Universidad de las Ciencias Informáticas

Facultad 1

Título: Diseño de una base de datos para controlar la información de Unión de Jóvenes Comunistas en la UCI.

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

Autor: Dayron Fernández Macias.

Tutor: Ing. Arian Abel Couso Linares

Junio del 2008

DECLARACIÓN DE AUTORÍA

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

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

_____ _____

Firma del Autor Firma del Tutor

OPINIÓN DEL TUTOR

Título: Diseño de una base de datos para controlar la información de Unión de Jóvenes Comunistas en la UCI.

Autor: Dayron Fernández Macias

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

El trabajo en los proyectos productivos es una de las actividades fundamentales de la Universidad de las Ciencias Informáticas, a estos proyectos se vinculan los estudiantes con el objetivo de ampliar sus conocimientos y demostrar el desarrollo de sus habilidades en la rama de la informática. En muchas ocasiones la actividad de los estudiantes en los proyectos productivos tributa a su trabajo de culminación de estudios. Este es el caso del diplomante que desarrolló el presente trabajo, el cual demostró una total entrega y abnegación en su confección.

El autor, estudiante de magníficos resultados académicos en la Facultad #1, mostró en todo momento, independencia, originalidad, creatividad, excesiva laboriosidad, una adecuada redacción, buena ortografía, que nos obligaba a capacitarnos cada día más en los conocimientos de Ingeniería de Software, que en ocasiones implicaba tener reunido a un grupo de expertos para poder dar respuesta a sus exigencias, mucha responsabilidad, pero sobre todo mucha tenacidad, amor por lo que hace y deseos de lograr la perfección, aún sabiendo que en la etapa de su trabajo era casi imposible lograr ese grado de perfeccionamiento.

De esta excelente labor se obtuvo el diseño de la base de datos para la UJC nacional, lo que facilitará la futura utilización de la misma y ayudará en gran medida al proyecto productivo del que hoy es parte este estudiante y del que surgió la necesidad del presente trabajo.

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.

A los ___ del mes de Junio del 2008.

_____

Firma del Tutor

AGRADECIMIENTOS

Agradezco a todas aquellas personas que hicieron posible que el trabajo se realizara de forma exitosa.

Agradezco a toda mi familia por su confianza y apoyo.

Agradezco a la revolución por hacer posible el sueño de tantos compañeros y amigos.

Agradezco a mi tutor y a todo el colectivo de mi proyecto por haberme apoyado en todo momento.

Agradezco a todos aquellos que una vez preguntaron "y la tesis".

DEDICATORIA

A mis padres

por todo el apoyo y cariño que han dado

A mi hermana y a Daily por estar siempre a mi lado

A mi abuela y demás familiares por tanta bondad.

RESUMEN

En el marco del proyecto Kainos, urge la necesidad de crear una Base de Datos (BD) con el objetivo de dar respuesta a las necesidades de almacenamiento de los documentos de la Unión de Jóvenes Comunistas (UJC) de la Universidad de las Ciencias Informáticas (UCI) .En la actualidad toda la información que se gestiona en esta organización es archivada en formato duro y tramitada de forma manual .Mucha de las operaciones que se llevan acabo sobre los documentos de los militantes no son certificadas, pues no hay forma de validarlas con rapidez; debido al gran volumen de datos en formato duro, provocando en muchos de los casos redundancia de la información y perdida de estas. La UJC afronta problemas en cuanto al traslado de los militantes de un comité de base hacia otro , no existe la forma de verificar si el militante fue incorporado en el nuevo comité de base al ser dado de baja en el comité anterior.

Para garantizar el cumplimiento de los objetivos propuestos en el centro, la investigación propone el diseño de una Base de Datos (BD), que permita almacenar y gestionar la información generada en la UJC de la UCI. A través del diseño propuesto es posible tener un control sobre la redundancia de los datos , asegurando su consistencia y evitando de esta forma que exista expedientes repetidos. Se registrará en todo momento los cambios realizados, fecha y hora ,así como el usuario que alteró los datos del militante. Los datos recogidos tendrán protección frente a usuarios no autorizados, habrá una independencia y mantenimientos de los mismos, servicios de copias de seguridad y recuperación ante fallos. El diseño propuesto se desarrolla sobre herramientas de software libre y propietario : PostgreSQL y Visual Paradigm respectivamente.

ÍNDICE

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

Cuando en el año 2002, Rusia decide retirar la base de radioescucha "Lourdes", el Comandante en Jefe Fidel Castro Ruz, decide crear en dichas instalaciones, la Universidad de las Ciencias Informáticas (UCI), lo que sería la ciudad universitaria más grande de Cuba y una de las más grandes de América. Nacido como un proyecto de la Revolución Cubana, denominado al principio "Proyecto Futuro", con dos objetivos: informatizar el país y exportar software para el desarrollo económico del mismo.

La joven Universidad de las Ciencias Informáticas (UCI), es la casa de altos estudios que reúne mayor potencial de capital humano en la rama de la informática del país. Entre los objetivos fundamentales de la Universidad se encuentra la formación de profesionales comprometidos con la Revolución y altamente calificados como ingenieros formados a partir de un modelo de formación profesional para la producción y desde la producción.

Con el fin de mejorar la eficiencia y flexibilidad de los servicios de informatización de la Unión de Jóvenes Comunistas (UJC), surge el proyecto Kainos. En el marco de este proyecto urge la necesidad de Diseñar una Base de Datos, para almacenar la información que se genera en la UJC de la UCI, y de esta forma tener un control centralizado de los documentos emitidos por los integrantes de esta organización, favoreciendo a su correcto funcionamiento.

Actualmente el proceso de almacenamiento de la información referente a la UJC en la UCI, se plasma en papel y de forma manual, provocando dificultades en la gestión de la información en los Comités de Base, Comités Primarios de la UJC. La ausencia de una Base de Datos para almacenar la información necesaria para esta Organización da lugar al siguiente **problema de Investigación**.

¿Qué diseño de Base de Datos proponer para la aplicación encargada de gestionar la información de la UJC en la UCI?

Como **objetivo general** de esta investigación: Proponer el Diseño una Base de Datos para almacenar la información que se genera en la UJC de la UCI.

A partir del objetivo general, se enuncian los siguientes **objetivos específicos**:

1. Estudiar los gestores de base de datos (software libre).
2. Definir herramientas para la modelación de las bases de datos.

3. Definir gestor de base de datos a utilizar.
4. Proponer un diseño óptimo y eficiente.

Objeto de Estudio

Gestión automatizada de la información mediante BD relacionales.

Campo de Acción

Gestión automatizada de la información en la UJC de la UCI.

Preguntas Científicas

¿En qué se almacena la Información en la actualidad?

¿Se almacena toda la información que la UJC necesita?

¿Cuenta la UJC con alguna Base de Datos para guardar la información?

¿Existe alguna propuesta de diseño?

Marco Conceptual

Almacenamiento de la información.

Para lograr el cumplimiento de los objetivos específicos planteados anteriormente se proponen las siguientes **tareas de la investigación**.

1. Estudio de otros sistemas similares y empresas que se dediquen al análisis y diseño de Bases de Datos.
2. Estudio de las diferentes herramientas que se utilizarán para el diseño de la aplicación.
3. Estudio de los requisitos funcionales y no funcionales.
4. Definición del modelo de objeto o modelo de clases persistentes de las bases de datos.
5. Diseño del diagrama Entidad Relación de la Base Datos.
6. Validación teórica y funcional del Diseño.

Población

Los jóvenes de la UCI entre 16 y 30 años.

Unidad de Estudio

Cada joven de la UCI entre 16 y 30 años.

Estrategia de Investigación

- **Investigación exploratoria**

Se llevó a cabo una Investigación exploratoria con el objetivo de conocer con más detalles la problemática que está afectando la UJC en la UCI.

Métodos científicos de investigación

Método Teórico

- ***Inductivo-Deductivo***

A través de un razonamiento lógico se es capaz de llegar a un conjunto de conocimientos previos para el posterior desarrollo de la solución.

- **Modelación**

Este método es una guía que representa de forma concisa a través de gráficos y palabras la realidad de la situación y su posible solución.

Método Empírico

- **Entrevista**

Con la ayuda de este método se hace una modelación del negocio acorde a las necesidades del cliente, además se lleva a cabo un levantamiento de requisitos óptimo en la solución de la problemática que afecta al cliente.

La Tesis está estructurada en los siguientes Capítulos.

Capítulo1: Fundamentación Teórica

En el capítulo 1 incluye un estado del arte del tema Bases de Datos a nivel internacional, nacional y de la Universidad. Se realiza una breve reseña histórica sobre el surgimiento de las Bases de Datos. Se enuncian algunos conceptos y definiciones de importancia para una mejor comprensión del tema. Se describen las herramientas empleadas para el diseño de la Base de Datos de la UJC y se propone un SGBD para su posterior creación.

Capítulo2: Descripción y análisis de la solución propuesta

En este capítulo se describe la solución propuesta, cumpliendo con los requisitos funcionales y no funcionales trazados a partir del pedido del cliente y la labor realizada por el analista. Se realiza un análisis de la integración con otros módulos y se describe la arquitectura propuesta. Se representa el diagrama de clases persistentes y se detalla cada una de las clases, así como el diagrama Entidad-Relación y la descripción de cada una de sus tablas. Incluye además, un análisis de optimización de consultas en la BD diseñada.

Capítulo3: Validación del diseño realizado

En este último capítulo se brinda información sobre la validación teórica y funcional del diseño realizado a través de la integridad, normalización, análisis de redundancia de información y seguridad de la base de datos.

CAPÍTULO #1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente Capítulo se hace una breve vista histórica del origen de las Bases de Datos, así como de las tendencias, técnicas, tecnologías y metodologías utilizadas en el mundo actual para su tratamiento; se describen las herramientas empleadas para el diseño de la Base de Datos de la UJC y se propone un SGBD para su posterior creación. Además se realiza un estudio de algunos diseños de Bases de Datos similares en la Universidad como es el caso del diseño de la BD de la FEU nacional. También se hizo un estudio a nivel nacional e internacional de algunos Diseños de Bases de Datos para la gestión de Información.

1.2 Historia y surgimiento de las Bases de Datos

Debido a la gran necesidad de almacenar y recuperar información en las distintas empresas, se crean las Bases de Datos automatizadas.

En un principio se comenzaron a utilizar archivos secuenciales como almacenes de datos, que accedían rápido a la información pero de forma secuencial, como alternativa a esta dificultad surgen los archivos indexados, que accedían directamente a la posición deseada del archivo.

Debido al creciente desarrollo de los programas y la complejidad de almacenamiento para el volumen de datos que se generaban, surgen las Bases de Datos, ya que los anteriores métodos no cubrían las necesidades que se presentaban. Se necesitaba un sistema de almacenamiento que permitiera realizar operaciones complejas sin violar las restricciones, garantizando la seguridad en el acceso de los usuarios e integridad de la información

La primera topología que surge es la jerárquica que seguía una jerarquía para la organización de los datos. Tenía como inconveniente que los accesos a los datos eran unidireccionales, lo que hacía más complejo el camino inverso. Como variante a esta topología se crearon las bases de datos de red que incorporaban la característica de que cada nodo podía tener más de un padre, pero no era aún una solución factible. Para dar absoluta libertad a las relaciones entre tablas surgieron las bases de datos relacionales que incorporaron un lenguaje común de acceso a los datos: SQL (del inglés *Structured Query Language, o Lenguaje Estructurado de Consultas*), supliendo de esta forma la dependencia de los aspectos físicos y la programación de aplicaciones con bases de datos y las propiedades ACID, que significa:

Atomicidad: Las operaciones se tratan en forma atómica, agrupadas en transacciones que se ejecutan en su totalidad o no se ejecutan; así si algo falla no debe quedar rastros de la información inconclusa en la base de datos.

Consistencia: Las transacciones deben cumplir las reglas de integridad definidas dentro de la BD que mantienen la coherencia entre los datos, de no cumplirlas se evita su ejecución.

Aislamiento: Cada transacción es tratada como una unidad de aislamiento. Se manejan posibles conflictos entre varias transacciones concurrentes, impidiendo que unas interfieran con otras y garantizando que todas se ejecuten y finalicen en el orden en que se iniciaron.

Durabilidad: Los resultados de las transacciones confirmadas son permanentes y deben sobrevivir a posibles caídas del sistema o de la Base de Datos.

Las propiedades ACID resuelven muchos conflictos ocurridos durante las transacciones, especialmente los causados por la concurrencia y la interrupción de procesos de grabación debido a errores o fallas.

Más adelante surgieron las Bases de Datos orientadas a objetos para superar algunas deficiencias de los modelos anteriores y son aplicadas generalmente en aplicaciones más complejas y sofisticadas.

1.3 Principales componentes de las Bases de Datos

Los principales elementos que componen una BD son:

Datos: Es lo que se conoce como BD propiamente dicha. Los datos serán tanto integrados, cuando se unifican varios archivos que de otra forma serían diferentes, como compartidos, cuando se comparten los datos individuales para que sean utilizados por lo usuarios en diferentes fines.

Hardware: Son los dispositivos de almacenamiento de la BD y los periféricos necesarios para su uso.

Software: Lo constituye un conjunto de programas que se conocen como Sistemas Gestores de Bases de Datos (SGBD)

Usuarios: Con las Bases de Datos se relacionan tres tipos de usuarios:

- Programador de aplicaciones: se encarga de crear programas de aplicación que utilizan la BD.
- Usuario final: es quien accede a la Base de Datos, ya sea por programas de aplicación o por un lenguaje de consultas

- Administrador de la BD (*DBA,* de sus siglas en inglés *Data Base Administrator*): se encarga del control general de la BD (1)

1.4 ¿Qué es una Base de Datos?

"Base de Datos es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquinas accesibles en tiempo real y compatible con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo. " (2)

1.5 Modelos de Bases de Datos

1.5.1 Bases de Datos Jerárquicas

En estas Bases de Datos se almacena la información en una estructura jerárquica, organizándose los datos en una forma similar a un árbol (visto al revés), en donde un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se les conoce como *hojas*. (3)

1.5.2 Bases de Datos de Red

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

1.5.3 Bases de Datos Relacional

Se basa en el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las Bases de Datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por *registros* (las filas de una tabla), que representarían las tuplas, y *campos* (las columnas de una tabla).

1.5.4 Bases de Datos Orientada a Objetos

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

- Encapsulación: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.

- Polimorfismo: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

1.5.5 Bases de Datos Documentales

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

1.5.6 Bases de Datos Deductivas

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

1.5.7 Gestión de Bases de Datos Distribuidas

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

1.6 Modelo Utilizado

Para el desarrollo de este trabajo se utiliza el modelo de Base de Datos relacional, además de ser precisamente, el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente, a través de este modelo se puede representar una eficiente redundancia de los datos, la forma en la cual se almacenan los datos, es mas fácil de entender y utilizar para un usuario esporádico de las Bases de Datos, facilitando de esta forma un mayor soporte.

1.7 Técnica de representación de esquema conceptual

Los modelos de datos convencionales descritos anteriormente no ofrecen la suficiente capacidad de abstracción, ni el poder expresivo como para captar la semántica del mundo real, haciendo difícil la comunicación del diseñador con el usuario. Entre los modelos de datos que surgen con el fin de resolver este problema se destaca:

1.7.1 Modelo entidad/interrelación (E/R)

El modelo E/R fue propuesto por Peter P. Chen entre los años 1976-1977, en sus dos artículos históricos,**" El modelo E/R puede ser usado como una base para una vista unificada de los datos",** adoptando **"el enfoque más natural del mundo real que consiste en *identidades* e**

***interrelaciones*"**. Posteriormente otros muchos autores han investigado y escrito sobre el modelo, proporcionando importantes aportaciones, por lo que realmente no se puede considerar que exista un único modelo E/R .El modelo E/R describe los datos como entidades, relaciones (vínculos) y atributos y permite representar el esquema conceptual de una Base de Datos de forma gráfica mediante los **diagramas E/R.**

1.7.2 Modelo entidad/interrelación (E/R) extendido

El modelo E/R extendido pretende aportar soluciones a requerimientos un tanto más complejos no contemplados en el modelo E/R propuesto por Chen. Así se incorporan al modelo E/R dos nuevos elementos: Superclases y Subclases.

El modelo E/R permite al diseñador concebir la Base de Datos a un nivel superior de abstracción, aislándolo de consideraciones relativas a la máquina (tanto a un nivel lógico como físico), y a los usuarios en particular (nivel externo), y centrándolo en un plano infológico, en el que la información desempeña un papel fundamental.

1.8 Arquitectura de los sistemas de BD

Para alcanzar 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.(ver figura# 1.1)

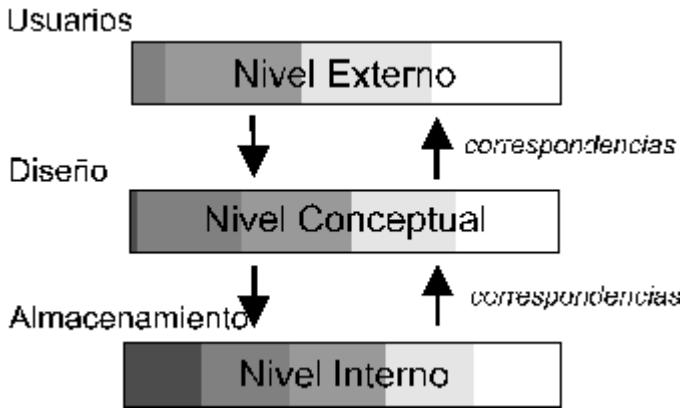
En esta arquitectura, el esquema de una Base de Datos se define en tres niveles de abstracción:

1.8.1 Nivel interno*:* Describe la estructura física de la BD 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. Es el más cercano al almacenamiento.

1.8.2 Nivel conceptual*:* 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 describe entidades, atributos, relaciones, operaciones de los usuarios y restricciones. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar el esquema.

1.8.3 Nivel externo*:* Describe varios esquemas externos o vistas de usuario. Cada uno describe la parte de la Base de Datos que interesa a un grupo determinado de usuarios y oculta a ese grupo el

resto de la Base de Datos. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar los esquemas. Es el más cercano al usuario.



