Universidad de las Ciencias Informáticas

Dirección de Informatización

Propuesta de diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0

Akademos
Gestión Académica

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Mariliennys Gloria Mesquida Peña

Eduardo Basulto Uribe

Tutores: Ing. Andry Suárez Hernández

Ing. Roberto Llerena Villar

Ciudad de La Habana, Cuba

Mayo 2009

"Año del 50 Aniversario de la Revolución"

"La educación empieza con la vida, y no acaba sino con la muerte, el cuerpo siempre es el mismo, y decae con la edad; la mente cambia sin cesar, y se enriquece y perfecciona con los años. Pero las cualidades esenciales del carácter, lo original y enérgico de cada hombre, se deja ver desde la infancia en un acto, en una idea, en una mirada..."

José Julián Martí Pérez.

Declaración de Autoría

Declaramos que somos los únicos autores del trabajo titulado: Propuesta de diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0 y autorizamos 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 _____.

Mariliennys Gloria Mesquida Peña

_____

Firma del Autor

Eduardo Basulto Uribe

_____

Firma del Autor

Ing. Andry Suárez Hernández

_____

Firma del Tutor

Ing. Roberto Llerena Villar

_____

Firma del Tutor

Datos de Contacto

Nombre y Apellidos: Roberto Llerena Villar.

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: rllerena@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el curso 2007-2008, miembro desde hace varios años del grupo de desarrollo del proyecto Akademos. Actualmente es profesor de programación de la facultad 1. Ha participado además en varios eventos científicos en representación de su proyecto, en los cuales han obtenido varios resultados relevantes.

Nombre y Apellidos: Andry Suárez Hernández.

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: ashernandez@uci.cu

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el curso 2007-2008, miembro desde hace varios años del grupo de desarrollo del proyecto Akademos en el que se desempeña como el Arquitecto principal. Actualmente es profesor de programación de la facultad 1. Ha participado en varios eventos científicos en representación de su proyecto con resultados relevantes en dichos eventos.

Agradecimientos

De Mariliennys:

Primero que todo agradecerle a la Revolución y a Fidel por darme la oportunidad de estudiar en esta maravillosa Universidad.

A mi mamá y mi papá pues este siempre fue su sueño, por todo su sacrificio, su apoyo incondicional y su infinito amor. Este trabajo también es de ellos pues a ellos les debo lo que soy.

A mi abuela querida por siempre estar ahí para mí, por la educación que me dio durante toda mi vida y por quererme tanto.

A mi novio "tito" que siempre estuvo a mi lado durante toda la carrera, por su amor y apoyo incondicional durante todos los años que llevamos juntos.

A Eduar que estuvo siempre a mi lado, que me crió como si fuera su hija y con quien siempre he podido contar para lo que necesite.

A mi tía Nivia pues siempre me ha dado toda su confianza y su cariño, por quererme como si fuera su propia hija.

A mi tío Yody por su confianza y su cariño.

A mis hermanos pues por ellos trato de ser mejor cada día, son lo que me inspira a salir adelante.

A mis tutores por su apoyo y ayuda en todo momento.

Por último quiero agradecer a todas las personas con las cuales he tenido la dicha de compartir estos maravillosos años que nunca olvidaré.

A todos los que me ayudaron de una u otra forma a realizar este trabajo…

Muchas gracias

De Eduardo:

En primer lugar a mi madre por creer en mí, por su apoyo incondicional y su infinito amor, porque este es su sueño hecho realidad pues lo que soy se lo debo a ella. Muchas gracias mami…

A mi hermano porque él me inspira cada día a ser un hombre mejor, por quererme y apoyarme durante todos los años de mi vida, por ser para mí un ejemplo a seguir, te quiero.

A mi papá, por toda la confianza que ha depositado en mí…

A mi novia, a la cual le debo parte de mi carrera pues en ella siempre he tenido una compañera, una amiga… por quererme tanto y apoyarme en todo momento, a ella le agradezco con todo mi amor.

A Dayron "el pinareño bobo", al "negro" Osmery, Ricardo, Dannys y a Fily con los cuales he compartido momentos inolvidables de mi carrera, mis amigos, mis compañeros, mis hermanos…

A mis suegros por apoyarme y compartir conmigo buenos momentos en estos años de alegría y felicidad, además por hacerme sentir como uno más de la familia, gracias.

A toda mi familia por siempre quererme tanto.

A mis tutores Roberto y Andry por su apoyo, su confianza y su amistad.

A todos aquellos que me han ayudado de una forma u otra a hacer este sueño realidad…

Muchas Gracias

Dedicatoria

A mis padres que son mi fuente de inspiración,

A mi abuela que es la luz de mi vida,

A mi novio "tito" por su amor y apoyo incondicional,

A mis hermanos, a Eduar, mi tía Nivia y mi tío Yody, en fin, a toda mi familia por el cariño y apoyo que me han brindado siempre….

