o comparaciones con LIKE, el "**Planner**" de PostgreSQL no identifica el campo que tiene índice y se lanza a leer secuencialmente la tabla entera. Al poner el campo como una comparación con **BETWEEN** se puede ir directo al índice y la consulta resultaría más rápida.

2.11 Diagramas de Clases Persistentes (DCP).

Las clases persistentes por lo general tienen como origen las clases entidad porque ellas modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto del mundo real o un suceso. Todas las clases identificadas en el dominio del análisis no son persistentes. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo.

A continuación se muestran los diagramas de clases persistentes esencial para lograr la persistencia de los objetos en los sistemas estudiados en el presente trabajo.

La posibilidad de sincronizar los diagramas en el Visual Paradigm posibilitó que el los DCP se obtuvieran de manera sincronizada con los Diagramas de Entidad Relación (DER).

2.11.1 Diagrama de clases persistentes del SCAPV.

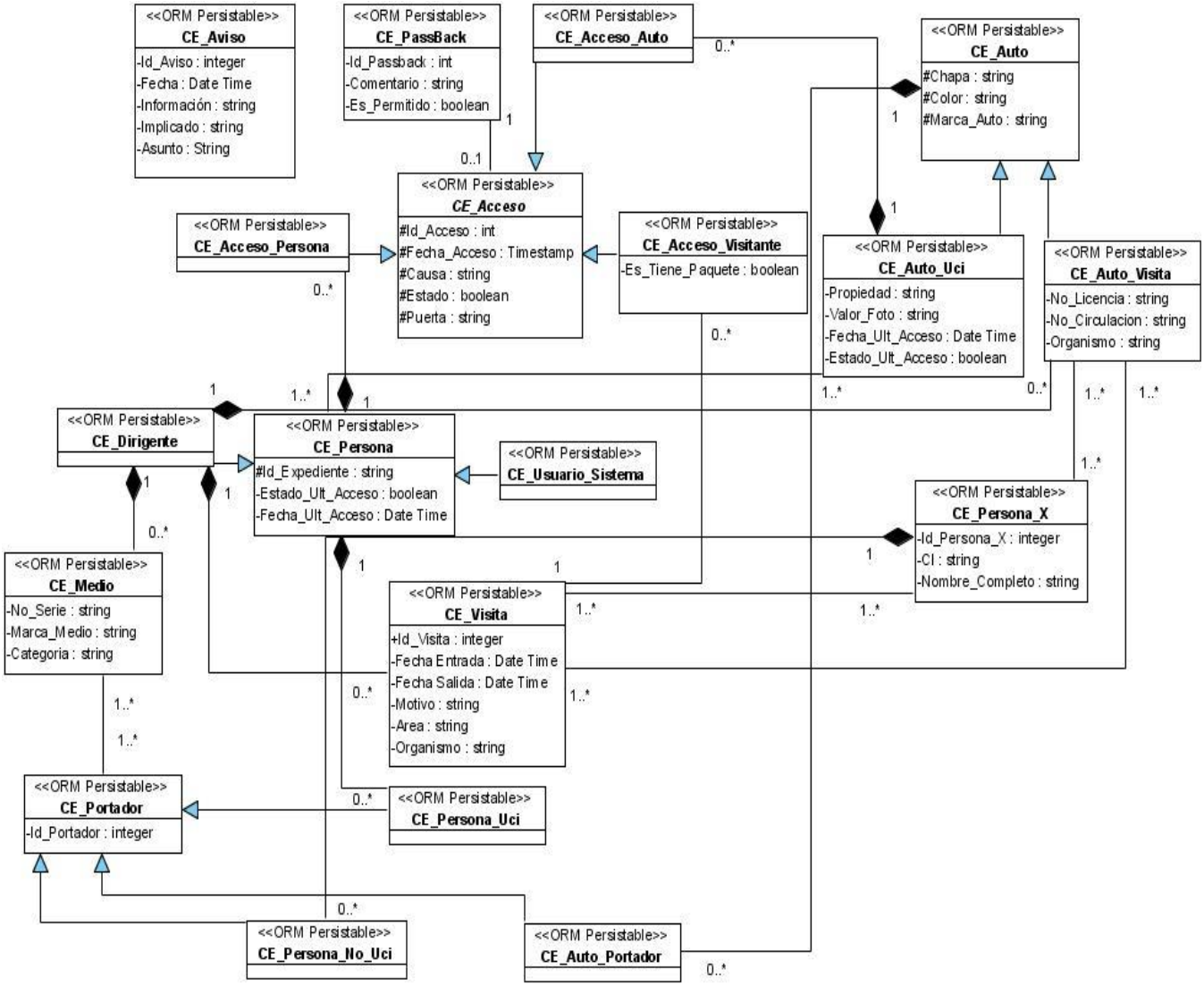

Figura 2. 4 DCP del SCAPV.

2.11.1.1 Descripción de las Clases Persistentes del SCAPV.

Nombre: CE_Aviso	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Aviso	integer
Fecha	Date Time
Asunto	string
Informacion	string
Implicado	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Aviso.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 1 Descripción de la clase CE_Aviso.

Nombre: CE_PassBack	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_PassBack	integer
Comentario	string
Es_Permitido	boolean
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_PassBack.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 2 Descripción de la clase CE_PassBack.

Nombre: CE_Acceso_Persona	
Tipo de clase: Entidad	
Atributo	**Tipo**
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Acceso_Persona.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 3 Descripción de la clase CE_Acceso_Persona.

Nombre: CE_Acceso_Auto	
Tipo de clase: Entidad	
Atributo	**Tipo**
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Acceso_Auto.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 4 Descripción de la clase CE_Acceso_Auto.

Nombre: CE_ Persona_X	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Persona_X	integer
CI	string
Nombre_Completo	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_ Persona_X.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 5 Descripción de la clase CE_Persona_X.

Nombre: CE_Auto	
Tipo de clase: Entidad	
Atributo	**Tipo**
Chapa	string
Marca_Auto	string
Color	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_ Auto.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 6 Descripción de la clase CE_ Auto.

Nombre: CE_Auto_Uci	
Tipo de clase: Entidad	
Atributo	**Tipo**
Valor_Foto	string
Propiedad	string
Fecha_Ult_Acceso	DateTime
Estado_Ult_Acceso	boolean
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Auto_Uci.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 7 Descripción de la clase CE_Auto_Uci.

Nombre: CE_Visita	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Visita	int
Fecha_Entrada	DateTime
Fecha_Salida	DateTime
Motivo	string
Organismo	string
Area	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Visita.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 8 Descripción de la clase CE_Visita.

Nombre: CE_Auto_Visita	
Tipo de clase: Entidad	
Atributo	**Tipo**
No_Licencia	string
No_Circulacion	string
Organismo	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Auto_Visita.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 9 Descripción de la clase CE_Auto_Visita.

Nombre: CE_Acceso_Visitante	
Tipo de clase: Entidad	
Atributo	**Tipo**
Es_Tiene_Paquete	boolean
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Acceso_Visitante.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 10 Descripción de la clase CE_Acceso_Visitante.

Nombre: CE_Usuario_Sistema	
Tipo de clase: Entidad	
Atributo	**Tipo**
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Usuario_Sistema.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 11 Descripción de la clase CE_Usuario_Sistema.

Nombre: CE_Acceso	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Acceso	string
Fecha_Acceso	timestamp
Estado	boolean
Causa	string
Puerta	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Acceso.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 12 Descripción de la clase CE_Acceso.

Nombre: CE_Persona	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Expediente	string
Estado_Ult_Acceso	boolean
Fecha_Ult_Acceso	DateTime
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Persona.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 13 Descripción de la clase CE_Persona.

Nombre: CE_Dirigente	
Tipo de clase: Entidad	
Atributo	**Tipo**
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Dirigente.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 14 Descripción de la clase CE_Dirigente.

Nombre: CE_Medio	
Tipo de clase: Entidad	
Atributo	**Tipo**
No_Serie	string
Marca_Medio	string
Categoria	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Medio.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 15 Descripción de la clase CE_Medio.

Nombre: CE_Portador	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Portador	integer
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Portador
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 16 Descripción de la CE_ Portador.

Nombre: CE_ Auto _Portador	
Tipo de clase: Entidad	
Atributo	**Tipo**
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Auto_Portador
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 17 Descripción de la CE_Auto_Portador.

Nombre: CE_Persona_Uci	
Tipo de clase: Entidad	
Atributo	**Tipo**
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Persona_Uci.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 18 Descripción de la clase CE_Persona_Uci.

Nombre: CE_Persona_No_Uci	
Tipo de clase: Entidad	
Atributo	**Tipo**
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Persona_No_Uci
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 19 Descripción de la clase CE_Persona_No_Uci.