Figura# 1.1 Arquitectura de tres niveles.

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la Base de Datos física.

Se pueden definir dos tipos de independencia de datos:

- ***Independencia lógica:*** Es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la Base de Datos o para reducirla.
- ***Independencia física***: Es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual o los externos.

1.9 Etapas del Diseño de BD

1.9.1 Diseño Conceptual

El modelado conceptual, también denominado ***diseño conceptual,*** constituye la primera fase de desarrollo de las Bases de Datos, y puede subdividirse en dos etapas claramente diferenciadas:

Análisis de requerimientos: Esta primera etapa, común para datos y procesos, es la etapa de percepción, identificación y descripción de los fenómenos del mundo real a analizar, es aquí donde se ha de responder a la pregunta **¿Qué representar?** (4)

Mediante el estudio de las reglas de una empresa y de entrevistas a los usuarios, se elabora un ***esquema descriptivo o percibido*** de la realidad, a través del lenguaje natural, este primer esquema, luego se irá refinando hasta llegar a un esquema más correcto: ***el esquema conceptual.***

Conceptualización: En esta segunda etapa se transforma este primer esquema descriptivo, refinándolo adecuadamente y responde a la pregunta **¿Cómo representar?**, es aquí donde se habrá de buscar una representación normalizada que se apoye en un modelo de datos que cumpla determinadas propiedades: coherencia, plenitud, no redundancia, simplicidad, etc., hasta llegar así al ***esquema conceptual***.

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

1.9.2 Diseño Lógico

El diseño lógico es la segunda fase del desarrollo de las Bases de Datos , tiene como objetivo transformar el esquema conceptual obtenido en la fase anterior, adaptándolo al modelo de datos en el que se apoya el SGBD que se va a utilizar. Es aquí donde se adquiere una estructura lógica adecuada que establezca el equilibrio entre las exigencias del usuario y la eficiencia, limando las redundancias, consiguiendo la máxima simplicidad, evitando cargas suplantarias de programación, optando por una amplia: flexibilidad, confidencialidad, integridad y tiempo de respuesta, estableciendo prioridades y adoptando compromisos.

En esta fase se pueden encontrar dos etapas:

Diseño lógico estándar: A partir del esquema conceptual resultante de la fase anterior, y teniendo en cuenta los requisitos de procesos y de entorno, se elabora un **esquema lógico estándar** (ELS), que se apoya en el modelo lógico estándar (MLS), el cual es el mismo que el modelo de datos (Jerárquico, Codasyl, Relacional), soportado por el SGBD que se vaya a utilizar, pero sin restricciones ligadas a ningún producto Comercial.

Diseño lógico específico: Una vez definido ELS, y con el modelo lógico específico (MLE), propio del SGBD (PostgresSQL, Oracle, MySQL, Interbase, etc.), se elabora el **esquema lógico específico** (ELE), que será descrito en el lenguaje de definición de datos (LDD), del producto comercial que se esté utilizando.

En esta fase se dispone además de técnicas como: la normalización que se utiliza para comprobar la validez de los esquemas lógicos basados en el esquema relacional y la técnica de grafos relacionales.

1.9.3 Diseño Físico

La última fase del desarrollo de las Bases de Datos, el diseño físico, es el proceso de producir la descripción de la implementación de la Base de Datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos. El objetivo del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

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

Cada fase es un proceso iterativo y, como tal, se van produciendo refinamientos sucesivos antes de pasar a las siguientes. Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

1.10 Diseños de BDR analizados

En la actualidad, la recogida de la información de los militantes de la UJC en la Universidad y en Cuba se plasma en papel, la carencia de una Base de Datos para la gestión de la información de la UJC hace más difícil el trabajo, y dificulta el rápido acceso a los datos debido al gran volumen de información recogida. Se realizó un estudio a nivel nacional, internacional, y en la universidad de algunos sitios de gestión de información juvenil como fuente de apoyo para el desarrollo de este trabajo.

1.10.1 Esfera Internacional

XesCampus: “Sistema de información de gestión académica”

XesCampus es una aplicación de gestión académica de la Universidad de Santiago de Compostela, España, desarrollada por Unixest. Esta aplicación web está construida en torno a una Base de Datos relacional a la que se accede mediante tecnologías estándar de Internet (HTML, XML, ASP…). Consta

de varios módulos que abordan procesos característicos de gestión académica como planificación, matrícula, expedientes, bolsas y títulos.

Este sistema fue desarrollado sobre la plataforma de Microsoft.Net, en el que se empleó tecnología como Microsoft SQL Server 2000, Microsoft Windows 2000 Advanced Server, Microsoft Internet Information Server 5.0, Microsoft ASP 3.0, Microsoft Server-SQLXML 3.0, Microsoft Office XP Web Components y Microsoft Office XP SmartTags**.**

Proyecto ALBA: "Sistema libre de gestión educativa"

El Proyecto AlBA también conocido como "Sistema Informático Abierto de Gestión Unificado para Unidades Educacionales", en la República de Argentina. AlBA fue desarrollada con herramientas robustas, confiables, flexibles y modulares que resuelven la fragmentación y redundancia innecesaria de la información, permitiendo a los responsables de los establecimientos educativos una administración más sencilla y ágil. ALBA se desarrolló en PHP5 con Symfony, utiliza un servidor de Base de Datos MySQL4.1 o superior. Es un sistema licenciado bajo GNU/GPL, estructurado mediante diferentes módulos ensamblables y su primera versión beta cubre tres grandes funciones:

- Gestión de Alumnos: elabora un legajo del alumno que articula información administrativa (los datos personales), el desempeño escolar (notas, asistencia, seguimiento docente) e información complementaria (como vacunas o exámenes médicos).
- Gestión de Docentes: contempla la información básica de los profesionales docentes, sus asistencias, horarios, materias dictadas y actividades.
- Gestión de las Unidades Educacionales: integra la información de los docentes (horarios, asignación de materias, asistencia), los alumnos (notas, asistencia, legajos) y toda la información relativa al establecimiento educativo.

1.10.2 Esfera Nacional

Sistema para el control de la plantilla de los militantes

El Sistema para el control de la plantilla de los militantes, fue la primera versión que tuvo la UJC de software de gestión a nivel nacional, en el cual gestionaba algunos de los datos que se tramitan en la UJC. Este software contaba con un intalador (setup), donde se podía escoger la ruta para su instalación.

Entre los servicios que brindaba se pueden mencionar la gestión de organizaciones de base, así como la inscripción de Comité UJC, Comité Primario y militantes. La BD contaba con nomencladores para la

creación de los Comité UJC y Primario. Se podía obtener reportes de las altas y bajas, así como la actualización de las plantillas de cada una de las organizaciones de base que sufría cambio producto a las altas y bajas. También se podía obtener de la base de datos reportes de los militantes con doble condición, de los desvinculados por organización de base y de las organizaciones de base activas.

Este sistema de gestión no estuvo activo por mucho tiempo debido a su incapacidad para realizar algunos requerimientos., además de múltiples problemas de rendimiento que fue presentando en el corto plazo de su explotación.

GESTACAD: Sistema para la gestión académica universitaria

GESTACAD es un sistema automatizado que permite mantener y actualizar toda la información referente a estudiantes y profesores de la Universidad de Matanzas, a través de una arquitectura cliente-servidor, está soportado por el SGBD Interbase FireBird y Borland Delphi como cliente, usa tecnología ActiveX Data Object (ADO) para la conexión a la base de datos remota. Este sistema brinda una herramienta de ayuda para el trabajo en los departamentos y secretarías de la universidad. Está estructurado a través de varios módulos entre los que se encuentran:

Un Módulo de Administración: para la gestión de las tablas del sistema vía Web así como agregar nuevas consultas al sitio oficial y establecer los distintos niveles de acceso a estas.

Un Módulo Web para las Secretarias Docentes: para la Gestión de Estudiantes que permite hasta el momento la realización de acciones generales comunes en una Secretaría Docente así como la obtención de reportes oficiales

Un Módulo Web para los Jefes de Departamentos docentes: donde se incluyen acciones relativas como la asignación de la carga docente y el control sobre los profesores del Dpto.

Un Módulo Web para los Profesores: donde estos pueden llevar el control docente de sus estudiantes, el control de las evaluaciones así como reportes relativos a su carga docente.

Un sitio Web con reportes en línea: con la utilidad del registro docente para los profesores además de la búsqueda de estudiantes la cual devuelve, además de algunos datos personales del estudiante, su ubicación según el horario docente detallando aula, asignatura y tipo de clases que esta recibiendo además de su estado si se ha pasado asistencia en el turno de clase.

A partir del estudio realizado sobre este sistema de gestión se llego a la conclusión de que fue necesario realizar un diseño de Bases de Datos que permitiera obtener información académicas necesarias para su posterior gestión, utilizándose para su modelación el modelo Entidad-Relación.

1.10.3 Esfera UCI

En la universidad no existe un sistema encargado de gestionar la información que necesita la UJC. En el marco del proyecto Kainos se desarrolló un sistema recientemente que gestiona la información de la FEU nacional, este sistema es similar al que se está construyendo para la UJC, consta de una Base de Datos relacional soportada por el SGDB PostgreSQL. La Base de Datos fue diseñada para almacenar toda la información referente a la Federación Estudiantil Universitaria. En ella está recogida toda la estructura de la FEU, la asignación de tareas por parte de los directivos de la organización, las diferentes formas de ingreso, así como también las brigadas estudiantiles existentes y su caracterización estadística.

1.11 Aporte

El estudio de los sistemas de gestión antes descritos sirvió de referencia para la identificación de los modelos de Bases de Datos que estos presentan, y en la adopción del modelo utilizado para la propuesta que se brinda. Se obtuvo experiencia en la forma de representar los datos en las distintas fases o etapas que conforman el modelado de las BD. Se realizó estudios de herramientas de modelación y SGDB para definir el más propicio y factible para el desarrollo de la BD de la UJC. Se adquirió conocimiento de Software libre a través del estudio de algunos de estos sistemas de gestión. La BD de FEU nacional sirvió de guía para la conformación de la estructura organizativa de la UJC .

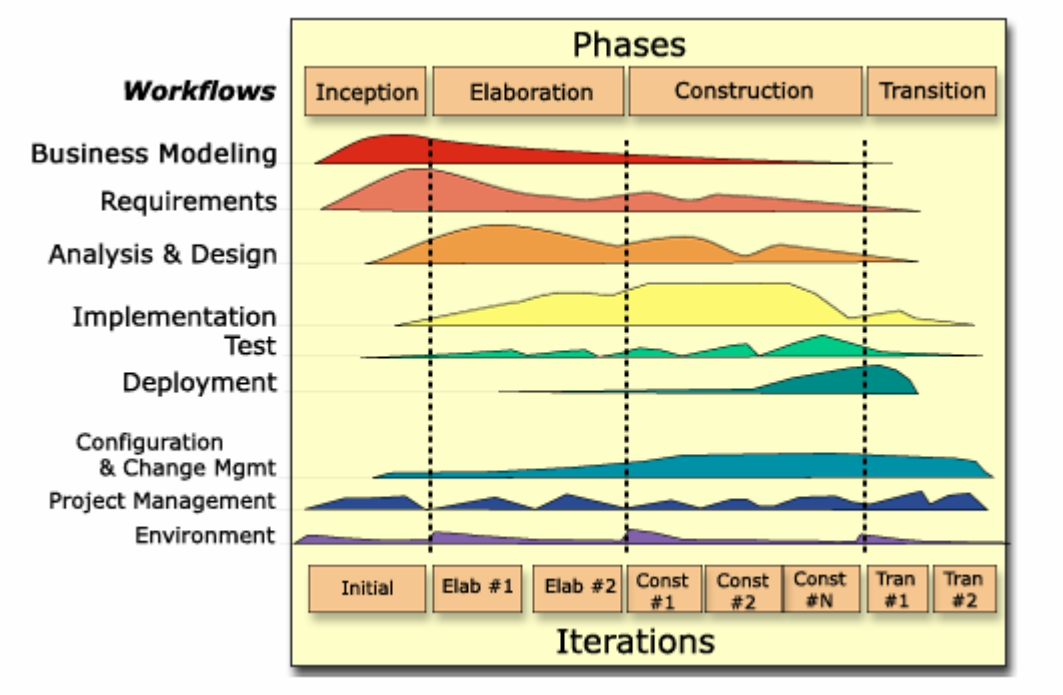
1.12 Metodología a Utilizar

Partiendo de que una metodología de desarrollo de software es un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software con la calidad requerida por el cliente. En el presente trabajo se utilizó como metodología el RUP conjuntamente con el Lenguaje Unificado de Modelado (UML).

1.12.1 RUP

El Proceso Unificado Racional (*Rational Unified Process* en inglés, habitualmente resumido como RUP), es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada una de ellos, estos a la vez se dividen en fases que finalizan con un hito donde se debe tomar una decisión importante (ver figura# 1.2).



Figura# 1.2 RUP en dos dimensiones

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo.

Características principales de RUP

- ✓ Guiado por Casos de Uso
- ✓ Iterativo e Incremental
- ✓ Centrado en la Arquitectura

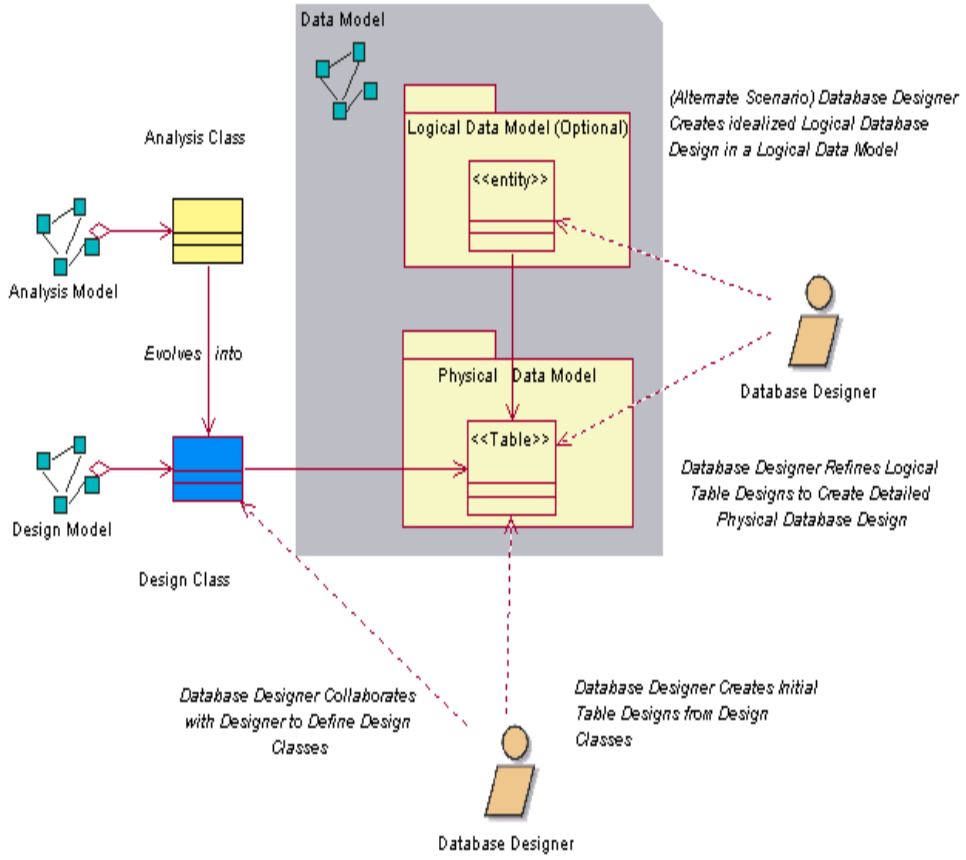
El rol de Diseñador de BD según esta metodología debe desarrollar una serie de artefactos y actividades para garantizar el cumplimiento satisfactorio de su labor, entre ellas está:

- ✓ Identificar las clases persistentes.
- ✓ Elaborar el diagrama de clases persistentes.
- ✓ Definir las tablas de referencia.

- ✓ Crear clave primaria y restricciones de integridad.
- ✓ Definir las reglas de integridad referencial y de la información.
- ✓ Normalizar el diseño de la BD para su optimización.

El Diseñador de BD debe tener sólido conocimiento sobre:

- ✓ Modelado de datos.
- ✓ Arquitectura del sistema
- ✓ Análisis y diseño de técnicas Orientado a Objetos.
- ✓ Administración de BD.
- ✓ Comprensión del entorno y lenguaje de implementación.
- ✓ El artefacto que se obtiene como resultado de las actividades desarrolladas por el Diseñador de BD es el Modelo de Datos. (ver figura1.3)



Figura# 1.3 Actividades realizadas por el diseñador de base de datos.

El modelo de datos describe las representaciones lógicas y físicas de los datos persistentes utilizados por la aplicación. Puede ser creado a partir de un conjunto de clases del diseño persistentes,

del modelo de diseño, a partir de un modelo entidad-relación creado por el diseñador de BD o mediante ingeniería inversa a partir de una BD existente.

Es necesario el modelo de datos aún cuando el mecanismo de almacenamiento de los datos persistentes no se base en la tecnología orientada a objetos. Se utiliza para definir la persistencia de las estructuras de datos y la correspondencia entre las clases de diseño y las estructuras de datos persistentes.

El Diseñador de BD es responsable de la integridad del modelo de datos, y de asegurar que los datos modelados sean correctos, coherentes y comprensibles.

UML

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software, permite la modelación de sistemas con tecnología orientada a objetos.

UML es un lenguaje para especificar y no para describir métodos o procesos. Ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de BD y componentes de software reutilizables.

UML cuenta con varios tipos de diagramas.