Mariliennys

A mi madre querida por ser lo más grande que tengo en la vida,

A mi hermano por ser mi fuente de inspiración,

A mi novia por estar siempre a mi lado,

A mi padre por su apoyo y cariño…

Eduardo

A todos los que con su amor han hecho de nosotros una mejor persona cada día, a todos ellos va dedicado este trabajo por todo el apoyo y cariño que siempre nos han brindado…

Eduar y Mary

Resumen

El grupo de desarrollo del sistema Akademos se encuentra trabajando en la segunda versión del software. Se han incorporado nuevas funcionalidades para la gestión académica, y además se ha reestructurado el área de base de datos con el objetivo de mejorar las deficiencias que posee la primera versión. Se ha detectado después de un profundo estudio que la base de datos aumenta considerablemente su volumen cada año por la entrada de nuevos estudiantes a la Universidad, pues los datos de los egresados y los estudiantes que provocan baja del centro se mantienen en la base de datos central[1], esto afecta el rendimiento del servidor y la velocidad de respuesta del mismo. Como solución a esta problemática se realizó el diseño de la Base de Datos Histórica de la gestión académica del sistema Akademos v_2.0 con el objetivo de mejorar la organización de los datos separando los históricos de los activos y de esta forma mejorar los problemas de rendimiento del servidor de la base de datos central del sistema. Durante la realización de este trabajo se estudiaron varios sistemas de gestión que aportaron experiencias novedosas a la investigación y además, se investigaron varias herramientas que son utilizadas en el área de base datos en el tratamiento de los datos históricos, y finalmente se obtuvo un modelo que se ajusta a las necesidades del sistema, el cual ha sido validado teórica y funcionalmente teniendo en cuenta un conjunto aspectos importantes para el área de las bases de datos relacionales.

[1] base de datos central: Es la base de datos general del sistema Akademos v_2.0 donde está recogida la información de los estudiantes y profesores de la Universidad.

Índice

Índice de tablas

Índice de figuras

Introducción

Los sistemas de información existen desde las primeras civilizaciones. Los datos se recopilaban, se estructuraban, se centralizaban y se almacenaban convenientemente. El objetivo inmediato de este proceso era poder recuperar estos mismos datos u otros datos derivados de ellos en cualquier momento sin necesidad de volverlos a recopilar, lo que solía ser el paso más costoso o incluso irrepetible. Desde la antigüedad ha sido necesario almacenar grandes cantidades de datos, es por ello que surge lo que en la actualidad se conoce como base de datos, que no es más que una colección de datos recopilados y estructurados que existe durante un período de tiempo. [1]

Desde el surgimiento de la informática hasta la actualidad se han desarrollado numerosos sistemas de gestión con el objetivo de informatizar la sociedad y a su vez resolver los problemas existentes en cuanto a organización y almacenamiento de grandes volúmenes de datos.

El desarrollo vertiginoso de la industria de software se ha convertido en un tema crítico en el mundo actual, es por ello que la Revolución Cubana ha apostado fuerte por la informatización, pues ha reconocido la importancia estratégica de la informática para la sociedad.

La Universidad de las Ciencias Informáticas (UCI) como una universidad surgida por la necesidad de integrar a nuestro país al avance tecnológico mundial, se ha visto enfrascada desde sus inicios en la tarea de desarrollar la incipiente industria de software en Cuba. Se han logrado cumplir varios de sus objetivos fundamentales como el de informatizar muchos procesos que se llevan a cabo dentro y fuera de la Universidad, uno de los más importantes para cualquier centro de estudio es la gestión académica; razón por la cual surge el software Akademos.

Este sistema es ampliamente utilizado pues en el mismo se gestiona toda la información referente a los procesos docentes de la Universidad. En la base de datos del sistema se encuentran registrados todos los estudiantes junto a sus evaluaciones, certificados, fotos y otros documentos, así como información de los profesores del centro.

En base a las necesidades de migrar a software libre y con el objetivo de obtener un producto más genérico y desplegable en cualquier centro universitario, el equipo de proyecto trabaja en una nueva

versión del software, donde se le añaden funcionalidades a los módulos ya existentes y se incorporan otros nuevos.

Después de realizar un estudio detallado de la base de datos de la primera versión y de su interacción con el sistema de manera general, se concluyó que presenta problemas de rendimiento y escalabilidad, debido a: una implementación en un corto período de tiempo, los constantes cambios y el gran volumen de información que se almacena [2]. Esta es la causa fundamental de que se proponga, para la segunda versión del sistema, un nuevo diseño de la base de datos que se adapte a las nuevas funcionalidades y requerimientos. Sin embargo, los datos históricos, que no son más que los datos poco consultados o datos pasivos, se propone, en este nuevo modelo de datos general del sistema, que sean guardados en otra base de datos para de esta forma descentralizar la cantidad de datos y así lograr un mejor y más rápido acceso a la información histórica del sistema.