2.11.2 Diagrama de clases persistentes Portal del Seguridad y Protección.

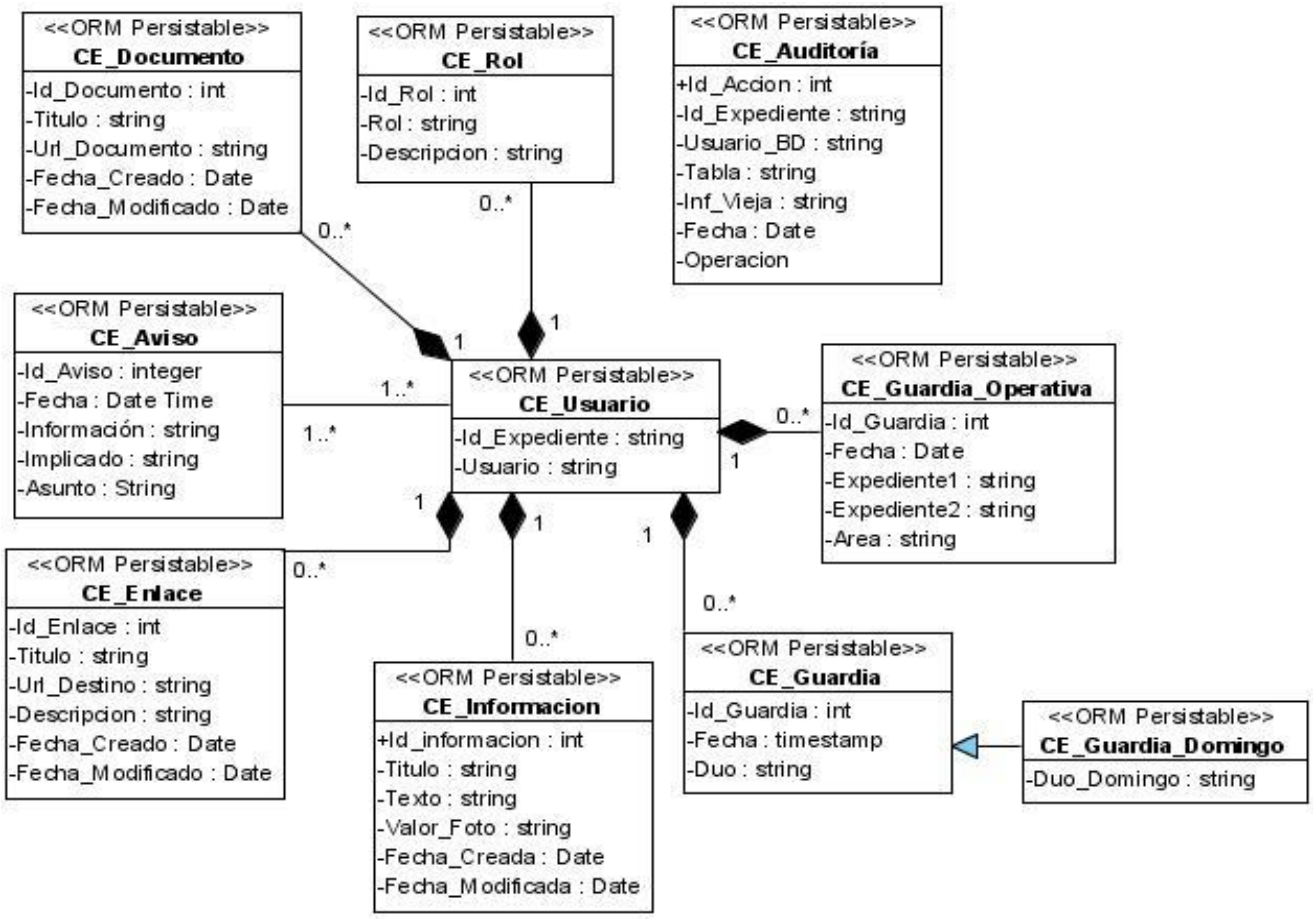

Figura 2. 5 DCP del Portal de Seguridad y Protección.

2.11.2.1 Descripción de las Clases Persistentes del Portal del Seguridad y Protección.

Nombre: CE_Guardia_Domingo	
Tipo de clase: Entidad	
Duo_Domingo	string
Atributo	**Tipo**
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Guardia_Domingo.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 20 Descripción de la clase CE_ Guardia _Domingo.

Nombre: CE_Documento	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Documento	integer
Titulo	string
Url_Documento	string
Fecha_Creado	string
Fecha_Modificado	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Documento.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 21 Descripción de la clase CE_Documento.

Nombre: CE_Aviso	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Aviso	integer
Asunto	string
Informacion	string
Fecha_Aparicion	string
Fecha_Caducidad	string
Implicado	string
Prioridad	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Aviso.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 22 Descripción de la clase CE_Aviso.

Nombre: CE_Informacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Informacion	integer
Titulo	string
Texto	string
Valor_Foto	string
Fecha_Creada	string
Fecha_Modificada	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Informacion.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 23 Descripción de la clase CE_Informacion.

Nombre: CE_Usuario	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Expediente	string
Usuario	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Usuario.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 24 Descripción de la clase CE_Usuario.

Nombre: CE_Enlace	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Enlace	integer
Titulo	string
Url_Destino	string
Descripcion	string
Fecha_Creado	string
Fecha_Modificado	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Enlace.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 25 Descripción de la clase CE_Enlace.

Nombre: CE_Guardia	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Guardia	integer
Fecha	string
Duo	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Guardia.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 26 Descripción de la clase CE_Guardia.

Nombre: CE_Guardia_Operativa	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Guardia	integer
Fecha	string
Area	string
Expediente1	string
Expediente2	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Guardia_Operativa.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 27 Descripción de la clase CE_Guardia_Operativa.

Tabla 28 Descripción de la clase CE_Rol.

Nombre: CE_Rol	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Rol	integer
Nombre	string
Descripcion	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Rol.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

2.11.3 Diagrama de clases persistentes del Sistema de Gestión de Fotos.

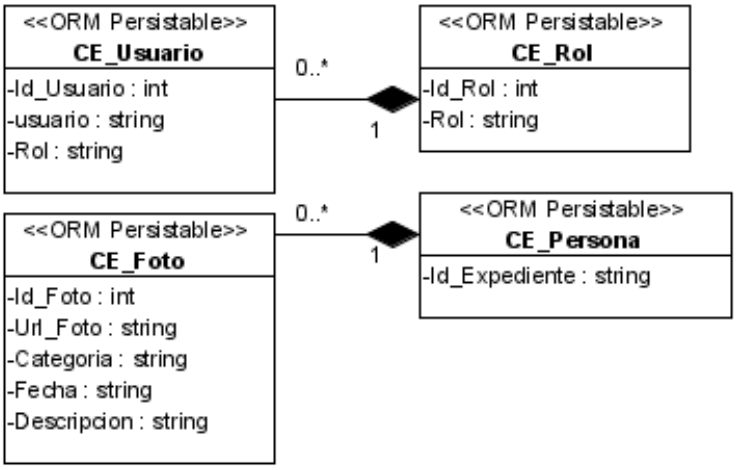

Figura 2. 6 DCP del Sistema de Gestión de Fotos.

2.11.3.1 Descripción de las Clases Persistentes del Sistema de Gestión de Fotos.

Nombre: CE_Usuario	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Usuario	integer
Usuario	string
Rol	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Usuario.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 29 Descripción de la clase CE_Usuario.

Nombre: CE_Rol	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Rol	integer
Rol	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Rol.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 30 Descripción de la clase CE_Rol.

Nombre: CE_Persona	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Expediente	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Persona
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 31 Descripción de la clase CE_Persona.

Nombre: CE_Foto	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Foto	string
Fecha	string
Url_Foto	string
Descripcion	string
Categoria	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad CE_Foto.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 32 Descripción de la clase CE_Foto.

2.12 Diseño de la base de datos.

Las bases de datos necesitan una definición de su estructura que les permita almacenar datos, reconocer el contenido y recuperar la información. Su estructura debe satisfacer los requisitos de los sistemas, siendo el punto de partida de los diseños de las bases de datos. La puesta en práctica de la base de datos es el paso final en el desarrollo de aplicaciones de soporte del negocio.

Para el diseño de la base de datos se realizó el diagrama de clases persistentes, mostrado anteriormente, y el diagrama entidad relación. Ambos fueron diseñados en el DB Visual Architect, herramienta del Visual Paradigm.

Todas las clases identificadas en el dominio del análisis no son persistentes.

Lo contrario son las clases temporales que son manejadas y almacenadas por el sistema en tiempo de ejecución por lo que dejan de existir cuando termina el programa.

En los diagramas entidad relación de las bases de datos diseñadas se especifican los detalles físicos de las bases de datos. A continuación se muestran dichos diagramas, con las respectivas descripciones de las tablas involucradas.