- ✓ Diagramas de Estructura
- ✓ Diagramas de Comportamiento
- ✓ Diagramas de Interacción

1.13 Herramientas

Para el diseño y construcción de una Base de Datos se necesita de herramientas que permitan la gestión de los procesos inherentes, control sobre la redundancia de los datos, así como su consistencia, comparación de datos por usuarios autorizados y nuevas aplicaciones que se vayan creando, la integridad de los datos, la seguridad con protección frente a usuarios no autorizados, la accesibilidad a los datos a través de SGBD, mantenimiento e independencia de los datos, concurrencia, servicios de copias de seguridad y de recuperación ante fallos, así como lograr que todo el proceso de su creación sea lo más sencillo y rápido posible. Existe una gran variedad de herramientas que se encargan del diseño y construcción de las Bases de Datos, entre ellas se encuentran:

1.13.1 Herramientas case

- **ERwin**

ERwin es una herramienta para el diseño de BD, que brinda productividad en el modelado de datos, diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la Base de Datos diseñada. Se caracteriza por su simplicidad de generar código para la mayoría de los manejadores de BD y la rapidez para el desarrollo de BD complejas, acelerar los tiempos de desarrollo. Esta herramienta ofrece una metodología para realizar diagramas entidad-relación y cuenta con una interfaz gráfica altamente intuitiva. Además ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de procedimientos almacenados y disparadores para los principales tipos de BD.

- **DBDesigne4**

DBDesigner 4 es un sistema de diseño visual de BD gratuito que integra la BD de diseño, modelado, creación y mantenimiento en un único entorno sin fisuras. Combina características profesionales y un claro y sencillo interfaz de usuario para ofrecer la forma más eficiente de manejar sus Bases de Datos. Dispone de una representación visual de las tablas y las relaciones que figuran en su proyecto. Se puede ver rápidamente los campos en una tabla o cómo cada cuadro se refiere a los demás. DBDesigner puede exportar el esquema de la base de datos en un SQL script, o conectarse directamente a un soporte de base de datos y crear allí. También puede importar de las BD ya existentes. Scripts SQL db o de generación. Por supuesto, puede guardar un proyecto en su formato original (XML) de manera que toda la información se mantiene (por ejemplo, no se puede guardar las posiciones de los cuadros en el espacio de trabajo en una SQL script). Debido a su arquitectura de plugin, DBDesigner es fácilmente extensible para trabajar con muchos servidores de Bases de Datos.

Soporta multitud de opciones, entre las que se encuentran:

- ✓ Modo de diseño o consulta.
- ✓ Ingeniería inversa de Bases de Datos MySQL, Oracle, MSSQL y cualquiera con drive ODBC.
- ✓ Control de versiones.
- ✓ Constructor de queréis.

- ✓ Soporte especial para MySQL.
- ✓ Es un producto gratuito bajo licencia GPL.

➢ **Visual Paradigm**

Visual Paradigm es una herramienta **CASE** (Computer-Aided Software Engineering) que utiliza "UML": como *lenguaje* de modelaje. Se integra con las siguientes herramientas Java:

- ✓ Eclipse/IBM WebSphere
- ✓ JBuilder
- ✓ NetBeans IDE
- ✓ Oracle JDeveloper
- ✓ BEA Weblogic

Está disponible en varias ediciones, cada una destinada según las necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Ofrece ingeniería inversa, generación de código, importación desde Rational Rose, exportación/importación XMI, generador de informes, editor de figuras, integración con MS Visio, plug-in, integración con Visual Studio. Entre sus nuevas características se incluyen el modelado colaborativo con CVS y Subversión, interoperabilidad con modelos UML2 a través de XMI, etc. Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

1.13.2 Sistema Gestor de Bases de Datos (SGBD)

➢ **PostgreSQL**

PostgresSQL es un servidor de Base de Datos relacional orientada a objetos de software libre, liberado bajo la licencia BSD. Es dirigido por una comunidad de desarrolladores y organizaciones comerciales denominada PGDG (PostgreSQL Global Development Group). PostgreSQL está basado en el proyecto POSTGRES, de la universidad de Berkeley. Es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99. Fue pionero en muchos de los conceptos del sistema objeto-relacional actual.

"Este proyecto lleva más de una década de desarrollo, siendo hoy en día, el sistema libre más avanzado con diferencia, soportando la gran mayoría de las transacciones SQL, control concurrente, teniendo a su disposición varios "language bindings" como por ejemplo C, C++, Java, Python, PHP y muchos más."

Entre sus principales características están:

- ✓ **DBMS Objeto-Relacional**

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

- ✓ **Altamente Extensible**

PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.

- ✓ **Soporte_SQL_Comprensivo**

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

- ✓ **Integridad Referencial**

PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la BD.

- ✓ **API Flexible**

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

- ✓ **Lenguajes Procedurales**

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

- ✓ **MVCC**

MVCC, o Control de Concurrencia Multi-Versión (Multi-Versión de Control de Concurrencia), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Cuando se utiliza un Database Management System(DBMS) con capacidades SQL, tal como MySQL o Access, probablemente habrá

notado que hay ocasiones en las una lectura tiene que esperar para acceder a información de la base de datos. La espera está provocada por usuarios que están escribiendo en la BD. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros.

Mediante el uso de MVCC, PostgreSQL se evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la BD. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

- ✓ **Cliente/Servidor**

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

- ✓ **Write Ahead Logging (WAL)**

La característica de PostgreSQL conocida como *Write Ahead Logging* incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual podremos restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos

- ➢ **MySQL**

Es un SGBD relacional, multihilo y multiusuario, creado por la compañía MYSQL AB, una de las más grandes empresas en el desarrollo de Software libre en el mundo y es la responsable de su desarrollo en un esquema de licenciamiento dual. MySQL en los últimos años ha tenido un crecimiento vertiginoso. Es la base de datos de código abierto más popular del mundo.

Está compuesto por un servidor SQL, varios programas clientes y bibliotecas, herramientas administrativas, y una gran variedad de interfaces de programación (APIs). Se puede obtener también como una biblioteca multihilo que se puede enlazar dentro de otras aplicaciones para obtener un producto más pequeño, más rápido, y más fácil de manejar.

Ventajas:

- ✓ Acceso a las bases de datos de forma simultánea por varios usuarios y/o aplicaciones.
- ✓ Seguridad: Determinados usuarios tendrán permiso para consulta o modificación de determinadas tablas en forma de permisos y privilegios, lo que permite compartir datos sin que peligre la integridad o protegiendo determinados contenidos.
- ✓ Potencia: Utiliza el potente lenguaje de consultas SQL.
- ✓ Escalabilidad: Es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.
- ✓ Conectividad: Permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es usual que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix.
- ✓ Recuperación automática ante fallas: Si se da de baja de forma anormal, no suele perder información ni corromper los datos y completa las transacciones que no se terminaron.
- ✓ Registros de longitud fija y variable, sin límites en el tamaño, soporta hasta 32 índices por tabla.
- ✓ Integridad referencial. Se pueden definir llaves foráneas entre tablas relacionadas para asegurarse de que un registro no puede ser eliminado de una tabla si aún está siendo referenciado por otra tabla.

Desventajas:

Aunque MySQL se incluye en el grupo de sistemas de bases de datos relacionales, sus desarrolladores fueron implementando únicamente lo que necesitaban para garantizar su funcionamiento óptimo, y carece de algunas de sus principales características como:

- Subconsultas.
- Procedimientos almacenados y desencadenadores.
- Transacciones: A partir de las últimas versiones se incluye el soporte para transacciones, pero hay que activarlo de modo especial, no existe por defecto.

1.13.3 Herramientas de desarrollo

EMS Data Generator 2005 for PostgreSQL

Es una potente herramienta para la generación de datos de prueba para BD PostgreSQL, brinda la posibilidad de generar datos para varias tablas a la vez. Permite seleccionar las tablas y campos que se quieren llenar, definir rango de valores para los mismos, generar campos de tipo char utilizando máscara, además de obtener los valores que son resultado de las consultas SQL. Brinda la posibilidad de definir plantillas de valores para su utilización futura.

Características:

- ✓ Posee una interfaz de usuario amigable.
- ✓ Genera datos para BD diferentes, ubicadas en una misma computadora.
- ✓ Soporta todos los tipos de datos que posee PostgreSQL: arreglos, direcciones de red, tipos geométricos, etc.
- ✓ Genera datos de diferentes formas para cada campo, ya sea desde una lista de valores predefinida, valores aleatorios haciendo uso de caracteres previamente definidos o valores incrementales.
- ✓ Permite utilizar el resultado de consultas SQL como posibles valores para la generación de datos.
- ✓ Control automático sobre la integridad referencial.
- ✓ Posee una amplia variedad de parámetros de generación para cada tipo de campo.
- ✓ Permite generar valores nulos para un por ciento definido de los valores generados para cada campo.
- ✓ Permite almacenar todos los parámetros de generación en la sesión del asistente en uso.
- ✓ Se seleccionó esta herramienta para la realización de las pruebas de carga intensiva a la BD.

1.14 Conclusiones

Después de realizar un estudio profundo sobre las BD y otros aspectos necesarios relacionados con el tema, se han adquirido un conjunto de conocimientos para el desarrollo del diseño de la BD de la UCJ de la UCI, donde:

- Se seleccionó el modelo de BD relacional.
- Se escogió RUP como metodología para el desarrollo de la actividad diseñar BD y UML como lenguaje de modelado, para la realización del diagrama de clases persistentes.
- Se seleccionó Visual Paradigm for UML 6.1 Enterprise Edition para la realización del diagrama de clases persistentes y para diseñar la base de datos.
- Se decidió utilizar el SGBD PostgreSQL 8.1, utilizando para la administración de la BD: EMS SQL Manager for PostgreSQL y para la generación de datos de prueba EMS Data Generador for PostgreSQL.

CAPÍTULO # 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

En este capítulo se realiza la descripción del diseño de la Base de Datos propuesta, haciendo mención a los requisitos funcionales y no funcionales referente al trabajo del analista. Además se presentan algunas estrategias de integración de la solución con otros módulos o sistemas y el análisis de optimización de querys. También se hará una descripción de las clases, del diagrama entidad relación, de las tablas y la arquitectura utilizada, considerando las ventajas que brinda para la elaboración del trabajo.

2.2 Estrategia de integración de la solución con otros módulos o sistemas

El sistema de almacenamiento de información de la UJC en la UCI está estructurado en seis módulos: Definición de la organización de base, Datos del expediente del militante, Control del estado de la militancia, Vida Interna, Cuadros y Reportes; a través de los cuales se garantiza toda la información referente a la organización juvenil. Considerando que se está desarrollando un sistema de gestión para control de los integrantes de la UJC, se puede integrar estos módulos para un mejor control.

Por lo que se identificaron las clases persistentes a partir del diagrama de diseño que se obtuvo del trabajo realizado por los analistas. Se integraron las clases persistentes en un único diagrama de clases, incluyendo de esta forma las relaciones entre clases de los distintos módulos. Se validó en todo momento la correcta representación del diagrama de clases, así como los campos que posee cada una de ella. Acto seguido se realizó el diagrama entidad-relación para la representación de la estructura lógica y física de la BD, a través de las relaciones entre las diferentes entidades obtenidas de cada módulo.

2.3 Requisitos

Un **requisito** es una necesidad documentada sobre el contenido, comprende todas las tareas relacionadas con condiciones o capacidades que debe hacer o poseer un sistema para satisfacer un contrato, estándar, especificación, u otra documentación formalmente impuesta.

2.3.1 Requisitos Funcionales

Un requisito funcional (RF) define el comportamiento interno del software, es decir, capacidades o condiciones que el sistema debe cumplir.

1RF. Gestionar la creación de estructura organizativa

1.1 Debe permitir crear, modificar o eliminar un Comité Nacional, Comités de Base provinciales, municipales, comités UJC, comités de base y comités de base independiente. Debe quedar el registro de los comités de base anteriores.

2RF. Gestionar el control del estado de la militancia

2.1 Debe permitir gestionar los datos de los militantes en cuanto a sus movimientos, los que pueden ser por alta o por baja, tener en cuenta que se debe guardar todas las direcciones particulares que ha tenido y utilizar la última, además tener en cuenta cuando fue el último mes que cotizó (mes y año), puede que sea más de un mes y permitir restringir el carné de identidad para que la edad no sea menor de 14 años y tenga 11 dígitos.

2.2 Debe validarse que los movimientos internos son entre los Comité de Bases dentro del Comité Primario y que antes de terminar esta acción deben quedar ejecutados los dos movimientos (alta por traslado interno, baja por traslado interno), los de municipios son entre los Comité de Bases de un Comité Primario a otro.

2.3 No se debe permitir eliminar a un militante ya que no se borran los militantes de la base de datos.

3RF. Gestionar el Control del estado de Vida Interna

3.1 Debe permitir gestionar los datos de los miembros del Universo Juvenil, ya sea insertar, modificar o eliminar ya que estos pueden ser eliminados de la Base de Datos.

3.2 Debe permitir realizar crecimiento a la UJC, para ello el joven debe estar entre los 14 y 30 años y debe formar parte del Universo Juvenil.

3.3 Debe permitir conocer cuántos militantes son arribantes al PCC y quienes son los mismos. Los militantes arribantes son aquellos que están en la edad entre 29 y 30 años.

3.4 Debe permitir realizar seguimiento al proceso de crecimiento al PCC realizado a un militante.

3.5 Debe permitir insertar, modificar y eliminar sanciones de los militantes, las que pueden ser internas o externas.

3.6 Debe permitir insertar, modificar y eliminar evaluaciones de los militantes. De este proceso es necesario conocer la fecha de la evaluación y que señalamientos fueron hechos a un militante.

4RF. Entrar al módulo de reportes

4.1 Debe permitir crear reportes de tablas (plantillas predefinidas) y mantenerlos actualizados, listos para ser consultados en el momento que consideren los usuarios, mostrando los campos según los datos contenidos en la BD, excluyendo los de valores numéricos que sean iguales a cero.

4.2 Debe permitir seleccionar los parámetros deseados para realizar la búsqueda de un reporte.

5RF. Permitir obtener listado de altas y bajas

5.1 Debe permitir obtener un listado de altas y bajas de los militantes

5.1.1 Nombre y apellidos.

5.1.2 CI.

5.1.3 Provincia.

5.1.4 Municipio.

5.1.5 Fecha del movimiento.

5.1.6 Dirección particular.

5.1.7 Comité de Base donde se incorpora.

5.1.8 Comité de base donde causa baja.

6RF. Permitir obtener informe de estructura

6.1 Debe permitir conocer el total de militantes del Comité UJC, de cualquier Comité Primario, del Comités de Base y de los Comités de Base Independientes.

7RF. Permitir obtener informe dobles militantes

7.1 Debe permitir obtener un informe sobre los dobles militantes:

7.1.1 Nombre y apellidos.

7.1.2 Comité Primario al que pertenece.

7.1.3 Comité de Base al que pertenece.

7.1.4 Sexo.

7.1.5 Color de la piel.

7.1.6 CI.

7.1.7 Fecha de ingreso al PCC.

8RF. Permitir obtener informe organizaciones activadas y desactivadas

8.1 Debe permitir obtener un informe sobre las organizaciones activadas y desactivadas:

8.1.1 Nombre del Comité UJC que pertenece.

8.1.2 Nombre del Comité Primario que pertenece.

8.1.3 Nombre del Comité de Base.

8.1.4 Meses en que ocurre la activación o desactivación.

8.1.5 Año en que ocurre la activación o desactivación.

9RF. Permitir obtener informe del Universo Juvenil

9.1 Debe permitir obtener informe del Universo Juvenil:

9.1.1 Cantidad total de universo.

9.1.2 Cantidad de universo por Comité Primario.

9.1.3 Sexo.

9.1.4 Raza.

9.1.5 Clasificador ocupacional.

10RF. Permitir obtener informe del Crecimiento

10.1 Debe permitir obtener informe del Crecimiento:

10.1.1 Jóvenes en proceso.

10.1.2 Nombre y apellidos.

10.1.3 CI.

10.1.4 Comité de Base.

10.1.5 Sexo.

10.1.6 Color de la piel.

10.1.7 Labor que realiza.

10.1.8 Clasificador ocupacional.

10.1.9 Asamblea de ejemplares (si o no).

10.1.10 Escuelas políticas (si o no).

10.1.11 Principio de voluntariedad (si o no).

10.1.12 Fecha de ingreso.

10.1.13 Incorporados (el momento en que se da alta en el Comité de Base).

11RF. Permitir obtener informe de Sanciones

11.1 Debe permitir obtener informe de Sanciones:

11.1.1 Nombre.

11.1.2 Fecha de aprobación por la organización de base.

11.1.3 Organización de base.

11.1.4 Tipo de sanción.

11.1.5 Causa(s) de la sanción.

11.1.6 Mes.

11.1.7 Año.

12RF. Permitir obtener informe de Apelaciones

12.1 Debe permitir obtener informe de Apelaciones:

12.1.1 Nombre.

12.1.2 Fecha de aprobación de la sanción.

12.1.3 Tipo de sanción.

12.1.4 Fecha de apelación.

12.1.5 Organización de base.

12.1.6 Resultado de la apelación.

12.1.7 Mes.

12.1.8 Año.

13RF. Permitir obtener informe de Evaluaciones

13.1 Debe permitir obtener informe de Evaluaciones:

13.1 Nombre del militante

13.1 Evaluación por mes y año.

14RF. Permitir obtener informe de Arribantes al PCC

14.1 Debe permitir obtener informe de Arribantes al PCC:

14.1.1 Nombre y apellidos.

14.1.2 Militantes avalados (30 o -30 años de edad).

14.1.3 Proceso iniciado (30 o -30 años de edad).

14.1.4 Proceso aprobado (30 o -30 años de edad).

14.1.5 Arribante acogido a principio de voluntariedad (X).

14.1.6 No ingresan (X).

14.1.7 Arribante no avalado (X).

14.1.8 No desea ser procesado (X).

2.3.2 Requisitos no funcionales aplicables a la BD

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido o confiable.

- **Usabilidad**

1RNF. Se debe garantizar un acceso fácil y rápido a los usuarios, así como de una documentación y manuales de ayuda para un mejor uso de la aplicación.

- **Rendimiento**

2RNF. La velocidad procesamiento debe ser rápida al igual que el tiempo de respuesta de la información.

- **Soporte**

3RNF. Se requiere para la instalación de un servidor Web con funcionalidades relacionadas con el manejo de la Base de Datos con un SGBD que soporte grandes volúmenes de datos (PostgreSQL) y el servicio de interpretación de códigos con php5 y Symfony.