Akademos brinda una gran cantidad de servicios, uno de los más utilizados son los reportes de información por parte de los directivos de la Universidad. Actualmente se gestionan en una sola base de datos, los datos de todos los estudiantes, incluyendo los egresados y los estudiantes que provocan baja de la Universidad. Esta situación trae consigo que se consuman más recursos cuando se realizan consultas y la sobrecarga del servidor que conlleva a la disminución del rendimiento y de la velocidad de respuesta del mismo. Por estas razones se ha decidido realizar la Base de Datos Histórica de la gestión académica de Akademos v_2.0, y además, para tener una mejor organización de la información histórica del sistema y, por consiguiente, un acceso más simple y rápido a la misma.

Luego de un análisis de la problemática anterior y con el fin de solucionar estas necesidades se propone el **problema científico** de esta investigación expresado con la siguiente interrogante: ¿Cómo realizar el diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0 para mejorar el rendimiento del modelo de datos general de sistema?

Para este problema se define como **objeto de estudio**: la base de datos de Akademos v_2.0, y como **campo de acción**: los datos históricos de la gestión académica de la base de datos de Akademos v_2.0.

Se define como **objetivo general** de la investigación:

- Diseñar la Base de Datos Histórica para la gestión académica de Akademos v_2.0.

A partir de un análisis del objetivo general se derivaron los siguientes **objetivos específicos**:

- Identificar y seleccionar los datos históricos de la gestión académica dentro de la base de datos de Akademos v_2.0.
- Definir y describir las herramientas para modelar la Base de Datos Histórica de la gestión académica de Akademos v_2.0.
- Generar los diagramas y documentación necesaria para obtener el diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0.

Para dar cumplimiento al objetivo anteriormente expresado, se deben cumplir las siguientes **tareas científicas:**

1. Estudiar el diseño de la base de datos de Akademos v_1.0 y v_2.0.
2. Realizar el estudio del arte sobre las alternativas más utilizadas actualmente a nivel nacional e internacional sobre software vinculadas al tratamiento de datos históricos.
3. Realizar el estudio de las principales herramientas utilizadas en el manejo de datos históricos.
4. Realizar el modelo lógico de la Base de Datos Histórica de la gestión académica de Akademos v_2.0.
5. Realizar el modelo físico de la Base de Datos Histórica de la gestión académica de Akademos v_2.0.
6. Validar el diseño realizado.

Para guiar la investigación se plantea la siguiente **idea a defender**:
La realización del diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0 permitirá un mejor rendimiento del modelo de datos general del sistema y descentralizar la cantidad de datos para lograr un acceso más simple y rápido a la información histórica.

Variables:

Variable independiente: - Base de Datos Histórica.

Variable dependiente: - rendimiento del modelo de datos general del sistema.

Métodos Científicos:

La investigación está sustentada en los métodos teóricos y los métodos empíricos, que facilitarán la investigación y servirán de guía para organizar mejor el trabajo y de esta forma posibilitar entender el problema, estudiarlo, analizarlo y llegar a conclusiones para la solución. A continuación se explica la utilización de los que han sido seleccionados:

Métodos teóricos:

- Método Analítico – Sintético: Este método será utilizado para realizar un análisis de la base de datos de Akademos v_1.0 y v_2.0, para identificar los datos históricos que se manejan en ella.
- Método Análisis Histórico – Lógico: Este método será utilizado en el estudio y análisis de las principales herramientas utilizadas para el manejo de datos históricos.
- Inductivo – Deductivo: Este método será utilizado en el momento de diseñar la Base de Datos Histórica, pues se debe definir un nuevo modelo de datos que cumpla con los requisitos necesarios.

Métodos Empíricos:

- Entrevista**:** Permitirá la coordinación de reuniones con el cliente final y el jefe del proyecto Akademos, para aprobar, según el levantamiento de requisitos realizado, los datos que debe almacenar la Base de Datos Histórica.

Estructura del contenido

La investigación consta de 3 capítulos, en los cuales se desarrollan aspectos de importancia para lograr el objetivo que se persigue.

Capítulo #1: Fundamentación teórica

- Estudio de varios sistemas de gestión de información tanto en el ámbito nacional como internacional. Tratamiento de los datos históricos dentro de estos sistemas.
- Definir herramientas y metodologías que se utilizarán en la propuesta de solución.

Capítulo #2: Análisis y descripción de la solución propuesta

- Estrategia de integración con los módulos que componen el sistema.
- Arquitectura de la propuesta de solución.
- Requisitos, tanto funcionales como no funcionales, que se tienen en cuenta en el desarrollo de la propuesta de solución.
- Diagramas que muestran el diseño de la solución, además de la descripción de cada una de las tablas que los componen.

Capítulo #3: Validación del diseño realizado

- Se valida el diseño realizado de forma teórica teniendo en cuenta aspectos como la integridad, la normalización, la seguridad, entre otros.
- Se valida funcionalmente el diseño realizado mediante la aplicación de pruebas de carga y estrés al mismo.