2.12.1 DER de la base de datos del SCAPV

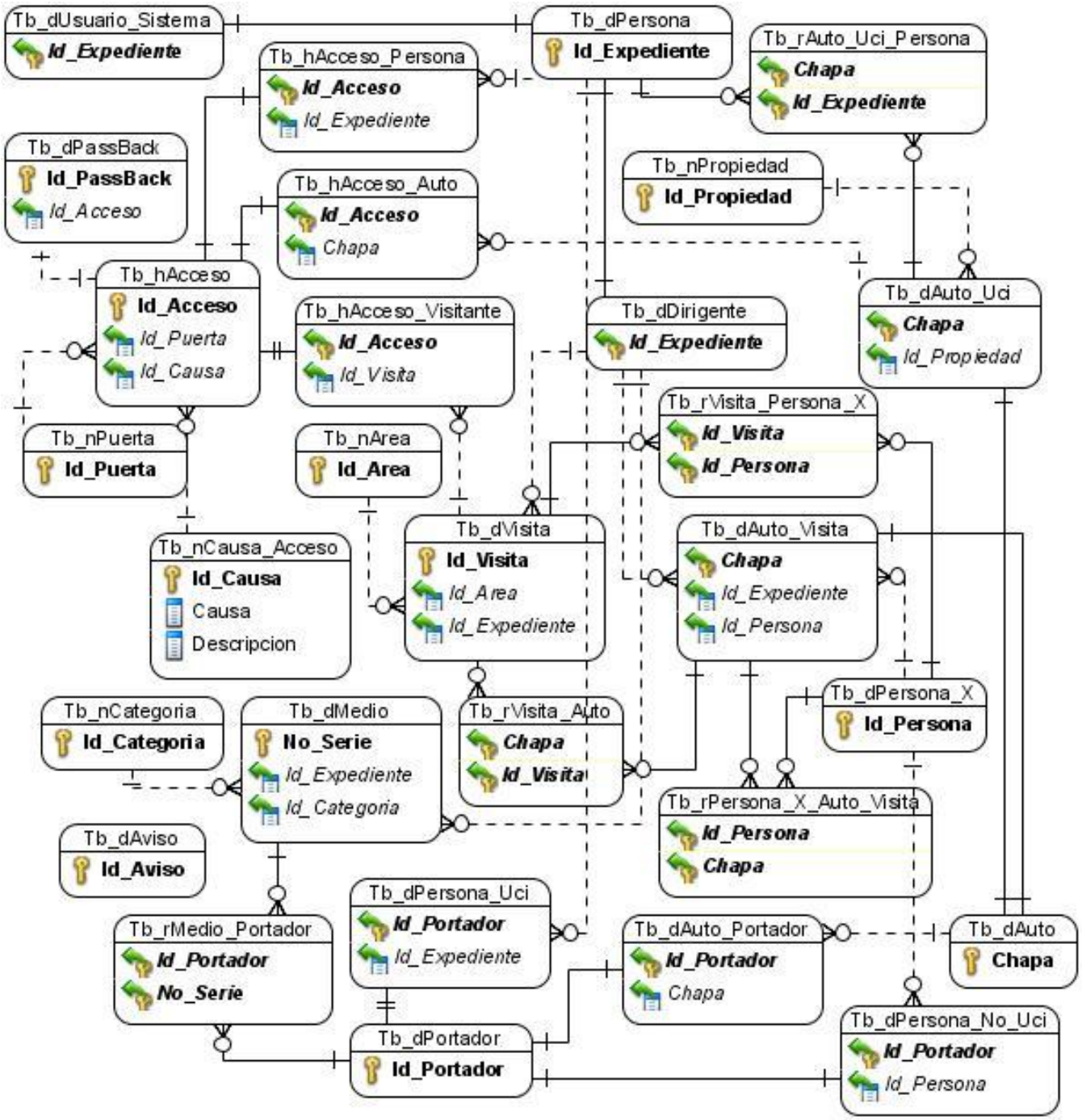

Figura 2. 7 DER del SCAPV

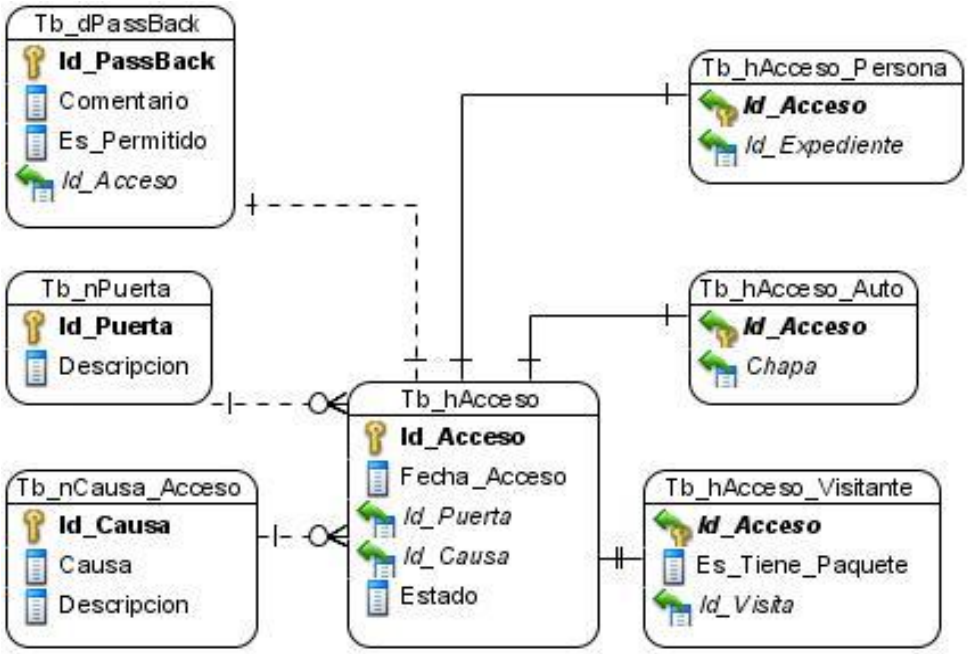

Figura 2. 8 Subdiagrama 1 del DER de la base de datos del SCAPV.

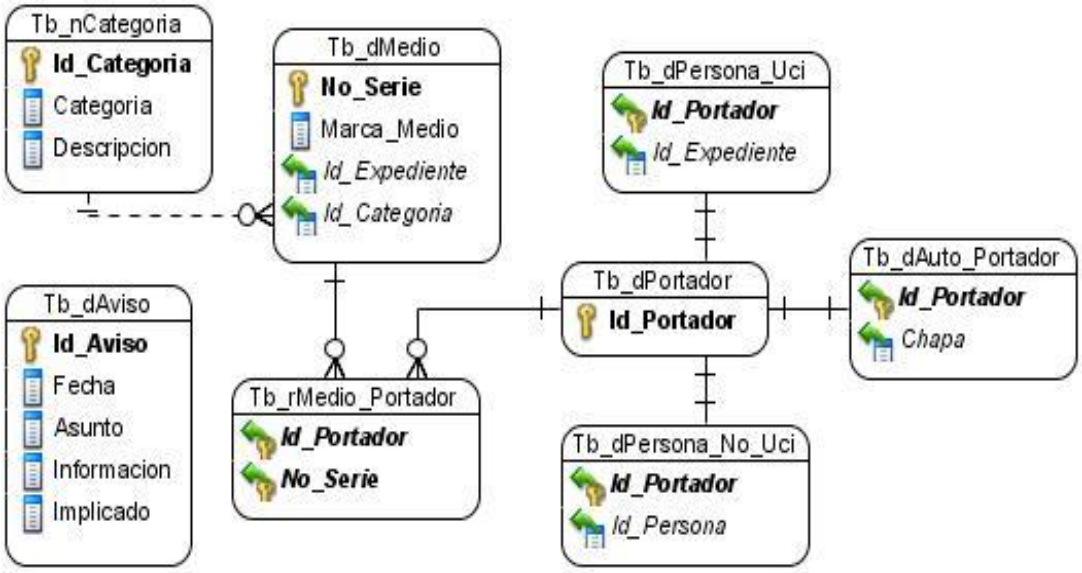

Figura 2. 9 Subdiagrama 2 del DER de la base de datos del SCAPV.

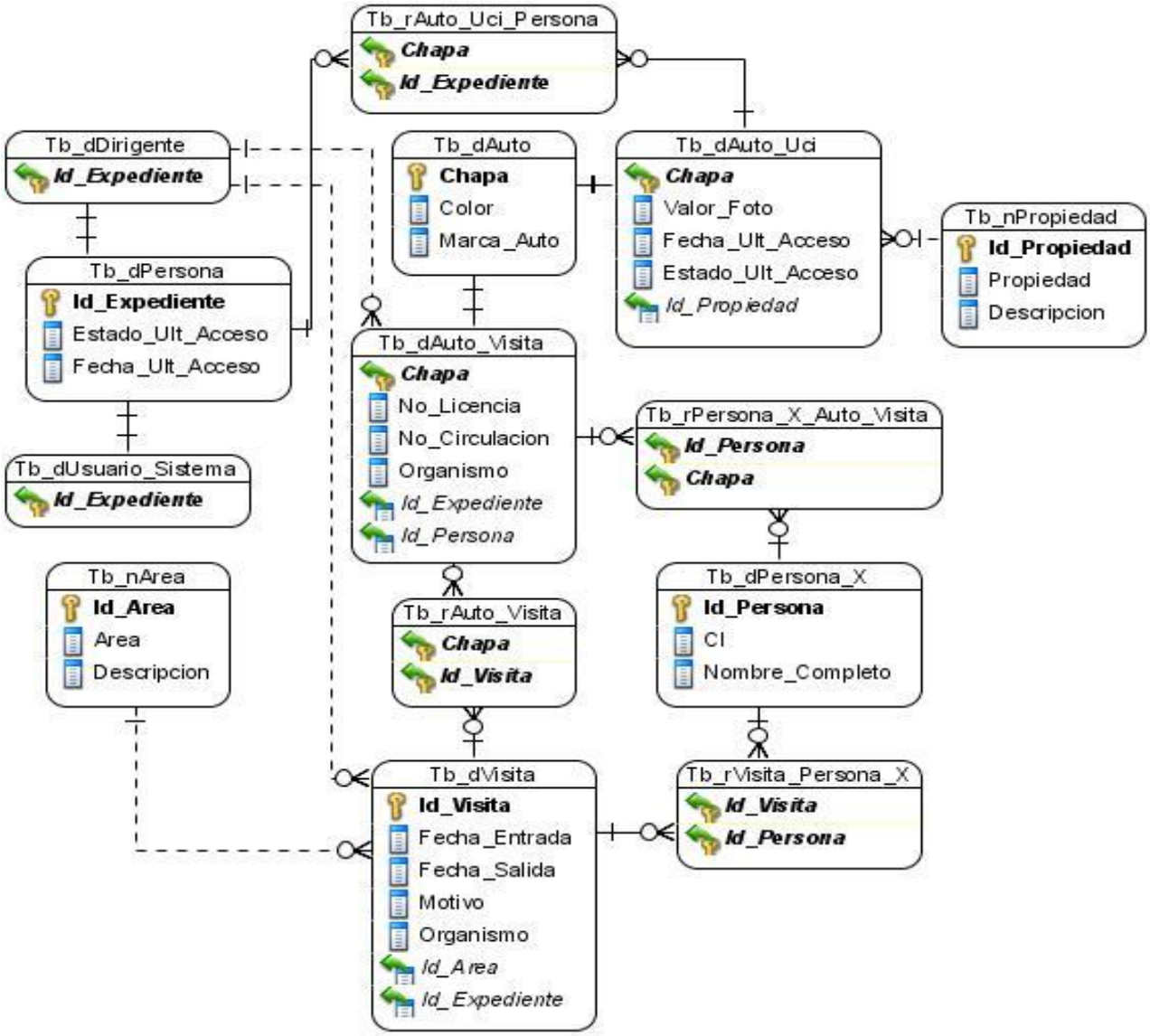

Figura 2. 10 Subdiagrama 3 del DER de la base de datos del SCAPV.

2.12.1.1 Descripción de las tablas de la base de datos remota del SCAPV.

Nombre: Tb_nPuerta			
Descripción: Guarda la información referente a las puertas de acceso.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Puerta	varchar(255)	No	Nombre de la puerta.
Descripcion	varchar(255)	Si	Breve descripción.

Tabla 33 Descripción de la tabla Tb_nPuerta.

Nombre: Tb_dAviso			
Descripción: Guarda la información de los avisos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Aviso	integer	No	Llave primaria.
Fecha	timestamp	No	Fecha del aviso.
Asunto	varchar(255)	No	Asunto del aviso.
Informacion	varchar(255)	No	Información del aviso.
Implicado	varchar(100)	No	Implicado en el aviso.

Tabla 34 Descripción de la tabla Tb_dAviso.

Nombre: Tb_dPassBack			
Descripción: Guarda la información referente a los Passback.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_PassBack	integer	No	Llave primaria.
Comentario	varchar(255)	Si	Comentario
Es_Permitido	bool	No	Si se permitió o no el acceso.
Id_Acceso	integer	No	Llave foránea.

Tabla 35 Descripción de la tabla Tb_dPassBack.

Nombre: Tb_nCausa_Acceso			
Descripción: Guarda la información referente a las causas de accesos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Causa	integer	No	Llave primaria.
Causa	varchar(255)	No	Causa de acceso.
Descripcion	varchar(255)	Si	Breve descripción.

Tabla 36 Descripción de la tabla Tb_nCausa_Acceso.

Nombre: Tb_dAuto_Visita			
Descripción: Guarda la información referente a los autos y choferes de las visitas.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Chapa	varchar(6)	No	Llave primaria.
No_Licencia	varchar(255)	No	Número de licencia.
No_Circulacion	varchar(255)	No	Número de circulación.
Organismo	varchar(255)	No	Organismo al que pertenece.
Id_Expediente	varchar(15)	No	Llave foránea.
Id_Persona	integer	No	Llave foránea.