- **Portabilidad**

4RNF. El sistema podrá ser usado sobre los sistemas operativos Linux y Windows.

- **Seguridad**

5RNF. Garantizar que la información sensible solo pueda ser vista por los usuarios con el nivel de acceso adecuado. El sistema debe contar con protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

- **Confiabilidad**

6RNF. El sistema será usado y administrado solamente por trabajadores del departamento de control de la universidad, por lo tanto la información que fluirá en el mismo, será la real emitida por cada uno de los grupos o departamentos. Podrán acceder a visualizar ciertas informaciones, directivos de otras áreas, previa consulta con la dirección del proyecto y los desarrolladores de la aplicación.

- **Software**

7RNF. Se deberá disponer para poder instalar la aplicación desarrollada, con Sistema Operativo Windows 98 o superior. Igual sistema operativo en las computadoras personales clientes de los usuarios, las cuales además accederán al sistema usando un navegador compatible o superior con Internet Explorer 5.5, Netscape, Mozilla 1.7 o superior o FireFox 0.9.3 o superior.

- **Restricciones en el diseño y la implementación**

8RNF. Debe ser una aplicación Web desarrollada con la tecnología para creación de páginas dinámicas PHP y base de datos en PostgreSQL. Se utilizarán herramientas de desarrollo que garanticen la calidad de todo el ciclo de desarrollo del producto. Se usará el lenguaje de programación php5 con Symfony y Visual Paradigm como herramienta CASE para el modelado de los artefactos que se elaboran en cada uno de los flujos de trabajo.

2.4 Descripción de la arquitectura y fundamentación

La BD propuesta consta de 50 tablas en el modelo físico, las cuales serán utilizadas por los distintos módulos del proyecto, haciendo uso principalmente de las tablas: militante, universo juvenil, sanción y evaluación; donde se almacena el mayor grueso de la información. La aplicación se

desarrollará en php5 con Symfony por las ventajas que brinda este framework de acuerdo al patrón de arquitectura que utiliza. Debido a que ambos son orientado a objetos y que la base de datos es objeto - relacional; para acceder a los datos se necesita de una capa ORM (Mapeo Objeto-Relacional) que traduzca la lógica relacional de la base de datos a un modelo de objeto de datos, toda esta lógica relacionada con los datos se incluye en el modelo, aquí es donde actúa la librería propel que genera las clases de la capa del modelo automáticamente y se realiza la abstracción de la base de datos.

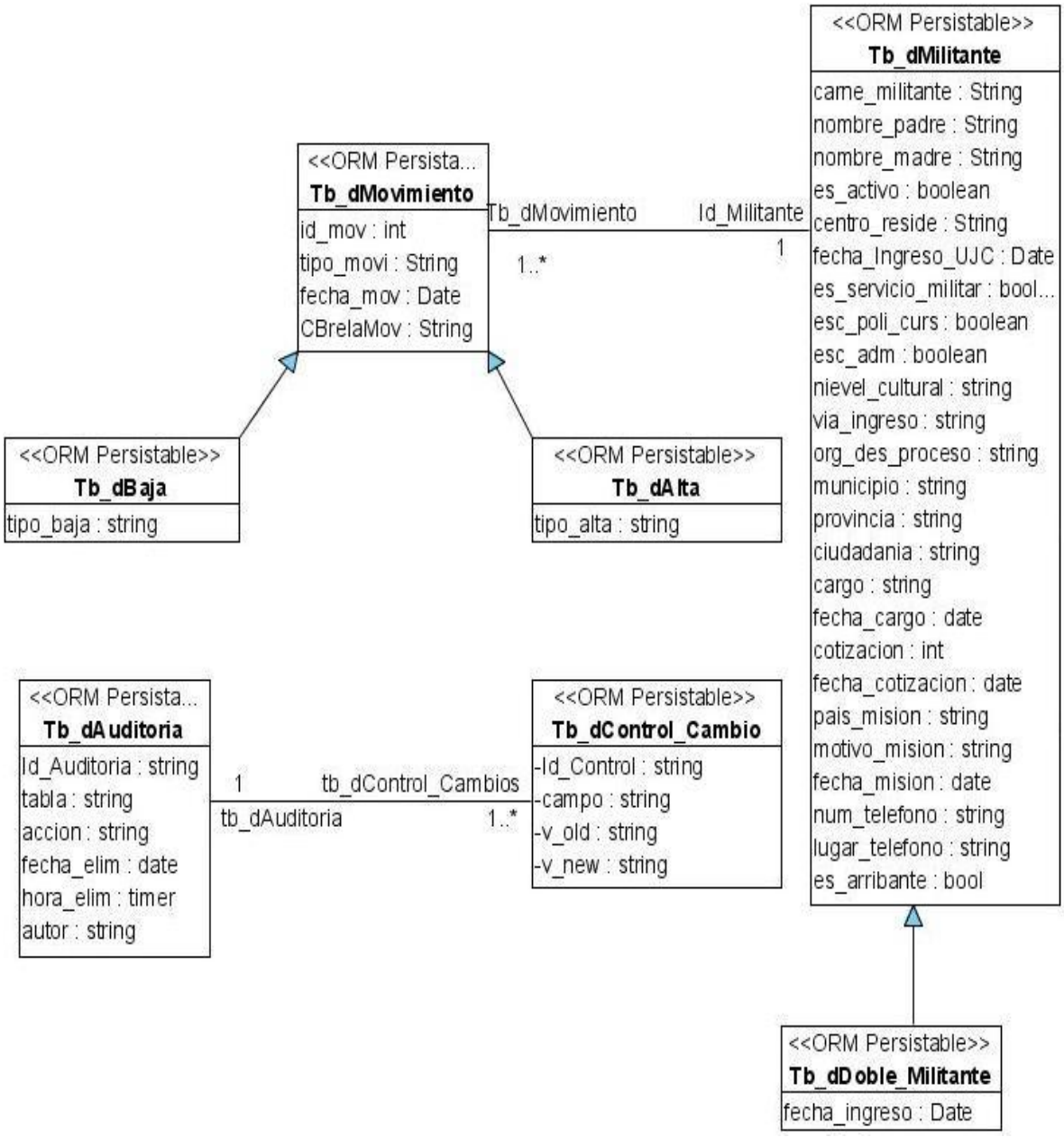
Php5: Es la última versión de un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

Symfony: Es un framework basado en el MVC (Modelo Vista Controlador), donde el Modelo representa la lógica del negocio, es decir las reglas, restricciones y condiciones definidas para la operación de la aplicación, La Vista se encarga de generar la página web con la cual el usuario va a interactuar, y el Controlador responde a las interacciones del usuario, y se encarga de generar cambios ya sea en las Vistas o en el Modelo.

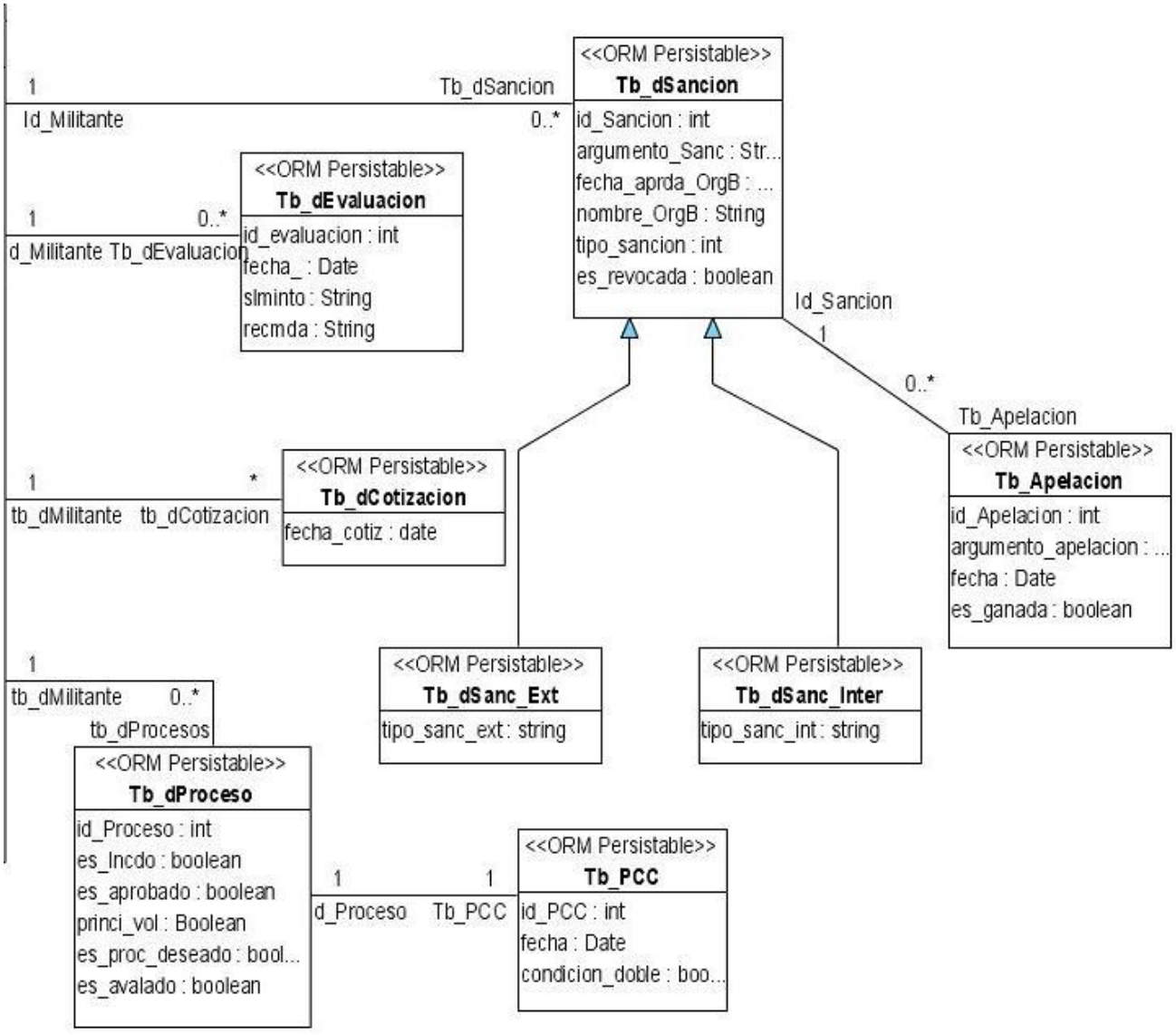
2.5 Diagrama de clases persistente

Las clases representan son la forma de modelar entidades del mundo real a través de atributos y métodos comunes. Los objetos o instancias de clases son persistentes cuando pueden almacenarse en memoria secundaria (una base de datos o un archivo) y recuperarse posteriormente, en lugar de existir solo en memoria principal, la persistencia es la propiedad de los objetos de trascender su estado en el tiempo y el espacio.

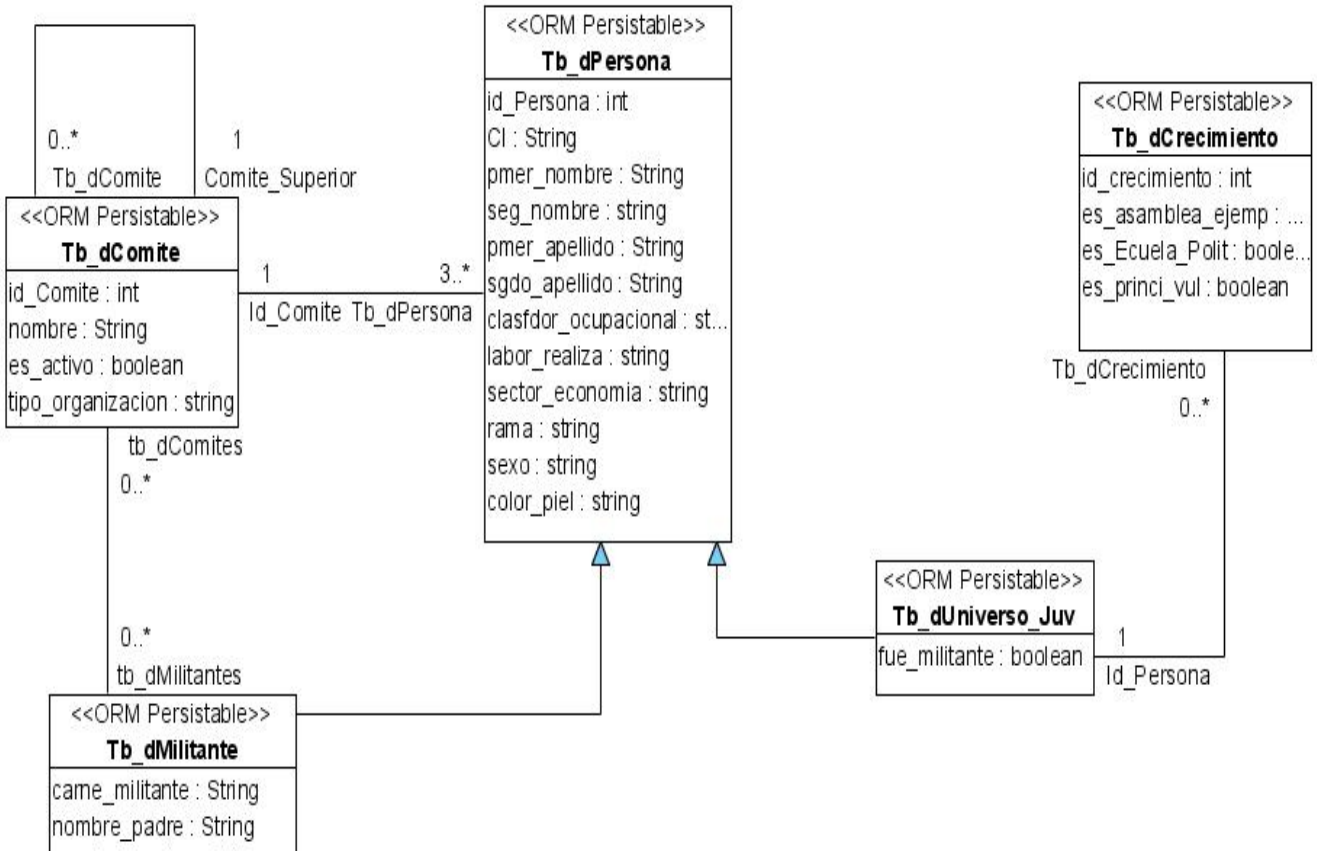
A continuación se muestra el diagrama de clases persistentes realizado a partir del diagrama de clases del diseño que se adquirió del trabajo realizado por los analistas.



Figura# 2.1 Diagrama de clases persistente



Figura# 2.2 Diagrama de clases persistente (continuación)



Figura# 2.3 Diagrama de clases persistente (continuación)

2.6 Descripción del diagrama de clases persistentes

Tabla 1.1Descripción de la clase "Auditoria"

Nombre: Tb_dAuditoria	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Auditoria	string
tabla	string
accion	string
fecha	date
hora	timer
autor	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dAuditoria.

Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 1.2 Descripción de la clase "Comité"

Nombre: Tb_dComite	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_Comite	int
nombre	String
es_activo	boolean
tipo_organizacion	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dComite.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 1.3 Descripción de la clase "Persona"

Nombre: Tb_dPersona	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_Persona	int
CI	String
pmer_nombre	String
seg_nombre	string
pmer_apellido	String
sgdo_apellido	String
clasfdor_ocupacional	string
labor_realiza	string
sector_economia	string
rama	string
sexo	string
color_piel	string
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dPersona.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 1.4 Descripción de la clase " Universo Juvenil"

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

Tabla 1.5 Descripción de la clase " Crecimiento"

Nombre: Tb_dCrecimiento	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_crecimiento	int
es_asamblea_ejemp	boolean
es_Ecuela_Polit	boolean
es_princi_vul	boolean
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dCrecimiento.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 1.6 Descripción de la clase " Militante"

Nombre: Tb_dMilitante	
Tipo de clase: Entidad	
Atributo	**Tipo**
carne_militante	String
nombre_padre	String
nombre_madre	String
es_activo	boolean
centro_reside	String
fecha_Ingreso_UJC	Date
es_servicio_militar	boolean
esc_poli_curs	boolean
esc_adm	boolean
nievel_cultural	string

via_ingreso	string
org_des_proceso	string
municipio	string
provincia	string
ciudadania	string
cargo	string
fecha_cargo	date
cotizacion	int
fecha_cotizacion	date
pais_mision	string
motivo_mision	string
fecha_mision	date
num_telefono	string
lugar_telefono	string
es_arribante	bool
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dMilitante.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 1.7 Descripción de la clase " Apelación"

Nombre: Tb_Apelacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_Apelacion	int
argumento_apelacion	String
fecha	Date
es_ganada	boolean
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_Apelacion.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 1.8 Descripción de la clase "Sanción”

Nombre: Tb_dSancion
Tipo de clase: Entidad

Atributo	Tipo
id_Sancion	int
argumento_Sanc	String
fecha_aprda_OrgB	Date
nombre_OrgB	String
tipo_sancion	int
es_revocada	boolean
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dSancion.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 1.9 Descripción de la clase " Evaluación"

Nombre: Tb_dEvaluacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_evaluacion	int
fecha_	Date
slminto	String
recmda	String
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dEvaluacion.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 2.1 Descripción de la clase " Movimiento"

Nombre: Tb_dMovimiento	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_mov	int
fecha_mov	Date
tipo_movi	String
CBrelaMov	String
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dMovimiento.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 2.2 Descripción de la clase " Cotización"

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

Tabla 2.3 Descripción de la clase " Sanción Externa"

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

Tabla 2.4 Descripción de la clase " Sanción Interna"

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

Tabla 2.5 Descripción de la clase " Historial"

Nombre: Tb_dHistorial	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_historial	int
fecha	Date
nombre	String
papellido	String

sapellido	String
ci	String
accion	String
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dHistorial.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 2.6 Descripción de la clase " Baja"

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

Tabla 2.7 Descripción de la clase " Alta"

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

Tabla 2.8 Descripción de la clase " Doble Militante”

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

Tabla 2.9 Descripción de la clase " Proceso"

Nombre: Tb_dProceso	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_Proceso	int
es_Incdo	boolean
es_aprobado	boolean
princi_vol	Boolean
es_proc_deseado	boolean
es_avalado	boolean
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_dProceso.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 3.1 Descripción de la clase " PCC"

Nombre: Tb_PCC	
Tipo de clase: Entidad	
Atributo	**Tipo**
id_PCC	int
fecha	Date
condicion_doble	boolean
Responsabilidad:	
Nombre:	Almacena todos los datos de la entidad Tb_PCC.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

Tabla 3.2 Descripción de la clase " Control de Cambio"

Nombre: Tb_dControl_Cambio	
Tipo de clase: Entidad	
Atributo	**Tipo**
Id_Auditoria	string
tabla	string
accion	string
fecha	date
autor	string
Responsabilidad:	

Nombre:	Almacena todos los datos de la entidad Tb_dControl_Cambio.
Descripción:	Contiene los valores y tipos, que le pertenezcan a la entidad.