Capítulo # 1

Fundamentación Teórica

1.1 Introducción

En el presente capítulo se abordan los conceptos claves de esta investigación relacionados con base de datos y los sistemas gestores de bases de datos. Se realiza, además, un estudio del estado del arte de los sistemas de gestión de información, herramientas y metodologías utilizadas en el mundo vinculadas al área de base de datos y al tratamiento de datos históricos, en este caso, se hace referencia al almacenamiento de los datos poco consultados o pasivos pero que tienen relevancia en el sistema y por tanto no pueden ser eliminados del mismo.

1.2 Ámbito Internacional

Se estudiaron varios sistemas de gestión en el ámbito internacional, pero el que más se ajusta a la investigación es el SIU.

1.2.1 SIU: Sistema de Información Universitaria

El SIU desarrolla soluciones informáticas y brinda servicios para el Sistema Universitario Nacional de Argentina. Su objetivo es contribuir a mejorar la gestión de las instituciones, permitiéndoles contar con información segura, íntegra y disponible, optimizar sus recursos y lograr que el software sea aprovechado en toda su potencialidad. El SIU forma parte de la Secretaría de Políticas Universitarias (SUP) del Ministerio de Educación, Ciencia y Tecnología de la Argentina**.**

Soluciones informáticas por área de aplicación:

Nivel gerencial:

Data Warehouse: Apoyo a la Gerencia Universitaria. Herramientas de *Data Warehouse* para el análisis de información en niveles gerenciales (desgranamiento, seguimiento de docentes, evolución de matrícula por

carrera, ejecución presupuestaria, servicios, evolución de liquidaciones de haberes, recursos humanos de planta, etc.).

SIU-Wichi: Sistema de consultas gerenciales vía Web. Ofrece consultas sobre información producida en la gestión (información contable y de personal). Su plasticidad permite la incorporación de nuevas consultas en base a información de la institución.

Data Mining: Herramienta de *Data Warehouse* para el análisis de los datos recopilados por las universidades a través de los sistemas de gestión, con el objetivo de estudiar las realidades y necesidades específicas de cada institución, con mecanismos que aseguran la confidencialidad de los datos.

Nivel académico:

SIU-Araucano: Sistema de información estadística universitaria. Contiene la información estadística de alumnos nuevos inscriptos, regulares y egresados de las universidades e institutos universitarios.

SIU-Guaraní: Sistema de gestión académica. Realiza la gestión de alumnos, desde la matriculación hasta el egreso, complementándose con gestión de aulas, mesas de exámenes, jurados, etc. posee una interfaz con el sistema de estadísticas de alumnos SIU-Araucano.

SIU-Kolla: Sistema de seguimiento de graduados. Encuesta que le permite a las universidades conocer, diagnosticar y evaluar los perfiles de los egresados que forma, relevando información sobre su inserción laboral, su relación con la universidad y otros datos relevantes.

SIU-Tehuelche: Sistema de gestión de becas universitarias. Utiliza SIU-Toba[2] como infraestructura de desarrollo, lo cual permitirá que cada universidad pueda realizar adaptaciones o personalizaciones al sistema.

Dentro de los sistemas ya mencionados, es objeto de estudio en el marco teórico de esta investigación el SIU-Guaraní: Sistema de gestión académica.

El sistema está diseñado con una arquitectura Cliente-Servidor. Se utilizó Power Builder 7 como herramienta de desarrollo para la parte cliente e Informix IDS 9.x como servidor de base de datos. La

[2] SIU-Toba: El SIU-Toba es un entorno de desarrollo Web creado por el SIU con la finalidad de disponer de una herramienta adecuada para la construcción de sistemas transaccionales de mediana y alta complejidad. El mismo está basado en un conjunto de herramientas libres ampliamente difundidas: Apache, PHP y PostgreSQL.

interfaz Web está desarrollada en PHP. Gran parte de las reglas de negocio están escritas en forma de procedimientos almacenados dentro de la base de datos.

El SIU-Guaraní es un sistema de gestión de alumnos que registra y administra todas las actividades académicas de la universidad, desde que los alumnos ingresan como aspirantes hasta que obtienen el diploma. Pasando por un proceso de matriculación, el registro de materias cursadas y de resultados académicos, los pedidos de equivalencia y la gestión del egresado.

Como características fundamentales del sistema podemos señalar que utiliza una única base de datos para la actualización, registro y consulta de información; maneja la captura y consulta de datos en forma descentralizada y además, promueve la estandarización de procesos y datos para un mejor aprovechamiento de la información.

El sistema cuenta con los siguientes módulos: Gestión de carreras y planes, Planificación, Gestión de matrícula, Gestión de cursado, Gestión de aulas, Gestión de exámenes, Gestión de equivalencias, Gestión de egresados, Administración y emisión de certificados, Mensajerías a casillas de e-mail o celulares y Gestión de encuestas para alumnos.

Dentro de las funcionalidades más importantes que el sistema brinda se podría mencionar las consultas a los planes de estudio y sus asignaturas, a las notas, a la asistencia. Un estudiante puede inscribirse o darse de baja en algún curso o materia, entre otras.