Tabla 37 Descripción de la tabla Tb_dAuto_Visita.

Nombre: Tb_nArea			
Descripción: Guarda la información referente a las áreas.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Area	integer	No	Llave primaria.
Area	Varchar(100)	No	Nombre del área.
Descripcion	varchar(255)	Si	Breve descripción.

Tabla 38 Descripción de la tabla Tb_nArea

Nombre: Tb_dAuto			
Descripción: Guarda la información referente a los autos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Chapa	varchar(6)	No	Llave primaria.
Marca_Auto	varchar(50)	No	Marca del auto.
Color	varchar(50)	No	Color del auto.
Fecha_Ult_Acceso	timestamp	No	Fecha del último acceso.
Estado_Ult_Acceso	bool	No	Estado del último acceso.

Tabla 39 Descripción de la tabla Tb_dAuto.

Nombre: Tb_dAuto_Uci			
Descripción: Guarda la información referente al auto portador del medio, el auto pertenece a la UCI.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Chapa	varchar(6)	No	Llave primaria.
Valor_Foto	varchar(255)	No	Valor de la foto.
Id_Propiedad	integer	No	Llave foránea.
Fecha_Ult_Acceso	timestamp	No	Fecha del último acceso.
Estado_Ult_Acceso	bool	No	Estado del último acceso.

Tabla 40 Descripción de la tabla Tb_dAuto_Uci.

Nombre: Tb_rAuto_Uci_Persona			
Descripción: Guarda la información referente a los autos y sus choferes.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Expediente	varchar(15)	No	Llave primaria.
Chapa	varchar(6)	No	Llave primaria.

Tabla 41 Descripción de la tabla Tb_rAuto_Uci_Persona.

Nombre: Td_nPropiedad			
Descripción: Guarda la información referente a las propiedad del auto.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Propiedad	integer	No	Llave primaria.
Propiedad	varchar(100)	No	Propiedad del auto.
Descripcion	varchar(255)		Breve descripción.

Tabla 42 Descripción de la tabla Tb_nPropiedad.

Nombre: Td_hAcceso_Auto			
Descripción: Guarda el historial de los accesos de los autos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Acceso	integer	No	Llave primaria.
Chapa	varchar(6)	No	Llave foránea.

Tabla 43 Descripción de la tabla Tb_hAcceso_Auto.

Nombre: Tb_dPersona			
Descripción: Guarda la información referente a las personas de la UCI.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Expediente	varchar(10)	No	Llave primaria.
Estado_Ult_Acceso	bool	No	Estado del último acceso de la persona.
Fecha_Ult_Acceso	timestamp	No	Fecha del último acceso de la persona.

Tabla 44 Descripción de la tabla Tb_dPersona.

Nombre: Tb_hAcceso_Persona			
Descripción: Guarda el historial de los accesos de las personas.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Acceso	integer	No	Llave primaria.
Id_Expediente	varchar(10)	No	Llave foránea.

Tabla 45 Descripción de la tabla Tb_hAcceso_Persona.

Nombre: Tb_ rVisita_Auto			
Descripción: Guarda la información referente a los cho0feres de los autos de visita.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Persona	integer	No	Llave primaria.
Chapa	varchar(6)	No	Llave primaria.

Tabla 46 Descripción de la tabla Tb_ rVisita_Auto.

Nombre: Tb_hAcceso			
Descripción: Guarda el historial de los accesos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Acceso	integer	No	Llave primaria.
Fecha_Acceso	timestamp	No	Fecha del acceso.
Id_Puerta	varchar(50)	No	Llave foránea.
Id_Causa	integer	No	Llave foránea.
Estado	bool	No	Estado del acceso.

Tabla 47 Descripción de la tabla Tb_hAcceso.

Nombre: Tb_dMedio			
Descripción: Guarda la información referente a los medios.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
No_Serie	varchar(255)	No	Llave primaria.
Marca_Medio	varchar(255)	No	Marca del medio.
Id_Categoria	integer	No	Llave foránea.
Id_Expediente	varchar(15)	No	Llave foránea.

Tabla 48 Descripción de la tabla Tb_dMedio.

Nombre: Tb_dVisita			
Descripción: Guarda la información referente a las visitas autorizadas.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Visita	integer	No	Llave primaria.
Motivo	varchar(255)	No	Motivo de la visita.
Fecha Entrada	timestamp	No	Fecha de entrada.
Fecha Salida	timestamp	No	Fecha de salida.
Organismo	varchar(50)	No	Organismo al que pertenece.
Id_Expediente	varchar(15)	No	Llave foránea.
Id_Area	integer	No	Llave foránea.

Tabla 49 Descripción de la tabla Tb_dVisita.

Nombre: Tb_ nCategoria			
Descripción: Guarda la información referente a las categorías de medios.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Categoria	integer	No	Llave primaria.
Categoria	varchar(100)	No	Categoría del medio.
Descripcion	varchar(255)	Si	Breve descripción.

Tabla 50 Descripción de la tabla Tb_nCategoria.

Nombre: Tb_dPortador			
Descripción: Guarda la información del portador del medio.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Portador	integer	No	Llave primaria.

Tabla 51 Descripción de la tabla Tb_dPortador.

Nombre: Tb_rMedio_Portador			
Descripción: Guarda el identificador del medio y su portador.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
No_Serie	varchar(100)	No	Llave primaria.
Id_Portador	integer	No	Llave primaria.

Tabla 52 Descripción de la tabla Tb_rMedio_Portador

Nombre: Tb_dAuto_Portador			
Descripción: Guarda la información referente al auto portador del medio.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Portador	integer	No	Llave primaria.
Chapa	varchar(6)	No	Llave foránea.

Tabla 53 Descripción de la tabla Tb_dAuto_Portador.

Nombre: Tb_dPersona_Uci			
Descripción: Guarda la información referente a la persona portador del medio. Perteneciente a la UCI.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Portador	integer	No	Llave primaria.
Id_Expediente	varchar(15)	No	Llave foránea.

Tabla 54 Descripción de la tabla Tb_dPersona_Uci.

Nombre: Tb_dPersona_No_Uci			
Descripción: Guarda la información referente a la persona portador del medio. No perteneciente a la UCI.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Portador	integer	No	Llave primaria.
Id_Persona	integer	No	Llave foránea.

Tabla 55 Descripción de la tabla Tb_dPersona_No_Uci.

Nombre: Tb_dUsuario_Sistema			
Descripción: Guarda la información referente a los usuario del sistema.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Expediente	varchar(15)	No	Llave primaria.

Tabla 56 Descripción de la tabla Tb_dUsuario_Sistema.

Nombre: Tb_rVisita_Auto			
Descripción: Guarda la información referente a las visitas en autos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Visita	integer	No	Llave primaria.
Chapa	varchar(6)	No	Llave primaria.

Tabla 57 Descripción de la tabla Tb_rVisita_Auto.

Nombre: Tb_rVisita_Persona_X			
Descripción: Guarda la información referente a las visitas de las personas.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Visita	integer	No	Llave primaria.
Id_Persona	integer	No	Llave primaria.

Tabla 58 Descripción de la tabla Tb_ rVisita_Persona_X.

Nombre: Tb_hAcceso_Visitante			
Descripción: Guarda el historial de los accesos de los visitantes.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Acceso	integer	No	Llave primaria.
Es_Tiene_Paquete	bool	No	Si trae paquetes o no.
Id_Visita	integer	No	Llave foránea.

Tabla 59 Descripción de la tabla Tb_hAcceso_Visitante.

Nombre: Tb_dPersona_X			
Descripción: Guarda la información referente a las personas que no son de la UCI.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Persona	integer	No	Llave primaria.
CI	varchar(11)	No	Número de identidad.
Nombre_Completo	varchar(255)	No	Nombre completo de la persona.

Tabla 60 Descripción de la tabla Tb_dPersona_X.

Nombre: Tb_dDirigente			
Descripción: Guarda la información referente a los dirigentes de la UCI.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Expediente	varchar(15)	No	Llave primaria.

Tabla 61 Descripción de la tabla Tb_dDirigente.