2.7 Diseño de la Base de Datos

El diseño de esta Base de Datos relacional, tiene el objetivo de generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, y que a la vez faciliten la recuperación de la información sin ningún problema.

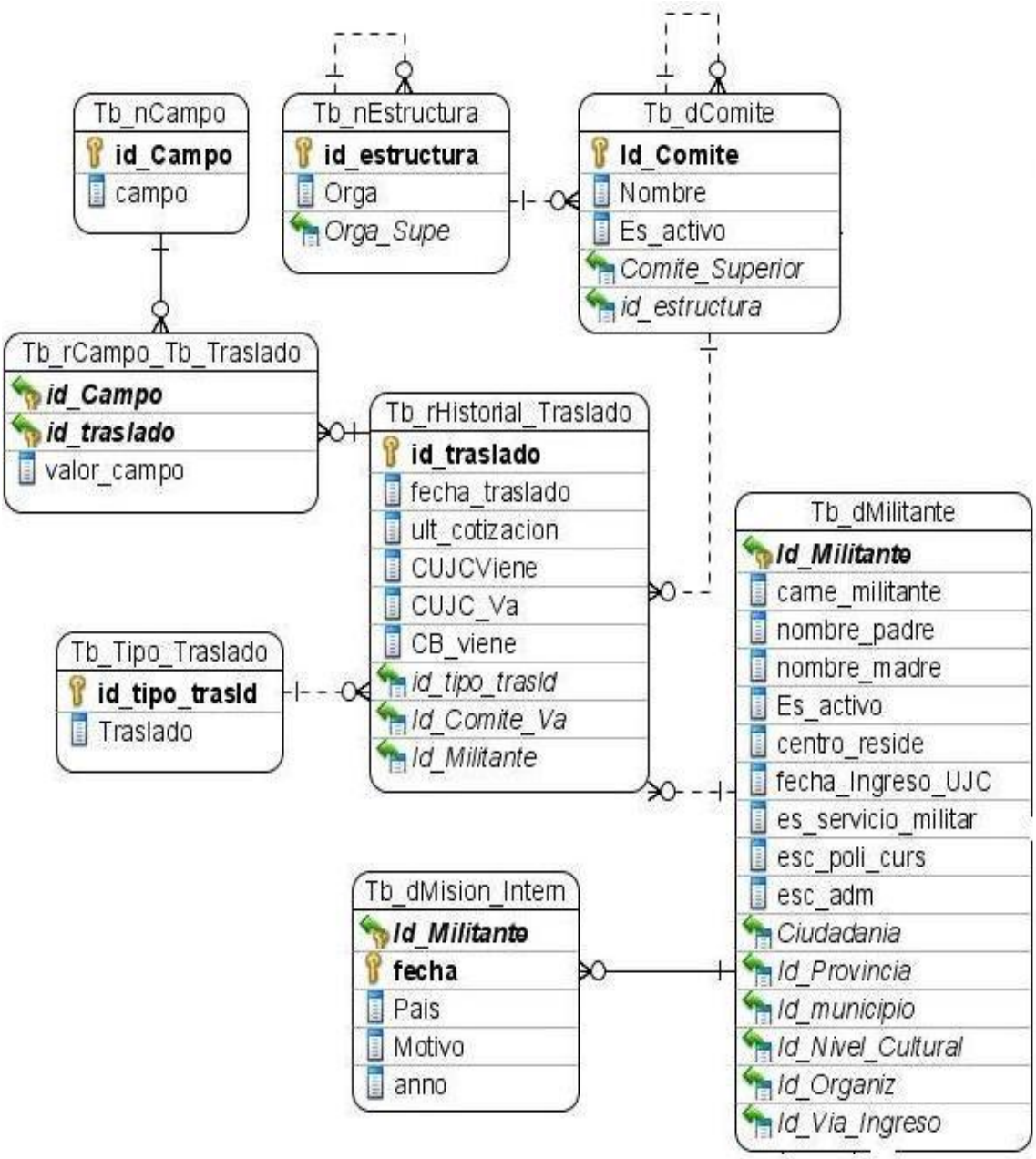
A continuación se mencionan algunos de los patrones de diseño utilizados en la construcción del diagrama entidad-relación.

Patrones Brown

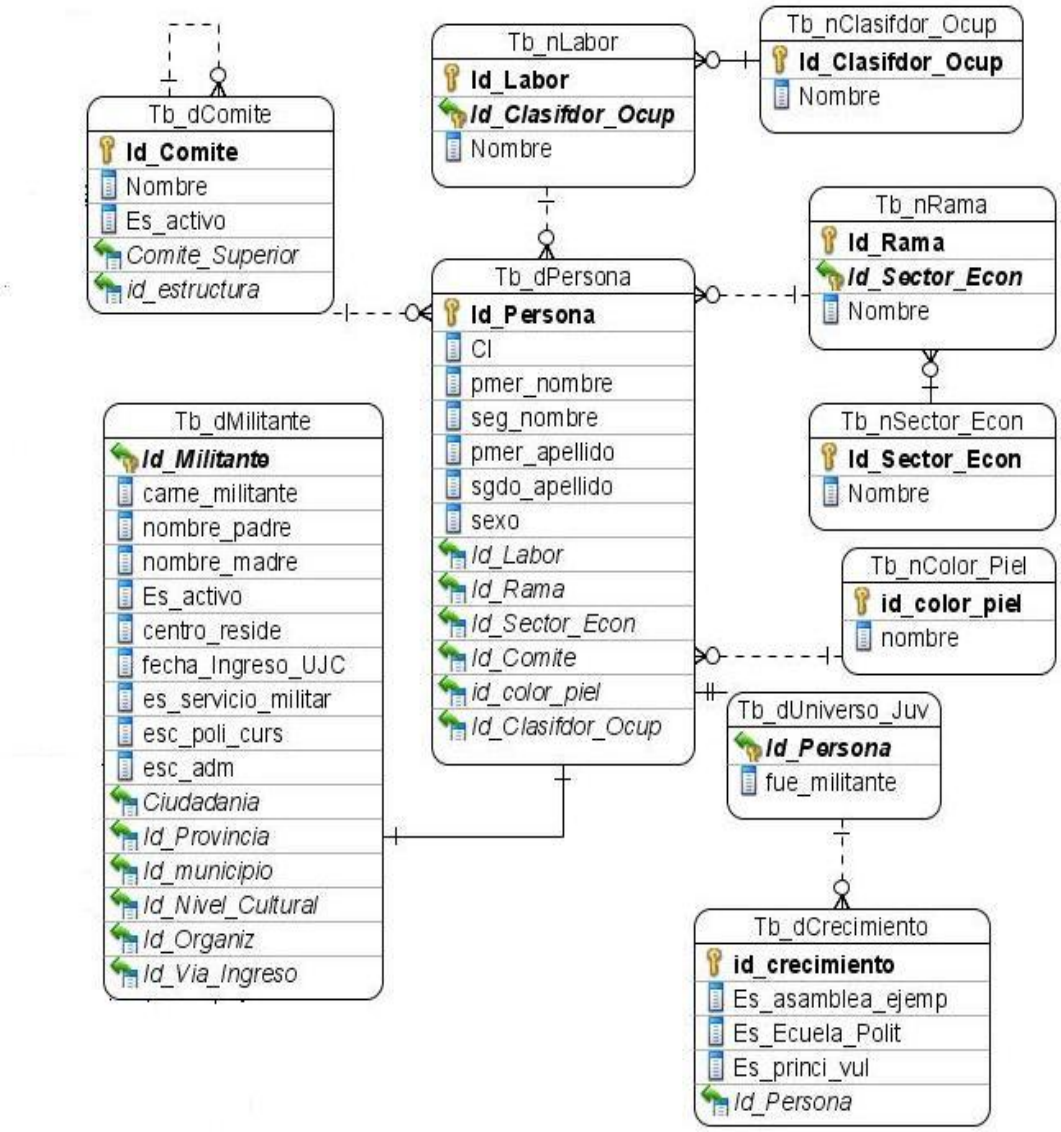
- ✓ Representar objetos como tablas.
- ✓ Identificador de objetos.
- ✓ Representar relaciones como tablas.

2.8 Diagrama entidad-relación de la BD

En este epígrafe se hace una representación de las tablas más importantes del diseño propuesto. Para ver el diseño en su forma general se puede consultar el anexo.



Figura# 2.4 Diagrama entidad-relación



Figura# 2.5 Diagrama entidad-relación (continuación)

2.9 Descripción del diagrama entidad-relación

Tabla 3.3 Descripción de la entidad "Auditoría"

Nombre: Tb_dAuditoria			
Descripción: Historial de afectaciones de todas las tablas			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Auditoria	varchar(31)	No	Llave primaria.
tabla	varchar(50)	No	Nombre de la tabla que sufre cambio

accion	varchar(10)	No	Nombre de la acción de cambio
fecha	date(0)	No	Fecha de afectación
hora	time(7)	No	Hora de afectación
autor	varchar(50)	No	Usuario que realizó la acción

Tabla 3.4 Descripción de la entidad "Clasificador Ocupacional"

Nombre: Tb_nClasifdor_Ocup			
Descripción: Nomenclador que guarda todas las ocupaciones adoptadas.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Clasifdor_Ocup	int4(0)	No	Identificador principal
Nombre	text(0)	No	Nombre de la ocupación

Tabla 3.5 Descripción de la entidad "Rama de la Economía"

Nombre: Tb_nRama			
Descripción: Nomenclador que guarda todas las ramas de los sectores de la economía.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Rama	int4(0)	No	Identificador principal
Id_Sector_Econ	int4(0)	No	Identificador del sector de la economía al cual pertenece.
Nombre	text(0)	No	Nombre de la rama

Tabla 3.6 Descripción de la entidad "Sector de la Economía"

Nombre: Tb_nSector_Econ.			
Descripción: Nomenclador que guarda todas las ramas de los sectores de la economía.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Sector_Econ	int4(0)	No	Identificador principal
Nombre	text(0)	No	Nombre del sector

Tabla 3.7 Descripción de la entidad "Comité"

Nombre: Tb_dComite			
Descripción: Guarda información referente a los comités			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Comite	int4(255)	No	Identificador principal
Nombre	varchar(255)	No	Nombre del Comité
Es_activo	bool(0)	No	Hace referencia si esta activo o no
Comite_Superior	int4(255)	Yes	Comité superior al que pertenece

id_estructura	int4(0)	Yes	Identificador de la estructura.

Tabla 3.8 Descripción de la entidad "Estructura"

Nombre: Tb_nEstructura			
Descripción: Nomenclador que guarda información referente estructura de los comité.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_estructura	int4(0)	No	Identificador Principal
Orga	text(0)	No	Tipo de Comité(CUJC,CP,CB,CBI)
Orga_Supe	int4(0)	Yes	Tipo de organización superior.

Tabla 3.9 Descripción de la entidad "Labor"

Nombre: Tb_nLabor			
Descripción: Nomenclador que guarda todas las labores de las ocupaciones adoptadas			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Labor	int4(0)	No	Identificador principal
Id_Clasifdor_Ocup	int4(0)	No	Tipos de clasificador ocupacional existentes
Nombre	text(0)	No	Nombre de la Labor

Tabla 4.1 Descripción de la entidad "Tipo de Alta"

Nombre: Tb_nTipoAlta			
Descripción: Nomenclador que guarda los nombres de los tipos de altas existente.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_tipoalta	int4(0)	No	Identificador principal
nombre	text(0)	No	Nombre del Alta

Tabla 4.2 Descripción de la entidad "Tipo de Baja"

Nombre: Tb_Tipo_Baja			
Descripción: Nomenclador que guarda los nombres de los tipos de Bajas existentes.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_tipo_baja	int4(0)	No	Identificador principal
nombre	text(0)	No	Nombre de la Baja

Tabla 4.3 Descripción de la entidad "Crecimiento"

Nombre: Tb_dCrecimiento			
Descripción: Guarda información a todos los crecimientos realizados al Universo Juvenil.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**

Id_crecimiento	int4(0)	No	Identificador principal
Id_Persona	int4(0)	No	Referencia al identificador de la persona
Es_asamblea_ejemp	bool(0)	No	Expresa a través de true o false si fue la asamblea de ejemplares quien le realizó el crecimiento
Es_Ecuela_Polit	bool(0)	No	Expresa a través de true o false se ha cursado escuela política
Es_princi_vul	bool(0)	No	Expresa a través de true o false si se acoge al principio de voluntariedad

Tabla 4.4 Descripción de la entidad "Historial de Traslado"

Nombre: Tb_rHistorial_Traslado			
Descripción: Historial de las estancias de todas las personas en los distintos Comités			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_traslado	int4(0)	No	Identificador Principal
fecha_traslado	date(0)	No	Fecha en la cual se realizó el traslado
ult_cotizacion	date(0)	No	Guarda la última cotización.
CUJCViene	text(0)	No	Comité UJC al que pertenece el CB donde se encuentra (es calculable)
CUJC_Va	text(0)	No	Comité UJC al que pertenece el comité CB para donde va
CB_viene	text(0)	No	Referencia del CB de donde viene
id_tipo_trasld	int4(0)	No	Referencia del tipo de traslado que se le va hacer
Id_Comite_Va	int4(255)	Yes	Referencia al CB donde se va a mover
Id_Militante	int4(0)	Yes	Identificador Principal

Tabla 4.5 Descripción de la entidad "Tipo de Traslado"

Nombre: Tb_Tipo_Traslado			
Descripción: Nomenclador que guarda la información referente a los tipos de Traslado que existen.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_tipo_trasld	int4(0)	No	Identificador principal
Traslado	text(0)	No	Nombre del Traslado

Tabla 4.6 Descripción de la entidad "Persona"

Nombre: Tb_dPersona
Descripción: Guarda información referente a todas las personas del Universo del Discurso(UD)

Atributo	Tipo	¿Acepta Nulos?	Descripción.
Id_Persona	int4(0)	No	Identificador principal
CI	varchar(11)	No	Carné identidad de la persona
pmer_nombre	varchar(255)	No	Nombre de la persona
seg_nombre	varchar(255)	Yes	Segundo nombre de la persona
pmer_apellido	text(50)	Yes	Primer apellido de la persona
sgdo_apellido	text(50)	Yes	Segundo apellido de la persona
sexo	char(1)	No	Sexo de la persona
Id_Labor	int4(0)	Yes	Hace referencia a la labor que realiza dentro del clasificador ocupacional al que pertenece
Id_Rama	int4(0)	Yes	Hace referencia a la rama que desarrolla dentro del sector de la economía
Id_Sector_Econ	int4(0)	Yes	Hace referencia al sector de la economía
Id_Comite	int4(255)	Yes	Identificador del Comité al cual pertenece
id_color_piel	int4(0)	No	Identificador del color de la piel
Id_Clasifdor_Ocup	int4(0)	Yes	Identificador del clasificador ocupacional.

Tabla 4.7 Descripción de la entidad "Universo Juvenil"

Nombre: Tb_dUniverso_Juv			
Descripción: Guarda la información referente del Universo Juvenil			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Persona	int4(0)	No	Identificador principal
fue_militante	bool(0)	No	Identifica si alguna vez fue militante

Tabla 4.8 Descripción de la entidad "Universo Juvenil"

Nombre: Tb_dAlta			
Descripción: Guarda información referente a todas las altas realizadas			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_mov	int4(0)	No	Identificador principal (PK y FK)
id_tipoalta	int4(0)	No	Referencia al tipo de alta

Tabla 4.9 Descripción de la entidad "Baja"

Nombre: Tb_dBaja			
Descripción: Guarda información referente a todas las Bajas realizadas			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_mov	int4(0)	No	Identificador principal. Llave foránea.

id_tipo_baja	int4(0)	No	Referencia al tipo de baja.

Tabla 5.1 Descripción de la entidad "Causa"

Nombre: Tb_nCausa			
Descripción: Nomenclador que guarda las distintas causas por la cuales se aplican las sanciones			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_causa	int4(0)	No	Identificador principal
causa	text(255)	No	Nombre de la causa

Tabla 5.2 Descripción de la entidad "Sanciones"

Nombre: Tb_dSancion			
Descripción: Guarda la información referente a las sanciones			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Sancion	int4(0)	No	Identificador principal
Argumento_Sanc	text(0)	No	Argumento de la sanción
Fecha_aprda_OrgB	date(0)	No	Aprobación de la organización de base
Nombre_OrgB	text(0)	No	Nombre de la organización de base
Id_Militante	int4(0)	No	Identificador del militante
tipo_sancion	int4(0)	No	Tipo de sanción
Es_revocada	bool(0)	No	Guarda si la sanción es revocada o no.

Tabla 5.3 Descripción de la entidad "Campo_Traslado"

Nombre: Tb_rCampo_Tb_Traslado			
Descripción: Guarda la información referente a los capos utilizados por el tipo de Traslado			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_Campo	int4(0)	No	Referencia al nombre del campo
id_traslado	int4(0)	No	Referencia al traslado
valor_campo	text(0)	No	Valor del campo seleccionado.