1.3 Ámbito Nacional

1.3.1 RN: Registros y Notarías

El proyecto RN surgió con el objetivo de estandarizar la gestión de las oficinas de Registros y Notarías, para garantizar la certeza, confiabilidad y seguridad jurídica de la República Bolivariana de Venezuela.

El sistema permite gestionar de forma eficiente todas las actividades relativas a registros mercantiles e inmobiliarios: atención al cliente, gestión documental, gestión administrativa y contable, comunicaciones con las diferentes instituciones, organismos públicos y entidades a través de servicios.

Posee, además, una alta integración entre los diferentes módulos, así como una interfaz gráfica común que facilita la comprensión y uso por parte de los usuarios. En general abarca los siguientes aspectos: Gestión Administrativa / Contable, Gestión de Recursos Humanos, Gestión Documental, Digitalización de

Documentos, Reportes y Estadísticas, Administración y Configuración, Control de Prohibiciones, Gerencia y Portal Web.

Dentro de las herramientas de desarrollo de la aplicación para oficinas que se utilizan se pueden citar Visual Studio .NET 2003, TierDeveloper 4.0 y PLSQL Developer 6.3. Para el desarrollo de la base de datos se utilizó para el centro de datos Oracle y como gestor de base de datos el SqlExpress 2005 para los servidores locales.

En RN existe una única base de datos, los datos históricos son tablas de historial que son creadas para separar estos datos del resto, sin embargo, no utilizan ninguna base de datos adicional ni otra herramienta para dividir los históricos de los activos en el sistema.

1.3.2 CTAISC: Centro de Tratamiento y Análisis de Información de Seguridad Ciudadana

Este sistema surge de la necesidad de diseñar e implantar el Centro de Tratamiento y Análisis de Información de Seguridad Ciudadana, a través del cual se permita:

- Disponer de la información relacionada con la seguridad ciudadana a nivel nacional de forma normalizada e integrada.
- Contar con una herramienta de base científica y tecnológica para la recolección, observación y análisis de la información requerida para la planificación de estrategias y la formulación de políticas de manera proactiva.
- Realizar el seguimiento y control de las acciones ejecutadas, con el fin de medir su efectividad y los resultados e impactos sobre los ciudadanos y la sociedad.

CTAISC cuenta con varias aplicaciones informáticas:

- SINSEC: Sistema de Información Nacional de Seguridad Ciudadana.
- Portal WEB: Portal Web institucional del Centro de Tratamiento y Análisis de Información de Seguridad Ciudadana.
- SIGEPOL: Sistema de Gestión Policial.
- SIGESC: Sistema de Gestión de Emergencias de Seguridad Ciudadana.

Dentro de estas aplicaciones la que está destinada al análisis de la información histórica es SINSEC.

El Sistema de Información Nacional de Seguridad Ciudadana o SINSEC, constituye una herramienta que garantiza la integración y el análisis multidisciplinario de la información generada por los diferentes Órganos de Seguridad Ciudadana de la República Bolivariana de Venezuela, para elevar la efectividad de las estrategias y políticas diseñadas por el gobierno, y así mejorar los niveles de seguridad ciudadana.

Está conformado por un Almacén de Datos (*Data Warehouse*) de Seguridad Ciudadana donde se encuentran disponibles a los analistas (usuarios), los datos operacionales de los diferentes Órganos de Seguridad u otros organismos para el análisis y toma de decisiones. Cuenta con los datos que brinda el Cuerpo de Investigaciones Científicas, Penales y Criminalísticas (CICPC), el Instituto Nacional de Estadística (INE), y la Coordinación Nacional de Ciencias Forenses (CNCF).

El SINSEC sustenta la capacidad de análisis y tratamiento de la información altamente eficiente y validada, orientada a estudiar la naturaleza y causas de la inseguridad, con vistas a disminuir el índice delictivo y de esta forma contribuir de forma objetiva a la mejora del sentir ciudadano en este sentido. Los datos operacionales integrados, pasan a ser información disponible para el análisis y la toma de decisiones en manos de analistas, expertos y altos ejecutivos del Ministerio del Poder Popular para las Relaciones Interiores y de Justicia.

1.3.3 CICPC: Cuerpo de Investigaciones Científicas, Penales y Criminalísticas

El proyecto CICPC surge de la necesidad de mejorar la aplicación que existía en Venezuela, el SIIPOL antiguo: Sistema Integrado de Información Policial, el mismo presentaba muchas limitaciones, solo manejaba los datos no el proceso en sí, y además era muy obsoleto; estas fueron las principales causas que provocaron que se desarrollara la nueva aplicación SIIPOL: Sistema de Investigación e Información Policial que maneja un mayor número de información, abarca más áreas del CICPC, agiliza el proceso de investigación y es más segura.