2.12.2 DER de la base de datos del Portal de Seguridad y Protección.

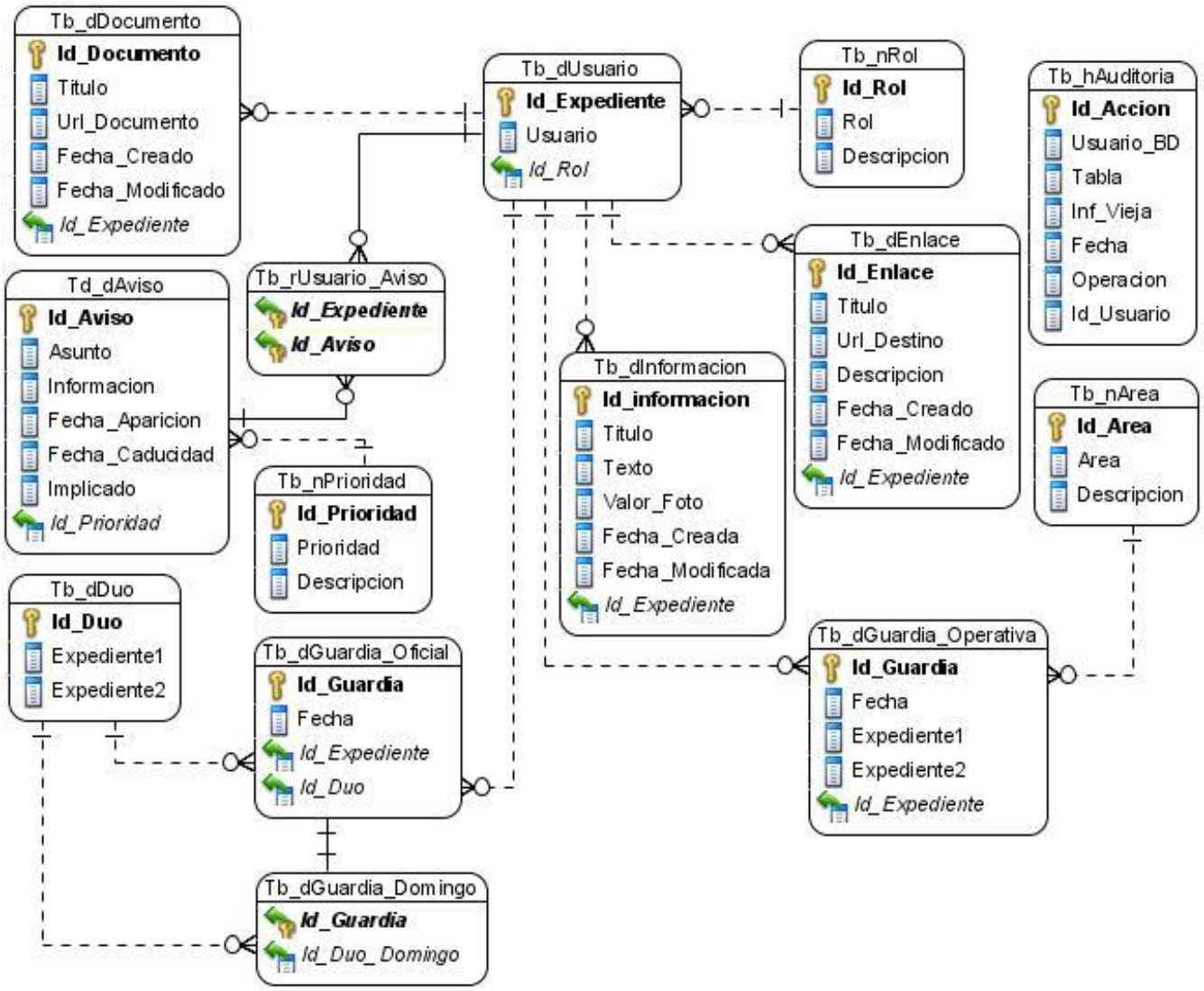

Figura 2. 11 DER del Portal de Seguridad y Protección.

2.11.2.1 Descripción de las tablas de la base de datos del Portal de Seguridad y Protección.

Nombre: Tb_dUsuario			
Descripción: Guarda la información de los usuarios.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Expediente	varchar(15)	No	Llave Primaria.
Usuario	varchar(255)	No	Usuario del sistema.
Id_Rol	integer	No	Llave foránea.

Tabla 62 Descripción de la tabla Tb_dUsuario.

Nombre: Tb_dAviso			
Descripción: Guarda la información de los avisos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Aviso	integer	No	Llave primaria.
Fecha_Aparicion	timestamp	No	Fecha de aparición del aviso.
Asunto	varchar(255)	No	Asunto del aviso.
Informacion	varchar(255)	No	Información del aviso.
Fecha_Caducidad	timestamp	No	Fecha de vencimiento del aviso.
Implicado	varchar(255)	No	Implicado en el aviso.
Id_Expediente	varchar(15)	No	Llave foránea.
Id_Prioridad	integer	No	Llave foránea.

Tabla 63 Descripción de la tabla Tb_dAviso.

Nombre: Tb_nPrioridad			
Descripción: Guarda la información de las prioridades de los avisos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Prioridad	integer	No	Llave primaria.
Prioridad	varchar(255)	No	Prioridad del aviso.
Descripción.	varchar(255)	Si	Breve descripción.

Tabla 64 Descripción de la tabla Tb_nPrioridad.

Nombre: Tb_nRol			
Descripción: Guarda la información de los roles.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Rol	integer	No	Llave primaria.
Rol	varchar(50)	No	Nombre del rol del usuario.
Descripción.	varchar(255)	Si	Breve descripción del rol.

Tabla 65 Descripción de la tabla Tb_nRol.

Nombre: Tb_rUsuario_Aviso			
Descripción: Guarda la información de los usuarios y avisos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Expediente	varchar(15)	No	Llave primaria.
Id_Aviso	integer	No	Llave primaria.

Tabla 66 Descripción de la tabla Tb_rUsuario_Aviso

Nombre: Tb_dDocumento			
Descripción: Guarda la información de los documentos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción**
Id_Documento	integer	No	Llave primaria.
Titulo	varchar(255)	No	Titulo del documento
Url_Documento	varchar(255)	No	Url_Documento
Fecha_Creado	timestamp	No	Fecha_Creado
Fecha_Modificado	timestamp	No	Fecha_Modificado
Id_Expediente	varchar(15)	No	Llave foránea.

Tabla 67 Descripción de la tabla Tb_dDocumento.

Nombre: Tb_dGuardia_Domingo			
Descripción: Guarda la información de las guardias de domingo.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Guardia	integer	No	Llave primaria.
Id_Duo_Domingo	integer	No	Llave foránea.

Tabla 68 Descripción de la tabla Tb_dGuardia_Domingo.

Nombre: Tb_dInformacion			
Descripción: Guarda las informaciones.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Informacion	integer	No	Llave primaria.
Titulo	varchar(100)	No	Titulo de la información.
Texto	varchar(255)	No	Texto de la información.
Valor_Foto	varchar(255)	No	Valor de la foto.
Fecha_Creado	timestamp	No	Fecha de creado el documento.
Fecha_Modificado	timestamp	No	Fecha de modificación del documento.
Id_Expediente	varchar(15)	No	Llave foránea.

Tabla 69 Descripción de la tabla Tb_dInformacion.

Nombre: Tb_dGuardia_Oficial			
Descripción: Guarda la información de las guardias.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Guardia	integer	No	Llave primaria.
Fecha	Date	No	Fecha de la guardia.
Id_Duo	integer	No	Llave foránea.
Id_Expediente	varchar(15)	No	Llave foránea.

Tabla 70 Descripción de la tabla Tb_dGuardia_Oficial.

Nombre: Tb_dDuo			
Descripción: Guarda la información de los dúos de guardia.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Duo	integer	No	Llave primaria.
Expediente1	varchar(15)	No	Identificador de la primera persona.
Expediente2	varchar(15)	No	Identificador de la segunda persona.

Tabla 71 Descripción de la tabla Tb_dDuo.

Nombre: Tb_dEnlace			
Descripción: Guarda la información de los enlaces.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Enlace	integer	No	Llave primaria.
Titulo	varchar(100)	No	Titulo de la información.
Url_Destino	varchar(255)	No	Texto de la información.
Descripcion	varchar(255)	Si	Descripcion del enlace.
Fecha_Creado	timestamp	No	Fecha de creación del enlace.
Fecha_Modificado	timestamp	No	Fecha de modificación del enlace.
Id_Expediente	varchar(15)	No	Llave foránea.

Tabla 72 Descripción de la tabla Tb_dEnlace.

Nombre: Tb_nArea			
Descripción: Guarda la información de las áreas de guardia			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Area	integer	No	Llave primaria.
Area	varchar(255)	No	Nombre del área de guardia.
Descripcion	varchar(255)	Si	Breve descripción.

Tabla 73 Descripción de la tabla Tb_nArea.

Nombre: Tb_dGuardia_Operativa			
Descripción: Guarda la información de la guardia operativa.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Guardia	integer	No	Llave primaria.
Fecha	Date	No	Fecha de la guardia.
Expediente1	varchar(15)	No	Identificador de la primera persona.
Expediente2	varchar(15)	No	Identificador de la segunda persona.
Id_Expediente	varchar(15)	No	Llave foránea.
Id_Area	integer	No	Llave foránea.

Tabla 74 Descripción de la tabla Tb_dGuardia_Operativa.

Nombre: Tb_hAuditoria			
Descripción: Guarda la información de las operaciones realizadas en la base de datos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Accion	integer	No	Llave primaria.
Id_Expediente	varchar(15)	No	Llave foránea.
Usuario_BD	varchar(100)	No	Usuario de la base de datos.
Tabla	varchar(100)	No	Nombre de la tabla donde ocurrió la acción.
Inf_Vieja	varchar(255)	No	Información antes de realizar la acción.
Fecha	timestamp	No	Fecha en que ocurrió la acción.
Operación	varchar(100)	No	Operación realizada en la base de datos.

Tabla 75 Descripción de la tabla Tb_hAuditoria.

2.12.3 DER de sistema de Gestión de Fotos.

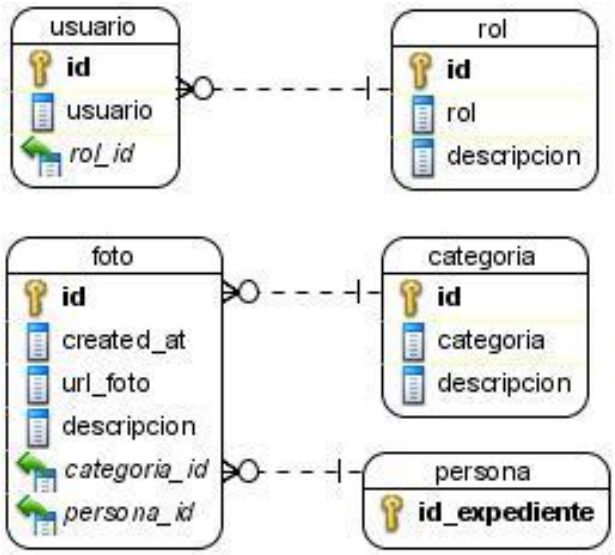

Figura 2. 12 DER del Sistema de Gestión de Fotos.

2.12.3.1 Descripción de las tablas de la base de datos del Sistema de Gestión de Fotos.

Nombre: foto			
Descripción: Guarda la información de las fotos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id	integer	No	Llave primaria.
created_at	timestamp	No	Fecha de creada.
url_foto	varchar(255)	No	Dirección donde esta guardada la foto.
descripción	varchar(255)	Si	Descripción de la foto.
persona_id	varchar(15)	No	Llave foránea
categoría_id	integer	No	Llave foránea.

Tabla 76 Descripción de la tabla foto.

Nombre: categoria			
Descripción: Guarda la información referente a las categorías de las fotos.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id	unteger	No	Llave primaria.
categoria	varchar(100)	No	Categoría de la foto.
descripcion	varchar(255)	No	Breve descripción.

Tabla 77 Descripción de la tabla categoria.

Nombre: usuario			
Descripción: Guarda la información referente a los usuarios.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id	integer	No	Llave primaria.
usuario	varchar(100)	No	Nombre de usuario
rol_id	integer	No	Llave foránea.

Tabla 78 Descripción de la tabla usuario.

Nombre: rol			
Descripción: Guarda la información referente a los roles.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id	integer	No	Llave primaria.
rol	varchar(100)	No	Nombre del rol.
descripcion	varchar(255)	Si	Breve descripción.

Tabla 79 Descripción de la tabla rol.

Nombre: persona			
Descripción: Guarda la información referente a las personas de la UCI.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Expediente	varchar(10)	No	Llave primaria.