Tabla 5.4 Descripción de la entidad "Campo"

Nombre: Tb_nCampo			
Descripción: Guarda la información referente a los tipos de Campos que varían en el Traslado			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_Campo	int4	No	Identificador del campo
campo	text	No	Nombre del campo

Tabla 5.5 Descripción de la entidad "Color de la Piel"

Nombre: Tb_nColor_Piel			
Descripción: Nomenclador que guarda los distintos tipos de color de la piel existentes.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_color_piel	int4	No	Identificador del campo
nombre	text	No	Nombre del campo

Tabla 5.6 Descripción de la entidad "Causa _Sanción"

Nombre: Tb_RCausa_Tb_dSancion			
Descripción: Guarda información referente a las distintas causas por las cuales se aplica la Sanción.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Sancion	int4(0)	No	Referencia al identificador de la Sanción
id_causa	int4(0)	No	Referencia al identificador de la Causa

Tabla 5.7 Descripción de la entidad "Movimiento"

Nombre: Tb_dMovimiento			
Descripción: Guarda información referente al registro de Altas y Bajas en un CB y de un Militante			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_mov	int4(0)	No	Identificador principal
fecha_mov	date(0)	No	fecha de alta y/o baja
tipo_movi	text(0)	No	Referente el tipo de movimiento si es alta o baja(discriminante)
CBrela_mov	text(0)	Yes	CB el cual realiza el movimiento
Id_Militante	int4(0)	No	Referencia al militante de que se le realizó el movimiento

Tabla 5.8 Descripción de la entidad "Misión Internacionalista"

Nombre: Tb_dMision_Intern			
Descripción: Guarda la información referente a todas las misiones internacionalista del militante			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Militante	int4(0)	No	Identificador del militante
País	text(0)	No	País donde desarrolló misión
Motivo	text(0)	Yes	Motivo de la misión
anno	int4(0)	No	Año en el cual cumplió la misión

Tabla 5.9 Descripción de la entidad "Apelación"

Nombre: Tb_Apelacion			
Descripción: Guarda la información referente a las apelaciones realizadas por determinada sanción			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Apelacion	int4(0)	No	Identificador principal
argumento_apelacion	varchar(255)	No	Argumento de la sanción
fecha	date(0)	No	Fecha de la apelación.
Es_ganada	bool(0)	No	Guarda si es ganada o no la apelación.

Tabla 6.1 Descripción de la entidad "Vía de Ingreso Utilizada"

Nombre: Tb_nVia_Ingreso_Utilizada			
Descripción: Guarda la información referente a todas las vías de ingreso a la UJC existentes			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Via_Ingreso	int4(0)	No	Identificador principal
nombre	varchar(255)	No	Nombre de la vía ingreso

Tabla 6.2 Descripción de la entidad "Sanción Interna"

Nombre: Tb_dSanc_Inter			
Descripción: Guarda la información referente a las sanciones internas			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Sancion	int4(0)	No	Identificador principal
Id_nSanc_Int	int4(0)	No	Tipo de sanción interna

Tabla 6.3 Descripción de la entidad "Sanción Externa"

Nombre: Tb_dSanc_Ext			
Descripción: Guarda la información referente a las sanciones externas			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Sancion	int4(0)	No	Identificador principal
Id_nSanc_Ext	int4(0)	No	Tipo de sanción externa

Tabla 6.4 Descripción de la entidad "Militante"

Nombre: Tb_dMilitante			
Descripción: Guarda la información referente a los militante de la UJC			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Militante	int4(0)	No	Identificador principal del militante
carne_militante	varchar(10)	No	Carné de la militancia en la UJC

nombre_padre	text(0)	Yes	Nombre del padre del militante
nombre_madre	text(50)	Yes	Nombre de la madre del militante
Es_activo	bool(0)	No	Si el militante es activo o no.
centro_reside	text(50)	Yes	Centro donde reside el militante
fecha_Ingreso_UJC	date(0)	No	Fecha de ingreso a la filas de la UJC
Id_Nivel_Cultural	int4(0)	No	Nivel cultural que tiene el militante
Id_Via_Ingreso	int4(0)	No	Vía de ingreso por la cual llegó a las fila de la UJC
Id_Organiz	int4(0)	No	Organización que desarrolló el proceso de ingreso a la fila de la militancia
Id_municipio	int4(255)	Yes	Identificador del municipio al que pertenece.
Id_Provincia	int4(0)	Yes	Identificador de la provincia ha la que pertenece.
es_servicio_militar	bool(0)	No	Si es o no servicio militar.
esc_poli_curs	bool(0)	No	Si ha cursado o no escuelas políticas.
esc_adm	bool(0)	No	Guarda la escuela.
Ciudadanía	int4(0)	Yes	Guarda la ciudadanía.

Tabla 6.5 Descripción de la entidad "Ciudadanía"

Nombre: Tb_nCiudadania			
Descripción: Nomenclador que guarda los tipos de ciudadanía que existen.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_ciudadania	int4(0)	No	Identificador principal.
nombre	varchar(255)	No	Nombre de la ciudadanía.

Tabla 6.6 Descripción de la entidad "Tipo Sanción Externa"

Nombre: Tb_nSanc_Exter			
Descripción: Nomenclador que guarda los tipos de sanciones externas existentes			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_nSanc_Ext	int4(0)	No	Identificador principal
Nombre	text(0)	No	Nombre del tipo de sanción externa

Tabla 6.7 Descripción de la entidad "Tipo Sanción Interna"

Nombre: Tb_nSanc_Inter			
Descripción: Nomenclador que guarda los tipos de sanciones internas existentes			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**

Id_nSanc_Int	int4(0)	No	Identificador principal
Nombre	text(0)	No	Nombre del tipo de sanción interna

Tabla 6.8 Descripción de la entidad "Provincia"

Nombre: Tb_nProvincia			
Descripción: Nomenclador que guarda todas las provincias del país			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Provincia	int4(0)	No	Identificador principal
Provincia	text(255)	No	Nombre de la provincia

Tabla 6.9 Descripción de la entidad "Cotización"

Nombre: Tb_dCotizacion			
Descripción: Guarda información referente a la Cotización realizada por los militantes mensualmente.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_Cotizacion	int4(0)	No	Identificador principal
anno	date(0)	No	Año de la cotización
Id_meses	int4(0)	No	Referencia al mes de la Cotización
Id_Militante	int4(0)	No	Referencia al identificador del militante

Tabla 7.1 Descripción de la entidad "Evaluación"

Nombre: Tb_dEvaluacion			
Descripción: Guarda las evaluaciones de los militantes.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_evaluacion	int4(0)	No	Identificador principal
fecha	date(0)	No	Fecha en la cual se realizó la evaluación
slminto	text(0)	No	Señalamiento a mejorar
recmda	text(0)	No	Recomendaciones
Id_Militante	int4(0)		Referencia al identificador del Militante

Tabla 7.2 Descripción de la entidad "Nivel Cultural"

Nombre: Tb_nNivel_Cultural			
Descripción: Nomenclador que guarda todos los distintos niveles culturales			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Nivel_Cultural	int4(0)	No	Identificador principal
nombre	varchar(255)	No	Nombre del nivel cultural

Tabla 7.3 Descripción de la entidad "Municipio"

Nombre: Tb_nMunicipio			
Descripción: Nomenclador que guarda todos los municipios del país			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_municipio	int4(255)	No	Identificador principal
Id_Provincia	int4(0)	No	Referencia a la provincia a la cual pertenece
Municipio	text(255)	No	Nombre del municipio

Tabla 7.4 Descripción de la entidad "Organización que Desarrollo el Proceso"

Nombre: Tb_nOrzganizacion_Des_Proceso			
Descripción: Guarda la información de todas las organizaciones capacitadas para desarrollar un proceso			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Organiz	int4(0)	No	Identificador de la organización
nombre	varchar(255)	No	Nombre de la organización

Tabla 7.5 Descripción de la entidad "Meses"

Nombre: Tb_nMeses			
Descripción: Nomenclador que guarda los meses.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_meses	int4(0)	No	Identificador principal.
meses	text(0)	No	Nombre del mes.

Tabla 7.6 Descripción de la entidad "Proceso Militante"

Nombre: Tb_rPuede_Tner_Pcso_dM_dP			
Descripción: Guarda información referente a los procesos de crecimiento al PCC realizados a los Militantes menores de 30			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Proceso	int4(0)	No	Referencia al identificador proceso
Es_deseado	bool(0)	No	Expresa a través de true o false si está de acuerdo a que se le realice el proceso
Es_Avalado	bool(0)	No	Expresa a través de true o false si es avalado para realizarle el proceso.
Id_Militante	int4(0)	No	Referencia al identificador del militante

Tabla 7.7 Descripción de la entidad "Militante_Telefono"

Nombre: Tb_dMilitante_Tb_Telefono
Descripción: Guarda información de la relación entre teléfonos y militantes

Atributo	Tipo	¿Acepta Nulos?	Descripción.
Id_Militante	int4(0)	No	Identificador principal
id_telefono	int4(0)	No	Identificador principal

Tabla 7.8 Descripción de la entidad "Historial del Militante"

Nombre: Tb_dHistorial			
Descripción: Historial de la vida de los militantes.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_historial	int4(0)	No	Identificador principal
Id_Militante	int4(0)	Yes	Identificador del militante
fecha	date(0)	No	Fecha en que se hace el historial
nombre	text(0)	No	Nombre del militante
papellido	text(0)	No	Primer apellido del militante
sapellido	text(0)	No	Segundo apellido del militante
ci	varchar(255)	No	Carné de identidad del militante
id_acc	int4(0)	Yes	Identificador de la acción realizada.

Tabla 7.9 Descripción de la entidad "Teléfono"

Nombre: Tb_dTelefono			
Descripción: Guarda la información referente a los teléfonos de cada militante			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_telefono	int4(0)	No	Identificador principal
número	varchar(255)	No	Número telefónico
Id_tipo_telef	int4(0)	No	Referencia al nombre del local al cual pertenece el teléfono

Tabla 8.1 Descripción de la entidad "Doble Militante"

Nombre: Tb_dDoble_Militante			
Descripción: Guarda la información referente a los dobles militante de la UJC			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Militante	int4(0)	No	Identificador del militante
fecha_ingreso	date(0)	No	Fecha de ingreso al PCC

Tabla 8.2 Descripción de la entidad "Arribante"

Nombre: Tb_dArribante
Descripción: Hace una especificación de todo aquel militante que tiene edad de 30 años

Atributo	Tipo	¿Acepta Nulos?	Descripción.
Id_Arribante	int4(0)	No	Referencia del identificador del Militante

Tabla 8.3 Descripción de la entidad "Proceso"

Nombre: Tb_dProceso			
Descripción:			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Proceso	int4(0)	No	Identificador principal
Es_Incdo	bool(0)	No	Expresa a través de true o false si el proceso es iniciado o no
Es_aprobado	bool(0)	No	Expresa a través de true o false si el proceso es aprobado o no
Princi_vol	bool(0)	Yes	Expresa a través de true o false si se acogió al principio de voluntariedad

Tabla 8.4 Descripción de la entidad "Proceso"

Nombre: Tb_rRealiza_Pcso_dM_dArrb			
Descripción: Guarda información referente del procesos realizados a cada arribante			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Arribante	int4(0)	No	Referencia al Identificador del Arribante(PK)
Id_Proceso	int4(0)	No	Referencia al identificador del Proceso(PK)
Es_Deseado	bool(0)	No	Expresa a través de true o false si desea que se le realice el proceso
Es_Avalado	bool(0)	No	Expresa a través de true o false si está avalado

Tabla 8.5 Descripción de la entidad "Tipo de Teléfono"

Nombre: Tb_nTipo_Telef			
Descripción: Nomenclador donde se guarda los distintos tipos de locales al que pertenece el teléfono.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_tipo_telef	int4(0)	No	Identificador principal
nombre	varchar(255)	No	Nombre del local

Tabla 8.6 Descripción de la entidad "PCC"

Nombre: Tb_PCC			
Descripción: Guarda información referente a militantes de la UJC que llegan al PCC			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**

Id_PCC	int4(0)	No	Identificador principal
fecha	date(0)	No	Fecha de llegada al PCC
condicion_doble	bool(0)	No	Expresa a través de true o false si posee la condición de doble militante
Id_Proceso	int4(0)	No	Referencia al identificador del proceso por el cual llegó al PCC

Tabla 8.7 Descripción de la entidad "Acción a Realizar"

Nombre: Tb_nAccion			
Descripción: Nomenclador que guarda las acciones existentes.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
id_acc	int4(0)	No	Identificador principal
nombre	text(0)	No	Fecha de llegada al PCC

Tabla 8.8 Descripción de la entidad "Cargo"

Nombre: Tb_dCargo			
Descripción: Guarda los cargos que ha ocupado el militante.			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_Militante	int4(0)	No	Identificador del militante
cargo	varchar(255)	No	Cargo.
anno	date(0)	No	Año en que fue ocupado

Tabla 8.9 Descripción de la entidad "Control de Cambio"

Nombre: Tb_dControl_Cambio			
Descripción: Historial de los Comités			
Atributo	**Tipo**	**¿Acepta Nulos?**	**Descripción.**
Id_control	varchar(31)	No	Llave primaria.
campo	varchar(50)	No	Nombre del campo que sufre cambio
v_old	varchar(10)	No	Valor viejo del campo
v_new	date(0)	No	Valor nuevo del campo
Id_Auditoria	time(7)	No	Llave foránea

2.10 Análisis de optimización de querys

En Bases de Datos, una consulta es el método para acceder a los datos, a través de las consultas se puede modificar, borrar, mostrar y agregar datos en una Base de Datos. Para esto se

utiliza un lenguaje de manipulación de datos (DML – Data Manipulation Language). Las consultas se pueden expresar de diferentes formas alcanzándose el mismo propósito para lo cual fue diseñado, de ahí que algunas sean más óptimas que otras en dependencia de la lógica empleada y el plan de ejecución.

Cuando hablamos de **optimización de consultas** nos referimos a mejorar los tiempos de respuestas en un sistema de gestión de Base de Datos, pues la optimización es el proceso de encontrar el mejor plan de ejecución con exclusiva eficiencia.

Para el análisis de la optimización de consultas de la BD propuesta se tuvo en cuenta algunos criterios y se definieron algunos parámetros para un adecuado rendimiento:

- Todas las tablas de la BD están normalizadas hasta la tercera forma normal, permitiendo que no exista duplicidad en los datos, contribuyendo a un almacenamiento más sólido y eficiente.
- Se tuvo en cuenta una sintaxis para la identificación de tablas: Tb_nombreTabla.
- Se utilizó el número de índice que postgreSQL admite por defecto; no siendo necesario la creación de nuevos índices.

Las consultas para la aplicación no se van a realizar en el mismo gestor de BD, aprovechando la posibilidad que brinda la capa de abstracción utilizada por Symfony de poder cambiar la aplicación a otro SGDB en cualquier momento del desarrollo del proyecto. Se recomienda que a la hora de confeccionar las consultas se tenga en cuenta algunos criterios buena optimización de consultas.

Por Ejemplo:

- En la medida de lo posible hay que evitar que las sentencias SQL estén embebidas dentro del código de la aplicación. Es mucho más eficaz usar vistas o procedimientos almacenados por que el gestor los guarda compilados. Si se trata de una sentencia embebida el gestor debe compilarla antes de ejecutarla.
- Seleccionar exclusivamente aquellos que se necesiten
- No utilizar nunca SELECT * porque el gestor debe leer primero la estructura de la tabla antes de ejecutar la sentencia
- Si utilizas varias tablas en la consulta, especificar siempre a qué tabla pertenece cada campo, le ahorras al gestor el tiempo de localizar a que tabla pertenece el campo.

- La cláusula DISTINTIC es costosa de ejecutar porque generalmente involucra un ordenamiento de las tuplas restantes para eliminar duplicados, por lo que es necesario analizar si realmente es favorable su utilización.
- Las condiciones de JOIN se pueden evaluar mas eficientemente comparando índices primarios, aunque es preferible evaluar una condición de igualdad numérica que una condición de igualdad de caracteres.

A continuación se muestran algunos ejemplos de optimización de consultas realizadas en la BD de la UJC.

Ejemplo:

SELECT public.tb_dpersona.pmer_nombre, public.tb_dcomite.nombre

FROM public.tb_dpersona, public.tb_dcomite

WHERE (public.tb_dpersona.ci = '01225706418')

AND (public.tb_dpersona.id_comite = public.tb_dcomite.id_comite)

2.11 Conclusiones

En este capítulo se describió la arquitectura utilizada y la forma en que se accederá a los datos. Se realizó una descripción de las clases persistentes, así como de las entidades que conforman la base de datos, explicándose de forma detallada cada uno de sus atributos y tipo. También se analizó el proceso de optimización de consultas, mostrándose algunas de ellas y dándose a conocer un conjunto de heurísticas a tener en cuenta para una mejor optimización de consultas.

CAPÍTULO #: 3 VALIDACIÓN DEL DISEÑO REALIZADO

3.1 Introducción

En este capítulo se describe la validación teórica y funcional del diseño realizado. Se exponen algunos aspectos que se tuvieron presentes para realizar el diseño de Base de Datos, tales como: la integridad de los datos, la normalización, la no redundancia de la información y la seguridad. Además se realiza un análisis de las consultas más utilizadas en el estudio de las pruebas de cargas intensivas.

3.2 Validación teórica del diseño

3.2.1 Integridad de los datos

Consiste en garantizar la no contradicción entre los datos almacenados de modo que, en cualquier momento del tiempo, los datos almacenados sean correctos, es decir, que no se detecte inconsistencia entre los datos. Está relacionada con la minimización de la redundancia, ya que es más fácil garantizar la integridad si se elimina la redundancia.

En el mundo real existen ciertas reglas que deben cumplir los elementos en él existentes, por ejemplo una persona debe tener uno y sólo un CI (Carné de identidad).Cuando diseñamos una Base de Datos deseamos que ésta refleje lo más fielmente posible el universo del discurso que se estaba recogiendo.

La semántica y la integridad de los datos son conceptos que en la Base de Datos suelen ir asociados, aunque no son idénticos. La integridad de los 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 entidad, de dominio, referencial, integridad de usuario.

- Integridad de entidad

La integridad de entidad define una fila como entidad única para una tabla determinada. La integridad de entidad exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones UNIQUE, o restricciones PRIMARY KEY.