El sistema permite gestionar de forma eficiente todas las actividades relativas a las investigaciones policiales. Dentro de los principales módulos que forman parte de la solución se encuentran: Investigación Forense, Investigación Penal, Investigación Criminalística, Gestión Administrativa, Análisis de Información y otros. Existe una alta integración entre cada uno de ellos pues todos forman parte importante de cualquier investigación policial que se inicie.

Dentro de las herramientas utilizadas en su desarrollo se puede señalar que como lenguaje de programación fue utilizado Java y como gestor de base de datos Oracle. El proyecto maneja una gran cantidad de datos históricos, es por ello que fue necesario crear un esquema completamente desnormalizado dentro de la base de datos que permitiera modelar la información histórica, y de esta forma mejorar la organización de la información y el rendimiento del servidor con el uso de varios *Tablespaces* aplicados a toda la base de datos, incluyendo este esquema.

1.3.4 Akademos

Akademos es un sistema informático cuya principal misión es el control de todos los procesos que intervienen en la gestión académica de un centro de estudios universitarios. Surge como respuesta a la necesidad de sustentar y dar soporte en la Universidad de las Ciencias Informáticas (UCI) a toda la labor del personal de secretaría, con la visión de obtener un producto genérico, capaz de ser aplicable y adaptable en cualquier centro que implemente el control docente universitario. Fue diseñado y desarrollado por un equipo de estudiantes del Plan CUJAE y la UCI, rectorados por la Dirección de Informatización de la UCI y bajo la tutoría del ingeniero Emil Lima Valdés. Los primeros módulos liberados fueron: Matrícula y Plan de Estudio, a pocos meses de haber comenzado su desarrollo y, posteriormente, se fueron incorporando los demás módulos que actualmente conforman el sistema: Expediente, Profesor, Estudiante, Control Docente y Reportes [3].

Akademos v_1.0 actualmente presta servicios en la UCI como el sistema de gestión académica de la Universidad, sin embargo, al mismo tiempo se desarrolla una nueva versión del software. Esto se debe a que los requerimientos, tanto funcionales como no funcionales del sistema, se han modificado algunos y otros se han incorporado, además, esta versión no constituye un producto por lo que no puede ser comercializada, por estas razones la Dirección de Informatización de la UCI y el equipo de desarrollo del proyecto han decidido migrar este sistema a software libre y así lograr un producto libre de licencias y trabas impuestas por los desarrolladores de software propietario.

Al sistema acceden todos los estudiantes de la Universidad, más de mil profesores, personal de secretaría y directivos del centro. Akademos brinda información a aplicaciones externas como son el Entorno Virtual de Aprendizaje, el Sistema de Control de Acceso, y los directorios de personas.

La base de datos de la primera versión de Akademos es capaz de gestionar la información relacionada con los procesos de gestión académica que el sistema ha automatizado, no obstante, en esta base de

datos existen problemas de rendimiento y escalabilidad, por estas razones se plantea una nueva estructura para la base de datos de la segunda versión del sistema con el objetivo de lograr un mejor rendimiento del servidor, y no sobrecargarlo de datos. Además, cada una de las facultades regionales tiene una base de datos con el mismo modelo que la versión que actualmente se utiliza en la UCI, sin embargo, no existe un sistema de réplicas que le permita al centro mantener una copia de cada una de las bases de datos de estas facultades para poder acceder más fácilmente a la información generada en cada una de ellas.

La nueva estructura estará compuesta por una base de datos central, una base de datos para cada facultad regional y una Base de Datos Histórica donde se almacenará la información pasiva o poco consultada del sistema; mediante esta solución se logra acceder de manera más simple y rápida a la información histórica sin utilizar grandes recursos cuando se realicen las consultas. También será implementado un sistema de réplicas que permitirá mantener una copia de las bases de datos de las facultades regionales en la base de datos central del sistema.

1.4 Aporte obtenido del estudio de los sistemas de gestión de información

El estudio realizado de varios sistemas de gestión y el tratamiento a los datos históricos que cada uno de ellos en particular realiza, ha servido para adquirir experiencias y poder llegar a una solución para la propuesta de diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0. Como esta investigación ha sido realizada para mejorar las deficiencias del sistema Akademos en cuanto al manejo de los datos históricos, se describen las principales funcionalidades y problemas que la primera versión del software presenta y algunas propuestas para solucionarlos en esta nueva versión. Las soluciones aplicadas en cada sistema han sido analizadas: estas no se ajustan a las necesidades del sistema**,** dada las particularidades del mismo y dado el objetivo fundamental que se persigue**,** el de mejorar en cuanto a organización de la información histórica y rendimiento del servidor; sin embargo, estas soluciones han realizado importantes aportes que han contribuido a la realización de la propuesta de solución.

1.5 Herramientas y Metodologías utilizadas en el manejo de datos históricos

1.5.1 Base de Datos

La base de datos es un componente fundamental de un sistema de información. El escenario actual de la tecnología de bases de datos es resultado del perfeccionamiento que ha tenido lugar en el procesamiento de datos y en la gestión de la información, por lo cual se hace necesario que su concepto y características fundamentales queden recogidos en esta primera parte investigativa.