Tabla 80 Descripción de la tabla persona.

2.13 Conclusiones

En los diseños de las bases de datos realizados se da cumplimiento a los objetivos del trabajo, permitiendo el almacenamiento de las informaciones vitales de los sistemas de Gestión de Fotos, Portal de Seguridad y Protección, y del SCAPV. Se eliminaron las deficiencias de la anterior base de datos del Sistema de Control de Acceso de Personas, y se agregaron nuevas tablas dándole cumplimiento a los nuevos requisitos.

Capítulo 3

CAPÍTULO 3: Validación del Diseño Realizado.

3.1 Introducción.

El presente capítulo tiene como objetivo la validación teórica y funcional del diseño de las bases de datos realizado. En la primera se recogen aspectos como la integridad, seguridad y normalización de las bases de datos, así como el análisis de la redundancia de la información y la trazabilidad de las acciones. La segunda se basa en las pruebas realizadas a las base de datos para comprobar su funcionamiento.

3.2 Validación teórica del diseño.

Al diseñar una base de datos aún no se completa el trabajo, es muy importante tener en cuenta otros aspectos con el objetivo de mantener la consistencia, integridad y seguridad de los datos almacenados.

Para lograr esto, fue necesario tomar algunas medidas para colaborar con las del SGBD.

3.2.1 Integridad.

Una condición o limitación que se aplica a un conjunto particular de datos usualmente se denomina control de integridad o restricción. Los controles de integridad se diseñan para minimizar las inconsistencias de los datos ocasionadas cuando los usuarios o los programas de aplicaciones cometen errores en la entrada de datos.

Al imponer restricciones semánticas sobre los datos, confiamos razonablemente en el contenido de las bases de datos es correcto y que no existen incongruencias entre los datos.

La integridad de los datos puede verse afectada añadiendo datos no válidos, modificando datos existentes, que se les asigne un valor incorrecto o eliminando datos que violen alguna regla. La integridad de datos presenta varias restricciones que posibilitan la declaración y

comprobación de condiciones para expresar la consistencia, corrección y exactitud de datos almacenados, estas son: Integridad de dominio, de clave, de entidad, referencial, entre otras.

Integridad de dominio: Se restringen los valores que puede tomar un atributo respecto a su dominio, en el caso de la base de datos para el SCAPV se definió que el atributo CI sólo acepta cadenas de longitud 11 su dominio sea un varchar(11). El tipo de datos varchar almacena cadenas de caracteres que incluye letras y números, en el caso anterior fue necesario validar que el atributo CI contuviera sólo números, para realizar esta validación se usó los "obligadores" (constraints) de PostgreSQL, que se encargan de chequear (**CHECK**) determinada condición que deben cumplir los atributos de una tabla, en caso de no cumplirse la condición que valida se lanza una excepción.

Estas funciones pueden ser programadas antes o después de un evento **INSERT, DELETE** o **UPDATE** de un atributo en determinada tabla**.** En este caso se muestra el ejemplo para la validación mediante los "obligadores" específicamente un **CHECK** para que el CI sólo acepte números.

(ci**)::text ~ similar_escape** ('%(1|2|3|4|5|6|7|8|9|0)'**::text, NULL::text)**

Este **CHECK** se usó en la tabla, Tb_dPersona_X
Otra restricción en el campo chapa de los autos que tienen como dominio un varchar (6), que permite ahorrar el espacio ocupado en disco por la base de datos, lo que garantiza una mayor eficiencia en la base de datos.

Integridad de clave: La clave no puede tomar valores repetidos.

Integridad de entidad: la clave primaria de una entidad no puede tener valores nulos y siempre deberá ser única. Un ejemplo de las buenas prácticas lo constituye la elección del campo de la tabla que será o formará parte de la llave primaria, partiendo de la obligación de que todos los valores del atributo son únicos y nunca nulos (**NULL**).
Integridad referencial: Las claves ajenas de una tabla hija se tienen que corresponder con la clave primaria de la tabla padre con la que se relaciona. Por ejemplo, en la tabla

Tb_hAcceso_Persona se necesita el Id_Expediente (llave primaria) de la tabla Tb_dPersona para poder insertar un nuevo acceso de persona, ya que Id_Expediente es una llave primaria foránea de la tabla *Tb_hAcceso_Persona*, por lo cual es necesario validar la existencia de la persona y del acceso, antes de insertar o actualizar alguna **tupla** en la tabla *Tb_hAcceso_Persona*.

Las ventajas de la integridad referencial es que evita que los usuarios:[14]

- Añadan registros a una tabla relacional si no existe un registro asociado en la tabla principal.
- Cambien valores de una tabla principal cuando existen registros relacionados en al tabla relacionada.
- Eliminen registros de una tabla principal si hay registros relacionados en la tabla relacionada.

Chequeo de validez: Cuando se crea una tabla cada columna tiene un tipo de datos y el SGBD asegura que solamente los datos del tipo especificado sean ingresados en la tabla. Aunque se puede hacer chequeo de validez, es el caso del número del carné de identidad, mencionado anteriormente.

Datos requeridos: Define que en una tabla al insertar una tupla no exista una columna determinada con valor nulo. Se define efectuando la declaración de una columna en **NOT NULL**.
Ejemplo en la tabla Tb_dVisita la Fecha_Entrada es un dato requerido, por lo que se declara **NOT NULL.**

Integridad entre varias entidades de una base de datos.

Una manera de mantener la integridad es el control de concurrencia en la base de datos. A diferencia de la mayoría de otros sistemas de bases de datos que usan bloqueos para el control de concurrencia, PostgreSQL mantiene la consistencia de los datos en un modelo

multiversión. Esto significa que mientras se consulta una base de datos, cada transacción ve una imagen de los datos (una versión de la base de datos) como si fuera tiempo atrás, sin tener en cuenta el estado actual de los datos que hay por debajo. Evitando que la transacción vea datos inconsistentes que pueden ser causados por la actualización de otra transacción concurrente en la misma fila de datos, proporcionando aislamiento transaccional para cada sesión de la base de datos.

El estándar ANSI/ISO SQL define cuatro niveles de aislamiento transaccional en función de tres hechos que deben ser tenidos en cuenta entre transacciones concurrentes. Estos hechos no deseados son:

- Lecturas "sucias"

Una transacción lee datos escritos por una transacción no esperada, no cursada.

- Lecturas no repetibles

Una transacción vuelve a leer datos que previamente había leído y encuentra que han sido modificados por una transacción cursada.

- Lectura "fantasma"

Una transacción vuelve a ejecutar una consulta, devolviendo un conjunto de filas que satisfacen una condición de búsqueda y encuentra que otras filas que cumplen la condición han sido insertadas por otra transacción cursada.

Atributo	Lectura sucia	Lectura no repetible	Lectura fantasma
Lectura con cursada	Posible	Posible	Posible
Lectura cursada	No posible	Posible	Posible
Lectura repetible	No posible	No posible	Posible
Serializables	No posible	No posible	No posible

Tabla 81 Niveles de aislamiento transaccional.

La serialización proporciona el nivel más alto de aislamiento transaccional. Cuando una transacción está en el nivel serializable, la consulta sólo ve los datos cursados antes de que la transacción comience y nunca ve ni datos sucios ni los cambios de transacciones concurrentes cursados durante la ejecución de la transacción. Por lo tanto, este nivel emula la

ejecución de transacciones en serie, como si las transacciones fueran ejecutadas un detrás de otra, en serie, en lugar de concurrentemente.[8]

En este trabajo se emplea el modo serializable, para evitar que cuando haya más de un usuario trabajando en la base de datos sobre los mismos datos se produzcan hechos no deseados entre las transacciones, que puedan afectar la integridad de los datos.

DB4o permite de manera similar el modo serializable, sus características están descritas en el Capítulo I.

3.2.2 Normalización de la Base de Datos.

La normalización es un proceso en el cual se va comprobando el cumplimiento de una serie de reglas, o restricciones, por parte de un esquema de relación; cada regla que se cumple aumenta el grado de normalización del esquema de relación; si una regla no se cumple, el esquema de relación se debe descomponer en varios esquemas de relación que sí la cumplan por separado.

Existen un grupo de anomalías que influyen en la integridad de los datos, ellas son:

- Anomalías de actualización: Inconsistencia de los datos como resultado de datos redundantes y actualizaciones parciales.
- Anomalías de borrado: Perdida no intencionada de datos, debido a que se han borrado otros datos.
- Anomalías de inserción: Imposibilidad de adicionar datos en la base de datos, debido a la ausencia de otros datos.

El proceso de la normalización implica el determinarse de qué datos se deben almacenar en cada tabla de la base de datos, además trabajar con los pasos bien definidos, llamados formas normales.

Existen varios niveles de normalización:

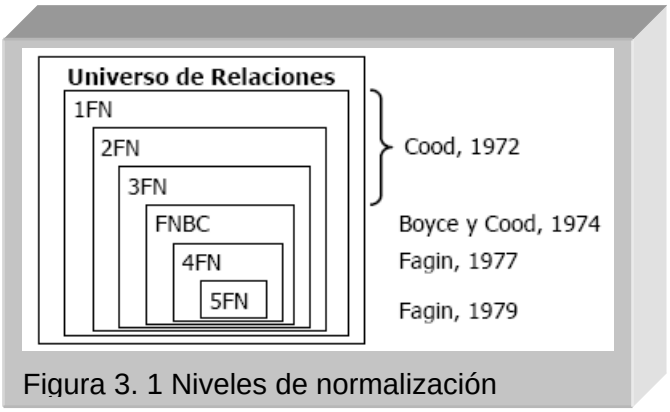

Figura 3. 1 Niveles de normalización

- Primera Forma Normal (1NF).
- Segunda Forma Normal (2NF).
- Tercera Forma Normal (3NF).
- Forma Normal Boyce-Codd.
- Cuarta Forma Normal.
- Quinta Forma Normal o Forma Normal de Proyección-Unión.

Cada nuevo nivel o forma nos acerca más a hacer una base de datos verdaderamente relacional. Cada una tiene sus propias reglas. Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización. No siempre es una buena idea tener una base de datos conformada en el nivel más alto de normalización.

Primera Forma Normal (1FN).

Una relación está en primera forma normal (1FN) si los valores en la relación son atómicos para cada atributo en la relación. Esto quiere decir simplemente que los valores de los atributos no pueden ser un conjunto de valores o un grupo repetitivo. La definición de Codd de una relación incluye la condición de que la relación esté en primera forma normal.