- La integridad de dominio

La integridad de dominio viene dada por la validez de las entradas de los datos para una columna determinada. Se puede exigir la integridad de dominio para restringir el tipo mediante tipos de

datos, el formato mediante reglas y restricciones CHECK, o el intervalo de valores posibles mediante restricciones FOREIGN KEY, definiciones DEFAULT, definiciones NOT NULL. En el caso de de la BD para UJC de la UCI se definió el carné de identidad de tipo de dato varchar y solo acepta cadenas de longitud 11, su dominio seria de varchar (11), pero como el tipo de dato acepta números, caracteres y letras, fue necesario restringir mediante una restricción de verificación (**check)** que solo aceptara números, haciendo uso de una expresión regular :

CI **varchar** (11) **check** (CI SMILAR **TO '**%(1|2|3|4|5|6|7|8|9|0)')

Las restricciones de verificación (**check)**, siempre van ligadas a un único elemento de la base de datos. También existen los disparadores, estos son de gran importancia si se quiere especificar una acción distinta al rechazo si se cumple una determinada restricción o cuando la condición se hace verdadera, los disparadores pueden interpretarse como reglas de tipo evento-condición-acción, un ejemplo de ellos se utilizó en esta base de datos para validar la inserción de un comité de acuerdo a la estructura predefinida, es decir, se creó un disparador que verificara antes de insertar un comité, si este estaba correctamente subordinado de acuerdo a su tipo de comité, por ejemplo: se valida que no se pueda insertar un CP(comité primario) y un CBI(comité de base independiente) sin antes haber insertado un CUJC(comité UJC), además verifica que de acuerdo al tipo de comité sea la subordinación , pues tanto un CP como un CBI sólo se puede subordinar a un CUJC , mientras que un CBN (comité de base normal), se subordina solamente a un CP.

- Integridad referencial

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

La integridad referencial protege las relaciones definidas entre las tablas cuando se crean o se eliminan filas. Se basa en las relaciones entre claves externas y claves principales, donde los valores de la clave ajena de la relación hija deben corresponderse con los valores de las claves primarias de la relación padre, garantizando que los valores de claves sean coherentes en las distintas tablas. Para conseguir esa coherencia en la base de datos de la UJC de la UCI, se precisó que no existieran referencias a valores inexistentes, se definió las opciones de borrado y modificación en las claves ajenas, al cambiar el valor de una clave, todas las referencias a ella se cambian en consecuencia en toda la base de datos. Un ejemplo de esto lo constituye entre comité y personas pues al modificar el

identificador del comité se modifica todas las tuplas relacionadas en la relación persona. Existen varias formas para la operación de actualización (borrado o modificación) esta son:

B: C Borrado en Cascada

B: N Borrado con puesta a nulos

B: D Borrado con puesta a valor por defecto.

B: R Borrado restringido.

M: C Modificación en Cascada.

M: N Modificación con puesta a nulos

B: D Modificación con puesta a valor por defecto.

B: R Modificación restringido.

- Integridad definida por el usuario

La integridad definida por el usuario permite definir reglas de empresa específicas que no pertenecen a ninguna otra categoría de integridad. Todas las categorías de integridad admiten la integridad definida por el usuario. Esto incluye todas las restricciones de nivel de columna y nivel de tabla en CREATE TABLE, procedimientos almacenados y desencadenadores. Un ejemplo de una de esta regla lo constituye que para crear un comité de base ya sea independiente o normal es preciso tener 3 militantes como mínimo asociado a este comité.

3.2.2 Normalización de la Base de Datos

"La teoría de normalización consiste en obtener esquemas relacionales que cumplan unas determinadas condiciones y se centra en las determinadas Formas normales. Se dice que un esquema de relación está en una determinada forma normal si satisface un conjunto determinado de restricciones. "(5)

- Primera forma normal (1FN)

La primera forma normal elimina los grupos repetitivos al establecer que todos los dominios de la relación contienen valores atómicos. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna.

- Segunda forma normal (2FN)

Una relación está en segunda forma normal si ya está 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria.

Para pasar una relación en 1FN a 2FN hay que eliminar las dependencias parciales de la clave primaria. Para ello, se eliminan los atributos que son funcionalmente dependientes y se ponen en una nueva relación con una copia de su determinante (los atributos de la clave primaria de los que dependen).

- Tercera forma normal (3FN)

Una relación está en tercera forma normal si ya está en 2FN y, además, cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria. La dependencia $x \rightarrow y$ es transitiva si existen las dependencias $x \rightarrow y$, $y \rightarrow z$ siendo x, y, atributos o conjuntos de atributos de una misma relación.

Para pasar una relación de 2FN a 3FN hay que eliminar las dependencias transitivas. Para ello, se eliminan los atributos que dependen transitivamente y se ponen en una nueva relación con una copia de su determinante (el atributo o atributos no clave de los que dependen).

Explicación de la normalización de la BD

La base de datos de la UJC se encuentra normalizada hasta la tercera forma normal

3.2.3 Análisis de redundancia de información

La redundancia es el almacenamiento repetitivo de datos en diferentes lugares de la base de datos o ficheros. Puede provocar problemas como: un incremento del trabajo, desperdicio de espacio de almacenamiento, inconsistencia de datos; es por ello que se deben crear bases de datos con un mínimo de redundancia, ya que es imposible evitarla en un 100% en proyectos grandes. Aunque a veces es considerada por cuestiones de rendimiento y controladas por el sistema, pudiéndose utilizar para ello disparadores. Teniéndose en cuenta de que no existía una BD para la gestión de la información en la UJC de la UCI, mucha de las informaciones que se manipulaba eran redundantes, se hacían copias de ficheros de información en diferentes lugares, que dificultaba la actualización de los datos, dando lugar a una mayor inconsistencia de los mismos, con la realización de esta base de datos gran parte de esta redundancia se verá resuelta.

3.2.4 Análisis de la seguridad de la base de datos

Hoy en día la información se ha convertido en el activo más importante de cualquier organización ya sea académica o con intereses lucrativos. Debido a esto la seguridad informática ha fundamentado su razón de ser en sus principios básicos: integridad, confidencialidad y disponibilidad.

Para el desarrollo de este trabajo se utilizó el DBMS (Sistema Gestor de Base de Datos) PostgresSQL para administrar los datos y disminuir el riesgo de la información.

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

- ✓ Se recomienda instalar el directorio de datos separado de la instalación debido a que se tiene una mejor organización a la hora de administrar los datos y los archivos.
- ✓ Protección de los ficheros de la base de datos. Todos los ficheros almacenados en la base de datos están protegidos contra escritura por cualquier cuenta que no sea la del súper usuario de PostgreSQL
- ✓ La conexiones de los clientes se pueden restringir por directorios IP y /o por nombres de usuarios mediante el fichero pg_hba.conf situado en PG_DATA.
- ✓ Las conexiones de los clientes pueden ser autentificadas mediante otros paquetes externos.
- ✓ A cada usuario de PostgreSQL se le asigna un nombre de usuario y (opcionalmente) una contraseña. Por defecto, los usuarios no tienen permisos de escrituras a bases de datos que no hayan creado.
- ✓ Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse de acuerdo a esos grupos.
- ✓ Se debe realizar copias de seguridad de las bases de datos regularmente.
- ✓ No se recomienda confiar en los sistemas de copia de seguridad del sistema para las copias de respaldo de las bases de datos.

Para la seguridad de la solución propuesta se recomienda aislar el directorio de datos, preferentemente en una partición por separado por si se llegara a dar el caso poder agregar espacio a la partición con mayor facilidad, la creación de un usuario que actúe como administrador del SGBD y que no sea el que brinda PostgreSQL por defecto “Postgres”. También se debe configurar, de acuerdo con las políticas de accesos permitidas, las conexiones locales y remotas mediante el archivo de configuración pg_hba.conf

De inicio contiene algo como lo siguiente:

```
# TYPE DATABASE    USER        CIDR-ADDRESS          METHOD

# IPv4 local connections:

#host    all         all         127.0.0.1/32          md5

# IPv6 local connections:

#host    all         all         : 1/128               md5
```

Esta configuración que trae por defecto solo permite tener acceso mediante conexión local, es por ello que lo primero que se hace es modificar estos permisos de acceso por defecto, para ello se debe tener en cuenta que representa cada uno de las 5 columnas que se muestran

Type: Es aquí donde se representa el tipo de conexión que se usará, local y remota.

Database: Es el nombre de la base de datos a la que podemos conectarnos, si se requiere se pueden poner todas se especifica con la palabra reservada all.

User: Es el usuario que tendrá acceso a la(s) base(s) de datos que le sea permitido. De igual forma se hace referencia a todos con all.

Cidr−address: Es la columna donde pone el IP y la netmask que puede acceder al servidor. Para el caso de una conexión local se deja en blanco.

Method: El método de autenticación especifica el tipo de autenticación que el servidor debería usar para un usuario que intenta conectar a PostgreSQL. Existen varios métodos posibles a utilizar

Para un mejor control de la base de datos, se propone que el usuario administrador de PostgreSQL se conecte solamente de forma local a la base de datos con el método de autentificación md5. Se incluiría además las restricciones de conexión necesarias de los clientes dependiendo de las reglas del negocio. Mediante PostgreSQL se puede restringir el acceso a una determinada base de datos por medio de dirección IP en específico o grupo de usuarios, mediante el archivo de configuración pg_ident.conf. Así como también permite configurar dentro de la BD los privilegios de acceso a las tablas, utilizando para ello los comandos GRAN y REVOKE.

3.2.5 Trazabilidad de la acciones

La trazabilidad es la habilidad de trazar o dejar huellas de los movimientos y procesos por los que pasa un determinado producto, 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.

Con vista a tener un control de los movimientos de traslados de los militantes en la BD, se modeló la tabla Tb_rtraslado, donde se registra la entrada y salida de cada militante en un comité. Además de acuerdo a la necesidad de tener constancia de todas las altas y bajas realizadas por un comité en una fecha determinada se modeló las tablas Tb_movimiento, Tb_movimiento_alta y Tb_movimiento_baja , las cuales registran las trazas de cada acción llevada a cabo por los Comités de Bases y Comités de Bases Independientes. También queda un registro de los Comités de Bases y Comités de Bases Independiente que existió.

3.3 Validación funcional

Para el llenado voluminoso e inteligente de la BD se utilizó como herramienta EMS Data Generator For PostgreSQL 2005, siendo de gran utilidad para generar los datos de pruebas a varias tablas a la misma vez. Esta herramienta da la posibilidad de escoger las tablas a las cuales se quiere llenar. Se realizó un registro de 20 000 tuplas para la tabla militante , 15 000 para universo juvenil , así como también otras tantas tuplas para las restantes. Se definió para cada campo el rango de valores admitidos en dependencia de la cantidad de tuplas a insertar, cumpliendo en todo momento las pruebas de carga intensiva y escalabilidad propuesto.

A continuación se muestra algunas de las consultas realizadas:

```
SELECT
  COUNT (public.tb_dmilitante.id_militante) AS cantidad
FROM
 public.tb_nestructura
 INNER JOIN public.tb_dcomite ON
(public.tb_nestructura.id_estructura=public.tb_dcomite.id_estructura)
 INNER JOIN public.tb_dpersona ON (public.tb_dcomite.id_comite=public.tb_dpersona.id_comite)
 INNER JOIN public.tb_dmilitante ON (public.tb_dpersona.id_persona=public.tb_dmilitante.id_militante)
Where public.tb_dmilitante.es_activo = true and public.tb_dcomite.id_comite = 1;
```

Esta consulta devuelve la cantidad de militantes activos que pertenecen a un Comité determinado.

3.4 Conclusiones

A través de este capítulo se han analizado una serie de aspectos de gran importancia para la propuesta de modelación de la BD. Se concluyó con un análisis de la consistencia de la misma, así como la integridad de sus datos, la normalización y las distintas formas de seguridad que se pueden adoptar para el control de la información.

CONCLUSIONES

En nuestros días el almacenamiento de los datos en la UJC nacional resulta un gran problema, debido a que se hace de forma manual, por lo que no existe calidad en la prestación de servicios a los clientes de esta organización, es por eso que surgió este trabajo de investigación y modelación de una base de datos capaz de eliminar dicho inconveniente.

Para el buen desarrollo de este trabajo se utilizaron herramientas multiplataforma y pertenecientes a la familia del software libre, dadas las características y condiciones en el mundo de la informática en la actualidad, entre las que se encuentran:

- El uso de PostgreSQL como SGBD.
- El uso de Symfony como framework de desarrollo web conjuntamente con php5.
- El uso de Visual Paradigm como herramienta CASE.

Se debe agregar que se obtuvieron los diseños de una Base de Datos relacional en Tercera Forma Normal, y se analizó la integridad y la redundancia de información, lográndose finalmente cumplir con el objetivo propuesto, pues cuenta con todos los requisitos y se logra guardar la información con la mayor seguridad posible.

Recomendaciones

Se recomienda darle seguimiento a este trabajo para finalmente llegar a la integración de la base de datos con una aplicación web y que así el cliente pueda aprovechar los servicios brindados por el diseño propuesto.

Otra recomendación válida es que una vez obtenida una aplicación se debe poner a disposición de todas las UJC del país.

Por último se propone establecer una política de seguridad ante fallos y recobrado de la información en la BD. Se pueden generar de manera automática backups periódicos así como la realización de tareas de mantenimiento.

REFERENCIA BIBLIOGRAFÍCA

1. **Chávez, Lic. Elsa Carolina Esparza.** clases de diseño de bases de datos. *fase de diseño conceptual.* [En línea] [Citado el: 16 de 2 de 2008.] http://disenobd.blog.com/1100262/.

2. **Rubén Rodriguez.** Comunicacion. *Bases de Datos(La Historia).* [En línea] [Citado el: 6 de 1 de 2008.] http://www.fudim.org/comunicacion/notas/nota.php?id=22&a=Adim.

3. **IBERCON.** ibercom.com. *¿Que modelos de Bases de datos existen?* [En línea] [Citado el: 4 de 2 de 2008.] https://www.ibercom.com/soporte/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=30.

4. **Cruz, Raúl Uranga.** monografias.com. *bases de datos.* [En línea] [Citado el: 8 de 2 de 2008.] http://www.monografias.com/trabajos12/basdat/basdat.shtml#COMPON.

5. **MasterMagazine.** MasterMagazine.com. *Definicóin de Normalización.* [En línea] [Citado el: 11 de 4 de 2008.] http://www.mastermagazine.info/termino/6105.php.

BIBLIOGRAFÍA

1. **Sanchez, María A. Mendoza.** *Informatizate.net.* [En línea] 7 de 2004. http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

2. **Pozo, Salvador.** Ficheros en C y C++. [En línea] [Citado el: 21 de 5 de 2008.] http://www.conclase.net/c/ficheros/index.php?cap=003.

3. **Medina, José Manuel Alarcón.** *Administracion SGBD postgreSQL.* [En línea] 11 de 2006. [Citado el: 12 de 3 de 2008.] http://www.gvpontis.gva.es/fileadmin/conselleria/images/Documentacion/migracionSwAbierto/SITARGES/manual.pdf.

4. **Espinoza, Humberto.** *PostgreSQL una alternativa de DBMS Open Sourse.* [En línea] 2005. [Citado el: 23 de 5 de 2008.] http://www.lgs.com.ve/pres/PresentacionES_PSQL.pdf.

5. **Chávez, Lic. Elsa Carolina Esparza.** clases de diseño de bases de datos. *fase de diseño conceptual.* [En línea] [Citado el: 16 de 2 de 2008.] http://disenobd.blog.com/1100262/.

6. **Cruz, Raúl Uranga.** monografias.com. *bases de datos.* [En línea] [Citado el: 8 de 2 de 2008.] http://www.monografias.com/trabajos12/basdat/basdat.shtml#COMPON.

7. **Andrés, María Mercedes Marqués.** *arquitectura de los sistemas de bases de datos.* [En línea] [Citado el: 18 de 11 de 2007.] http://www3.uji.es/~mmarques/f47/apun/node33.html.

8. —. [En línea] [Citado el: 23 de 4 de 2008.] http://www3.uji.es/~mmarques/f47/apun/node1.html.

9. **Alvarez, Sara.** desarrolloweb.com. *Modelos de bases de datos.* [En línea] [Citado el: 14 de 1 de 2008.] http://www.desarrolloweb.com/articulos/modelos-base-datos.html.

10. **EMS Database Management Solutions, Inc.** *sharewareconnection.* [En línea] 8 de 3 de 2008. http://www.sharewareconnection.com/ems-data-generator-2005-for-postgresql.htm.

11. *Proyecto S.O.B.L. Traducciones. .* [En línea] 1 de 3 de 2008. http://www.sobl.org/traducciones/practical-postgres/node19.html .

12. *Proyecto S.O.B.L. TRaducciones.* [En línea] 3 de 3 de 2008. http://www.sobl.org/traducciones/practical-postgres/node12.html .

13. *Programacion en castellano.* [En línea] 5 de 7 de 2005. [Citado el: 2 de 6 de 2008.] http://www.programacion.net/noticia/1363/.

14. *PstgreSQL.* [En línea] 27 de 2 de 2008. http://es.tldp.org/Postgresql-es/web/navegable/todopostgresql/intro.html.

15. *MySQL.com.* [En línea] 7 de 3 de 2008. http://dev.mysql.com/doc/refman/5.0/es/index.html.

16. *MySQL.* [En línea] 6 de 3 de 2008. http://www.netpecos.org/docs/mysql_postgres/x57.html .

17. Monografias.com. *Bases de datos .* [En línea] http://www.monografias.com/trabajos11/basda/basda.shtml.

18. Monografias.com. *Bases de Datos.* [En línea] [Citado el: 1 de 8 de 2008.] http://www.monografias.com/trabajos11/basda/basda.shtml.

19. *Monografias.com.* [En línea] 22 de 2 de 2008. http://www.monografias.com/trabajos24/herramientas-case/herramientas-case.shtml .

20. Monografias. [En línea] [Citado el: 3 de 1 de 2008.] http://www.monografias.com/trabajos7/arch/arch.shtml.

21. Monografia. [En línea] [Citado el: 16 de 12 de 2007.] http://www.monografias.com/trabajos6/sistar/sistar.shtml.

22. **MasterMagazine.** MasterMagazine.com. *Definicóin de Normalización.* [En línea] [Citado el: 11 de 4 de 2008.] http://www.mastermagazine.info/termino/6105.php.