Según Dr. Pérez Márquez Graells, 1999 (última revisión: 9/06/03): "las bases de datos, proporcionan unos datos organizados, en un entorno estático, según determinados criterios, y facilitan su exploración y consulta selectiva. Se pueden emplear en múltiples actividades como por ejemplo: seleccionar datos relevantes para resolver problemas, analizar y relacionar datos, extraer conclusiones, comprobar hipótesis… Una base de datos es una recopilación de datos o información relacionados entre sí, la misma está compuesta por filas que son los registros y por columnas que son los campos."

El término base de datos ha sido definido de varias maneras. Algunas de ellas:

- "Colección de datos interrelacionados." [Elmarsi, R, Navathe, S.B.1989].
- "Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una base de datos puede considerarse una colección de datos variables en el tiempo." [García, 1999].
- "Colección no redundante de datos que son compartidos por diferentes programas de aplicación." [Howe, 1983].
- "Conjunto de datos de la empresa memorizados en un ordenador, que es utilizado por numerosas personas y cuya organización está regida por un modelo de datos." [Flory, 1982].
- "Una base de datos es un conjunto de información almacenado en memoria auxiliar que permite acceso directo y un conjunto de programas que manipulan esos datos." [Decups, 2002].
- "Una base de datos es un conjunto exhaustivo, no redundante de datos estructurados, organizados independientemente de su utilización y su implementación en la máquina, accesible en tiempo real y compartible por usuarios concurrentes que tienen necesidad de información diferente y no predecible en el tiempo." [Decups, 2002].

- "Colección o depósitos de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes a estas, y su definición y descripción, únicas para cada tipo de datos, han de estar almacenados junto con los mismos. Los procedimientos de actualización y recuperación, comunes y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de los datos." [Decups,2002] [4].

Según la variabilidad de los datos almacenados se pueden encontrar bases de datos estáticas y dinámicas. Las bases de datos estáticas son de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones mientras que las bases de datos dinámicas son aquellas donde la información almacenada se modifica con el tiempo, que permite operaciones como actualización y adición de datos y operaciones fundamentales de consulta.

1.5.2 Data Warehouse

Uno de los primeros autores en escribir sobre el tema de los almacenes de datos fue Bill Inmon, este define un almacén de datos (*Data Warehouse*) en términos de las características del repositorio de datos:

- Orientado a temas.- Los datos en la base de datos están organizados de manera que todos los elementos de datos relativos al mismo evento u objeto del mundo real queden unidos entre sí.
- Variante en el tiempo.- Los cambios producidos en los datos a lo largo del tiempo quedan registrados para que los informes que se puedan generar reflejen esas variaciones.
- No volátil.- La información no se modifica ni se elimina, una vez almacenado un dato, éste se convierte en información de sólo lectura, y se mantiene para futuras consultas.
- Integrado.- La base de datos contiene los datos de todos los sistemas operacionales de la organización, y dichos datos deben ser consistentes [5].

Un almacén de datos contiene datos que son necesarios o útiles para una organización, se utiliza como un repositorio de datos para posteriormente transformarlos en información útil para el usuario. Un almacén de datos debe entregar la información correcta a la persona indicada en el momento óptimo y en el formato adecuado. El almacén de datos da respuesta a las necesidades de usuarios expertos, utilizando Sistemas

de Soporte a Decisiones (DSS), Sistemas de Información Ejecutiva (EIS) o herramientas para realizar consultas o informes. Los usuarios finales pueden hacer fácilmente consultas sobre sus almacenes de datos sin tocar o afectar la operación del sistema.

La tecnología de almacenes de datos integra las técnicas de bases de datos y las técnicas de análisis de datos. Es fundamental señalar que son de mucha utilidad en empresas en las que es necesario hacer análisis de tendencias y crecimientos de ventas o la producción de un determinado artículo donde los datos se extraen y se cargan de varias bases de datos para su estudio, sin embargo, siguen siendo ineficientes en el modelado de los sistemas de gestión académica como Akademos, precisamente porque en estos almacenes que se utilizan en el manejo de enormes cantidades de datos, no se tiene en cuenta un cliente en específico sino un perfil de ese cliente, para tratar dicha información y trazar estrategias a partir de los resultados. En el caso específico de esta investigación, de cada estudiante es necesario mantener todos sus datos personales por interés de la Dirección de la Universidad, junto al índice académico y otros datos específicos, por lo que no es factible la implementación de un almacén de datos.

1.5.3 Ventajas de las Bases de Datos Relacionales

El modelo de bases de datos relacional ofrece varias ventajas importantes dentro de las se pueden citar:

1- Independencia estructural: Los cambios en la estructura de la base de datos relacional no afectan, de ninguna forma, el acceso a los datos del Sistema Gestor de Bases de Datos (SGBD), es así como el modelo relacional logra la independencia estructural.