Las próximas dos formas normales, segunda forma normal y tercera forma normal, se aplicarán a las relaciones que estén restringidas por dependencias funcionales.

Segunda Forma Normal (2FN).

Una relación está en segunda forma normal (2FN) si el atributo no clave no es funcionalmente dependiente de una parte de la clave. Por tanto, la 2FN puede violarse sólo cuando una clave sea una clave compuesta o, en otras palabras, se conste con más de un atributo.

En situaciones como esta, se descompone dicha relación por otras más pequeñas llamadas proyecciones. Esta proyección selecciona ciertos atributos de una relación existente y los representa como una nueva relación.

Como resultado se elimina la redundancia en la información y la posibilidad de anomalías.

Tercera Forma Normal (3FN).

Un esquema de relación está en tercera forma normal (3FN) si, y sólo si, está en segunda forma normal (2FN) y, además cada atributo del esquema de relación que no está en la clave primaria sólo depende funcionalmente de la clave primaria, y no de ningún otro atributo.

Cuando las tablas están en la Tercera Forma Normal se previenen errores de lógica cuando se insertan o eliminan registros. Cada columna en una tabla está identificada de manera única por la llave primaria, y no debe haber datos repetidos.

En este trabajo se llevaron las bases de datos hasta la 3FN.

3.2.3 Análisis de redundancia de información.

La redundancia de los datos es información duplicada en una base de datos, lo que trae consigo que las consultas no se realicen en un tiempo óptimo, se ocupa mayor espacio en disco.

Una dificultad en que se cae frecuentemente, es que en ocasiones se tiene un mismo dato duplicado en la base de datos en varias tablas, algunas veces con un nombre diferente o con otro tamaño, pero en fin resulta ser que representa un mismo atributo. Esta redundancia aumenta los gastos de administración para el mantenimiento y el almacenamiento, así como el riesgo de inconsistencia entre las diversas versiones de los datos comunes. En estas bases de datos se solucionan estos problemas, creando clases que agrupen los atributos en común para otras clases. Sería así:

Figura 3. 2 Relación de herencia.

Con el diseño de la nueva base de datos remota para el SCAPV se ha disminuido considerablemente la redundancia en los datos que existía en la base de datos anterior. Un aspecto importante fue realizar el proceso de normalización de la base de datos, logrando tener sólo los datos necesarios.

Por otra parte el diseño de la base de datos del Sistema de Gestión de Fotos, ha permitido que las fotos no estén duplicadas, además que se puedan localizar fácilmente en el directorio donde se guardan.

3.2.4 Análisis de la seguridad de la base de datos.

La información que almacena una base de datos, está en constante riesgo de sufrir ataques que puedan provocar su modificación o pérdida, por ello es de vital importancia velar la seguridad de la misma, protegiéndola, en un primer momento, contra accesos no autorizados o cualquier acción que puedan violar la integridad de los datos o la confidencialidad de los mismos. En un segundo momento es necesario proteger los datos que se almacenan en la base de datos, para lo cual se deben realizar salvas de la misma.

Los datos guardados en una base de datos deben estar protegidos contra los accesos no autorizados y ante destrucción y/o alteración malintencionadas.

Los usuarios pueden tener varios tipos de privilegios o autorización de acceso para diferentes usos de la base de datos:

- Autorización de Lectura (**SELECT**): permite la lectura de datos pero no la modificación de datos existentes.
- Autorización de Inserción (**INSERT**): permite la inserción de nuevos datos pero no la modificación de datos existentes.
- Autorización de Actualización (**UPDATE**): permite la modificación de datos pero no su borrado.
- Autorización de Borrado (**DELETE**): Permite el borrado de datos.

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

- El nivel 0 es el que se encarga sobre las máquinas (host) y los usuarios. Es decir es el permite configurar que máquina/s y/o usuario/s se pueden conectar a la Base de Datos. Utilizando las opciones del fichero de configuración pg_hba.conf.
- El nivel 1 es el que se encarga de los usuarios y las Bases de Datos. Es el que permite configurar a que Bases de Datos se pueden conectar. Utilizando las opciones del fichero de configuración pg_ident.conf.
- El nivel 2 es el que se encarga de las tablas. Es el que permite configurar a que tablas pueden acceder. Utilizando los comandos **GRANT** para dar permisos y **REVOKE** para eliminar los permisos.

Ejemplo de la configuración del archivo **pg_hba.conf.**

Este método deniega la conexión de cierto rango de direcciones Ip.				
TYPE	DATABASE	USER	CIDR-ADDRESS	METHOD
host	all	all	10.33.3.3 10.33.3.254	reject

Tabla 82 Configuración del archivo pg_hba.conf. Denegar conexiones.

Este método permite la conexión de cierto rango de direcciones Ip.				
TYPE	DATABASE	USER	CIDR-ADDRESS	METHOD
host	all	all	10.33.3.3 10.33.3.254	md5

Tabla 83 Configuración del archivo pg_hba.conf. Permitir conexiones.

En la base de datos local del SCAPV es necesario tomar algunas medidas de seguridad con el fin de evitar posibles acciones que vayan en contra de la integridad y disponibilidad de los datos.
Como seguridad física se implementará la encriptación de los datos.

Autentificación de Usuarios.

El acceso a la base de datos comúnmente requiere la autorización y autenticación del usuario. Para la autenticación del usuario, el primer nivel de seguridad establece que la persona que solicita la entrada a la base de datos es un usuario autorizado.

Autentificación es el proceso mediante el cual el servidor de la base de datos y el postmaster se aseguran de que el usuario que está solicitando acceso a la base de datos es en realidad quien dice ser. Todos los usuarios que quieren utilizar postgres se comprueban en la tabla pg_user para asegurarse que están autorizados a hacerlo. Actualmente, la verificación de la identidad del usuario se realiza de distintas formas:[15]

Desde la shell del usuario.

Un demonio que se lanza desde la shell del usuario anota el id original del usuario antes de realizar un setuid al id del usuario Postgres. El id original del usuario se emplea como base para todo tipo de comprobaciones.

Desde la red.

Si Postgres se instala como distribuido, el acceso al puerto TCP del postmaster está disponible para todo el mundo. El ABD configura el fichero **pg_hba.conf** situado en el directorio **PG_DATA** especificando el sistema de autentificación a utilizar en base al equipo que realiza la conexión y la base de datos a la que se conecta. Por supuesto la autentificación basada en equipos no es perfecta incluso en los sistemas Unix. Es posible, para determinados intrusos, enmascarar el equipo de origen. Estos temas de seguridad están fuera del alcance de Postgres.

Medidas tomadas.

Las aplicaciones nunca deben conectarse a la base de datos con el usuario que es propietario de la misma o un súper-usuario, ya que estos usuarios pueden, ejecutar cualquier consulta a su antojo, modificando el esquema (por ejemplo eliminando tablas) o borrando su contenido.

Teniendo en cuenta lo expuesto anteriormente se crearon usuarios para cada base de datos, con permisos sobre determinados objetos y determinadas acciones. Estos usuarios serán los usados en la capa de persistencia de las aplicaciones para la conexión a las bases de datos.

Garantizándose así el menor daño posible a las bases de datos en caso de que intruso se apodere de una de estos usuarios y contraseñas.

Se tomó además otra medida importante, la creación de una tabla Tb_hAuditoria en la base de datos del Portal de Seguridad y Protección con el objetivo de almacenar las acciones realizadas por los usuarios sobre determinados datos, se registra la acción realizada, el responsable, la tabla sobre la que realizó la acción, la fecha y la hora de cualquier modificación en los datos almacenados.

En las bases de datos del Sistema de Gestión de Fotos y en el Portal de Seguridad y Protección se crearon además las tablas Usuario y Rol que permiten controlar el usuario que tiene acceso a los distintos módulos del sistema, en dependencias del rol al que pertenece.

Para validar que no exista pérdida de información en caso de corrupción de los datos, PostgreSQL, permite la realización de salvas (backups) mediante el comando pg_dump el cual indica al SGBD que se va a realizar la salva de la base de datos, esta puede realizarse de forma automática o dejarlos a elección del cliente, además el programa usado para administrar estas bases de datos (pgAdmin III) posee una opción (Resguardo) que brinda la misma posibilidad.

El resguardo de la base de datos se puede realizar en caliente, es decir, con el servicio Postgres corriendo, lo cual constituye una ventaja, además se puede realizar a una o varias base de datos, además mantiene compatibilidad entre versiones, sus desventajas están

marcadas por la lentitud que alcanza con bases de datos de gran volumen y el consumo de recursos del sistema.

En caso de desear restaurar una salva para su análisis o uso, se utiliza el comando pg_restore indicándole la salva que deseamos restaurar, igualmente pgAdmin III posee la opción restaurar, brindando una mayor facilidad de eso.

La base de datos local del Sistema de Control de Acceso estará sincronizando sus datos cada ciertos períodos de tiempo, acción que posibilitará que se tenga actualizado el sistema. Esta base de datos podrá además replicar sus datos hacia otra localización física

3.2.5 Trazabilidad de la acciones.

Es la capacidad que tiene una organización o sistema para rastrear, reconstruir o establecer relaciones entre objetos monitoreados, para identificar y analizar situaciones específicas o generales en los mismos. La trazabilidad de las acciones es una de las medidas más importantes para el aseguramiento de las bases de datos.[16]

Esta medida es importante debido a que permite detectar acciones indebidas y detenerlas antes de que influyan negativamente en la integridad de la información almacenada en las bases de datos.

Para lograr la trazabilidad de las acciones de los usuarios cuando interactúan con las bases de datos, se creo una tabla historial Tb_hAuditoria en la base de datos del Portal de seguridad y Protección, en la cual se almacena un historial ante eventos DML (**UPDATE, INSERT, DELETE**), que son las que de una manera u otra pueden alterar la integridad y disponibilidad de los datos. Se guarda el usuario, la fecha y hora en que se llevó a cabo la acción, además el nombre de la tabla sobre la que se realizó la acción.

No se debe confiar en los privilegios y roles asignado a los usuarios, es imprescindible mantener un control estricto de las acciones que realizan dentro de la base de datos.

Mantener las **trazas de auditoría** es otra medida tomada en las bases de datos realizadas, con el objetivo de aumentar la trazabilidad de las acciones.