23. **IBERCON.** ibercom.com. *¿Que modelos de Bases de datos existen?* [En línea] [Citado el: 4 de 2 de 2008.] https://www.ibercom.com/soporte/index.php?_m=knowledgebase&_a=viewarticle&kbarticleid=30.

24. **fabFORCE.net.** *Fabforce.net.* [En línea] 8 de 2003. [Citado el: 27 de 11 de 2007.] http://www.fabforce.net/dbdesigner4/.

25. *Diseño de bases de datos.* [En línea] 20 de 2 de 2008. http://www3.uji.es/~mmarques/f47/apun/node68.html .

26. **Rubén Rodriguez.** Comunicacion. *Bases de Datos(La Historia).* [En línea] [Citado el: 6 de 1 de 2008.] http://www.fudim.org/comunicacion/notas/nota.php?id=22&a=Adim.

GLOSARIO DE TÉRMINOS

UJC: Unión de Jóvenes Comunistas de Cuba.

FEU: Federación Estudiantil Universitaria.

Sistema Gestor de Bases de Datos (SGBD): Tipo de software específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Lenguaje Estructurado de Consultas (SQL): Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

IPv4-IPv6: Protocolos de comunicación que identifican con un número único las computadoras conectadas a Internet o intranet corporativa.

Java Database Connectivity (JDBC): Marco de programación para los desarrolladores de aplicaciones Java que tienen acceso a la información guardada en bases de datos, hojas de cálculos o archivos planos.

Open Database Connectivity (ODBC): Estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation cuyo objetivo es hacer posible el acceso a los datos desde cualquier aplicación , sin importar el SGDB que almacena la información.

Lenguaje de Definición de Datos (DDL): Es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.

Licencia GPL: Proteger la libre distribución, modificación y uso de software, declarar que el software cubierto por esta licencia es software libre.

Licencia BSL: Licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Pertenece al grupo de licencias de software Libre. Permite el uso del código fuente en software no libre a diferencia de la GPL.

Cardinalidad: El cardinal indica el número o cantidad de los elementos constitutivos de un conjunto.

Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

GLOSARIO DE TÉRMINOS

Query: Consiste en una cadena de consulta que se utiliza para : Insertar, Actualizar o Editar valores de la Bases de Datos.

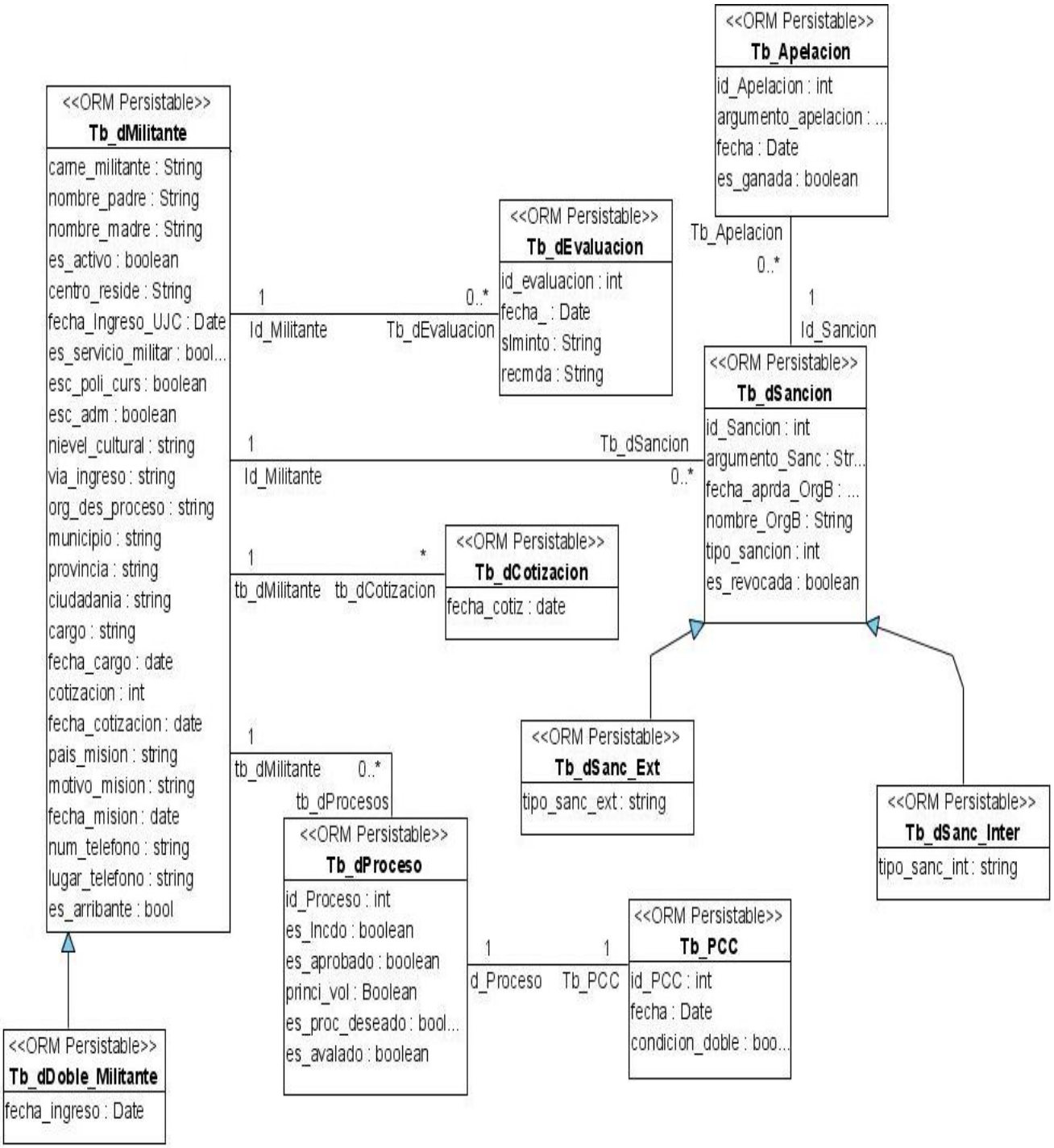
Tablas: Tipo de modelado de datos, donde se guardan los datos recolectados por un programa. Están compuestas por dos estructuras: campo y registros.

Tuplas: Es una secuencia ordenada de objetos; una función que asocia unívocamente los nombres con algunos valores.

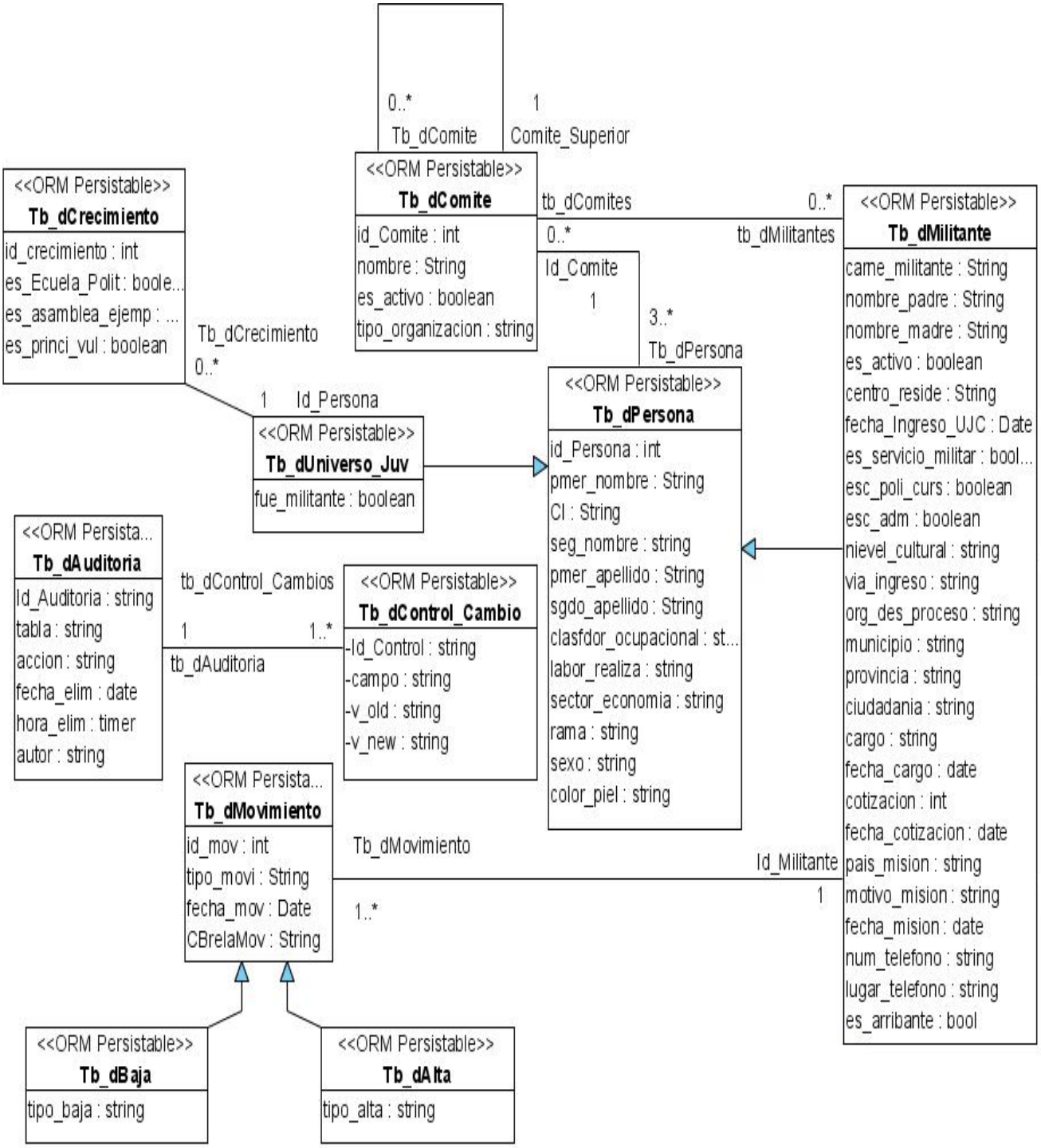
Llave primaria: Es un conjunto de atributos de una relación que son escogidos para identificar una tupla.

ANEXO

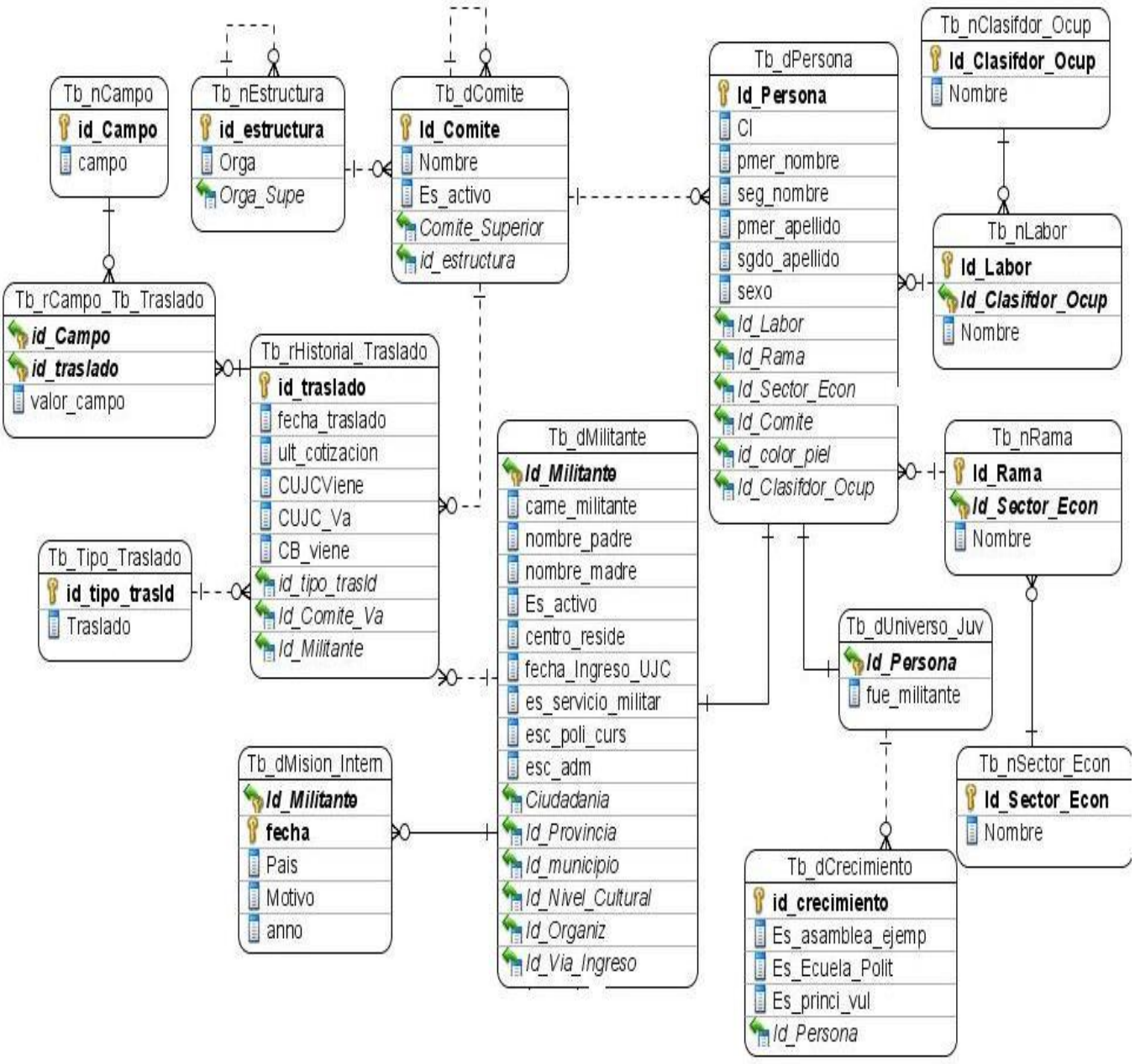
Anexo1. Diagrama de clases persistente



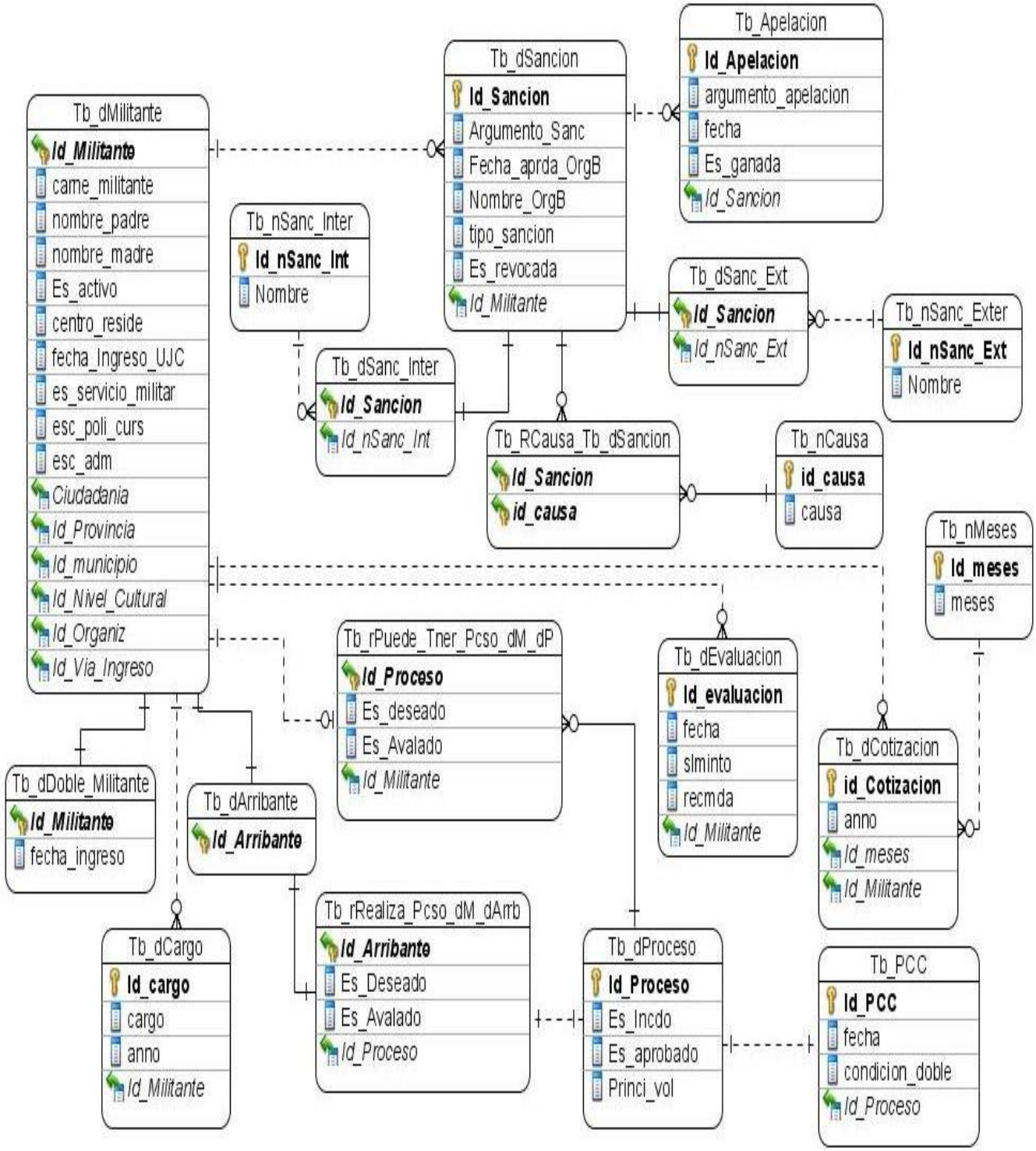
Anexo2. Diagrama de clases persistente



Anexo3. Diagrama de entidad-relación



Anexo4. Diagrama de entidad-relación



Anexo5. Diagrama de entidad-relación