Postgres garantiza las trazas de auditoría configurando esta opción en el archivo **postgresql.conf.**

Para el registro de las trazas de auditoría de las acciones realizadas sobre la base de datos del Sistema de Gestión de Fotos se usará **sfPropelAuditPlugin**.

sfPropelAuditPlugin es un **plugin** de Symfony que proporciona la funcionalidad de auditoría para los objetos de Propel. Esto significa que puede mantener una pista de auditoría de todo lo que está sucediendo en la base de datos, acciones de inserción, actualización y eliminación.

3.3 Validación funcional.

Con el objetivo de que las bases de datos cumplan con los requisitos y validar el trabajo realizado, es preciso realizarles algunos procedimientos de pruebas.

- Es necesario cargar las bases de datos sin violar la integridad de los datos.
- Realizar algunas consultas de manera que sirvan para medir su comportamiento con otra similares.

3.3.1 Búsqueda o diseño de herramientas para pruebas de carga intensiva.

Para el llenado voluminoso de bases de datos existen varias herramientas. En ocasiones se utiliza scripts para el llenado. En este trabajo se utilizó la herramienta EMS Data Generador For PostgreSQL 2005 para el llenado de las bases de datos.

El tiempo de llenado de las bases de datos fue:

- Sistema de Control de Acceso de Personas, Vehículos y Visitantes, duración 45 min.
- Portal de Seguridad y Protección 20 min.
- Sistema de Gestión de Fotos, duración 5 min.

Una de las consultas más usadas en el SCAPV es para saber el estado actual de una determinada persona, visita, o auto. A continuación se muestra como se obtiene esta información de las personas de la UCI.

Tiempo de respuesta de la consulta en 19000 tuplas de la tabla persona: 0.19 segundos.

SELECT
public.tb_dpersona.id_expediente
FROM
public.tb_dpersona
WHERE
(public.tb_dpersona.id_expediente = ‘50317') AND
(public.tb_dpersona.fecha_ult_acceso
BETWEEN ('2008-06-05 00:00:0':: **timestamp AND** ('2008-06-05 24:00:0')::**timestamp**)

3.4 Conclusiones.

Con la validación teórica de los diseños realizados se tuvieron en cuenta aspectos muy importantes para lograr un correcto funcionamiento de las bases de datos. Se impusieron algunas restricciones de integridad para garantizar que los datos introducidos sean válidos. Además se tomaron medidas para garantizar la seguridad de los datos, entre ellas la trazabilidad de las acciones realizadas por los usuarios en las bases de datos, con el objetivo de detectar cualquier acción indebida sobre la base de datos y poder tomar una correcta toma de decisiones. Mediante el proceso de normalización se redujeron las posibilidades de anomalías que influyen igualmente en la integridad de los datos.

Conclusiones.

Se ha resuelto el problema planteado anteriormente, se eliminaron las deficiencias en la anterior base de datos.

Los diseños de bases de datos realizados permiten la gestión y almacenamiento de las informaciones del SCAPV, Sistema de Gestión de Fotos y del Portal de seguridad y Protección.

Permitirán incrementar la seguridad en el centro, respondiendo a las nuevas necesidades del mismo.

Las herramientas empleadas para la realización de este trabajo permitirán que sea empleado en cualquier entorno de trabajo, sin importar el sistema operativo que se use.

Recomendaciones.

Este trabajo es sólo un comienzo, es necesario seguir algunas recomendaciones para conocer el alcance que tiene el mismo.

- Se recomienda darle seguimiento a este trabajo. Integrarle las funcionalidades necesarias con el objetivo de fortalecer la seguridad del centro.
-
- Realizar un estudio sobre los servicios que pudieran brindar este trabajo.
-
- Hacerlo extensivo a todas las áreas del centro.

Referencias bibliográficas.

1. Gil, F.A., Javier;Rosario, Javier Do *Sistema de Gestión de base de datos SGBD / DBMS.* 2005 [cited 2008 1 de Junio 2008]; Available from: http://alfa.facyt.uc.edu.ve/computacion/pensum/cs0347/download/exposiciones/1/SGBD.pdf.

2. John, W.D., Joshua *PostgresSQL Practico*. 2004 [cited 2008 23 de Mayo]; Available from: http://www.sobl.org/traducciones/practical-postgres/node19.html.

3. Pérez Zurita, P.O.C., Tiuska Lilia *Sistema de Control de Acceso a la Universidad de las Ciencias Informáticas.* 2006, (UCI , ISPJAE): Ciudad de La Habana.

4. Hernández González, A. *Un Método para el Diseño de la Base de Datos a partir del Modelo Orientado a Objetos.* 2003 [cited 2007 20 de Septiembre]; Available from: http://www.ejournal.unam.mx/cys/vol07-04/CYS07402.pdf.

5. Camacho, L.C., Gonzalo; Fernandez , Marcos; Fuentes, Gerber. *db4o Bases de datos orientadas a objetos*. 2007 [cited 2008 12 de Abril]; Available from: http://pgi.umsa.bo/enlaces/investigacion/pdf/INGSW3_70.pdf?PHPSESSID=46116be0be58bf4097bf49eee2bf40e4.

6. C.J.Date, *Introducción a los Sistemas de Base de Datos*. Vol. 1. 2006, La Habana.

7. Alberca Manzaneque, A.D.T., Jesús Galvez. *Bases de datos Orientadas a Objetos y Bases de Datos Objeto-Relacionales*. 2007 [cited 20 de Mayo 2008]; Available from: http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r23897.PDF.

8. Cardero Tasé, L.B.Z., Daliagna, *Diseño e implementación de la Base de Datos del Sistema Informático de los Tribunales Militares de Región.* 2007, UCI.

9. JACOBSON, I., *El Proceso Unificado de Desarrollo de Software*. 2004, Ciudad de La Habana, Félix Varela. p.165-253.

10. James Rumbaugh, I.J., y Grady Booch., *Unified Modeling Language Reference Manual, The (The Addison-Wesley Object Technology Series) Addison-Wesley Professional,*. 2da Edición ed. 2004.

11. Domínguez, V., Arodys E; Miranda Gutiérrez, Duniel, *Sistema de Manejo de Datos de Ensayos Clínicos: Diseño e implementación de la Base de Datos.* 2007, UCI.

12. Erich Gamma, R.H., Ralph Johnson, y John Vlissides. , *Design Patterns:Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series). Addison-Wesley Professional*. 1ra ed. 1995.

13. Nock., C., *Data Access Patterns: Database Interactions in Object-Oriented Applications* 1era Edición ed. 2003.: Addison Wesley Professional.

14. Arora, G.A., Balasubramaniam; Pandey Nitin, *Programación C#*. Vol. 1. 2007: Féliz Varela. 784.

15. Tonet, C., *Propuesta e implementación del diseño de una base de datos geográfica relacionada con la plataforma de servicios postales*. 2007, UCI.

16. Jeimy J. Cano, P.D., CFA, CFE. *Trazabilidad de las Operaciones Electrónicas. Un Reto para la Gerencia de Tecnologías de Información.* 2005 [cited 6; Available from: http://www.isaca.org/Template.cfm?Section=Home&Template=/ContentManagement/ContentDisplay.cfm&ContentID=28013.

Bibliografía

1. Hibernate. [En línea] [Citado el: 15 de Mayo de 2008.] http://www.hibernate.org

2. DB Visual Architect. [En línea] [Citado el: 20 de Mayo de 2008.] http://personales.ya.com/javierhz/mactec.htm

3. Dirección de Informatización. Arquitectura para los Sistemas que Conforman la Intranet Universitaria. [En línea] 9 de Mayo de 2007. [Citado el: 25 de Noviembre de 2007.] http://uddi.uci.cu/files/arquitectura.2007.5.9.pdf

4. db4objects. [En línea] [Citado el: 2 de Junio de 2008.] http://www.db4o.com/about/news/newsletter/2007_11.aspx

5. Guia del Administrador de PostgreSQL. [En línea] [Citado el: 20 de Febrero de 2008.] https://www.unoweb-s.uji.es/IS24/lista2/theList/guia_administrador.pdf

6. *NetPecos.* [En línea] [Citado el: 10 de Noviembre de 2007.] http://www.netpecos.org/docs/mysql_postgres/x15.html

7. Java Enterprise Edition (Java EE). [En línea] [Citado el: 12 de Enero de 2008.] http://java.sun.com/javaee

8. PHP 5 . [En línea] [Citado el: 15 de Mayo de 2008.] http://www.php.net/.

9. *PostgreSQL Global Development Group PostgreSQL.* [En línea] [Citado el: 25 de 1 de 2008.] http://www.postgresql.org

10. Seguridad de Bases de Datos. [En línea] [Citado el: 24 de Marzo de 2008.] http://gva1.dec.usc.es/~antonio/docencia/2005bd/teoria/PDSeguridad_gris.pdf.

11. Sitio oficial de SQLite. [En línea] [Citado el: 16 de Diciembre de 2007.] http://www.sqlite.org/

12. Symfony la guía definitiva. [En línea] [Citado el: 15 de Mayo de 2008.] http://www.librosweb.es

Glosario de términos.

SGBD: Sistema gestor de Base de Datos.

DBMS: DataBase Management System (Sistema de Gestión de Bases de Datos).

CASE: Computer-Aided Software Engineering.

Traza de Auditoria: Registro histórico de todos los cambios (inserciones, borrados o actualizaciones) de la base de datos, junto con la información sobre el usuario que realizó el cambio y en qué momento.

DER: Diagrama de Entidad Relación.

DCP: Diagrama de Clases Persistentes.

SQL: Structured Query Language (Lenguaje Estructurado de Consultas).

ORM: Object-Relational Mapping. Mapeador Objeto- Relacional.

PDA: Personal Digital Assistant (Asistente Digital Personal).

BDOR: Base de Datos Objeto Relacional.

BDOO: Base de Datos Orientadas a Objetos.

Tupla: Fila de una tabla de la base de datos, donde se almacena información de un determinado objeto.

MVC: Modelo Vista Controlador. Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

DML: Un Lenguaje de Manipulación de Datos (Data Manipulation Language).

DBM: Manejador de Bases de Datos (DBM por sus siglas en inglés) facilita las funciones de: almacenar físicamente, garantizar consistencia, garantizar integridad, atomicidad transaccional y manejar vistas a la información.

Propiedades ACID Atomicity, Consistency, Isolation, Durability. Conjunto de propiedades que garantizan que las transacciones de una base de datos sean procesadas de forma confiable.

RUP: Rational Unified Process (Proceso Unificado de Desarrollo)z es la metodología de IBM Rational para el desarrollo y construcción de software basado íntegramente en UML como soporte a la metodología.

Plugin: Aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.