2- Diseño, ejecución, administración y uso más fácil de la base de datos: Como el modelo relacional logra al mismo tiempo independencia de los datos e independencia estructural, es más fácil diseñar la base de datos y administrar su contenido.

3- Capacidad de consultas: Una de las razones por las que el modelo relacional tiene una posición dominante en el mercado es su muy poderosa y flexible capacidad de consulta. En la mayoría del software de la base de datos relacional, el lenguaje de consulta es un lenguaje estructurado (SQL, por sus siglas en inglés). Este lenguaje le permite al usuario especificar qué debe hacer sin tener que decir cómo se debe hacer. El RDBMS (Sistema Administrador de Bases de Datos Relacionales por sus siglas en inglés) utiliza SQL para transformar la consulta del usuario en el código tecnológico requerido para recuperar los datos solicitados, por tanto, la base de datos relacional requiere menos programación que otras bases de datos

o ambientes. El mantenimiento del sistema de la base de datos es la tarea crucial que el RDBMS maneja, en gran medida, sin el conocimiento del usuario final.

1.5.4 Gestor de Base de Datos

Ante la notable demanda de soluciones informáticas para la progresiva informatización de todas las organizaciones con la necesidad de optimizar servicios y productos, han surgido diferentes gestores de bases de datos; estos son programas que permiten manejar la información de modo sencillo y que prestan servicios para el desarrollo y el manejo de bases de datos.

Los Sistemas Gestores de Bases de Datos (SGBD) ofrecen un control centralizado de la información teniendo como objetivos evitar la redundancia de los datos, mejorar los mecanismos de seguridad de los mismos y la privacidad, mantener la integridad de los datos realizando las validaciones necesarias y mejorar la eficacia del acceso a los datos.

Los SGBD pueden definirse como un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos a los datos y actuando de interfaz entre los datos físicos y el usuario. Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad. [6].

Desde 1970 y hasta 1999 la tecnología de gestión de datos ha evolucionado con el surgimiento y desarrollo de los SGBD que no son más que un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Actualmente se ha extendido el uso de los SGBD. Las bases de datos son el soporte del sistema de información de las organizaciones y además son diseñadas para dar respuesta (eficiente) a las funciones básicas de la organización (gestión, producción,…).

El propósito general de los gestores de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos. Los principales objetivos de los gestores de base de datos son:

- Abstracción de la información: Los gestores ahorran a los usuarios detalles acerca del almacenamiento físico de los datos.

- Independencia: Consiste en la capacidad de modificar el esquema físico o lógico de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Respaldo y recuperación: Los gestores deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.
- Seguridad: La información almacenada en una base de datos puede llegar a tener un gran valor, los gestores deben garantizar que esta información se encuentre segura frente a usuarios malintencionados, que intenten acceder a la misma con el objetivo de manipularla o destruirla. Los gestores de base de datos cuentan con un complejo sistema de permisos y grupos de usuarios, que permiten otorgar diversas categorías de permisos. [7].

Hoy en día existe un número significativo de gestores, cada uno con características distintas, así como ventajas y desventajas en el momento de ser utilizados, los principales gestores libres son: PostgreSQL y MySql, ambos muy populares. En la investigación es utilizado el PostgreSQL, este gestor ha sido seleccionado por la Dirección de Informatización de la UCI y la dirección del proyecto por sus características, las que se describen a continuación junto a otros datos del mismo.

1.5.4.1 PostgreSQL

PostgreSQL está considerado como uno de los gestores de bases de datos de código abierto más avanzados del mundo. Es básicamente una herramienta Cliente/Servidor para la gestión de bases de datos. Se considera uno de los SGBD más completos del cual destaca su soporte de transacciones, estabilidad, escalabilidad, además es multiplataforma. PostgreSQL tiene soporte para lenguajes procedurales internos que son aquellos en los cuales el usuario instruye al sistema para que lleve a cabo una serie de operaciones en la base de datos con el fin de calcular el resultado deseado y están fundamentados en la utilización de variables para almacenar valores y en la realización de operaciones con los datos almacenados.

PostgreSQL es un gestor de base de datos relacional orientada a objetos, ya que incluye características como: la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

El único costo asociado a PostgreSQL es el de conocerlo, este posee licencia BSD[3], permitiendo la libertad de uso, modificación y distribución en productos comerciales o no comerciales. PostgreSQL tiene todo lo que se exige de un gestor de base de datos relacional, de manera general las principales características del gestor donde se refleja el por qué fue elegido por el equipo de desarrollo para la nueva versión del sistema son:

- La implementación SQL de Postgre se sujeta fuertemente a los estándares ANSI-SQL 92/99 (y últimamente se han añadido características del SQL2003). Es altamente escalable; tanto en lo que se refiere a cantidad de datos que puede manejar así como, al número de usuarios concurrentes que puede acomodar. Soporta la mayoría de los tipos de datos de los estándares SQL92 y SQL99 (integer, numeric, boolean, char, varchar, date, interval, timestamp, entre otros).
- Diseñado para entornos de gran volumen de información y alta concurrencia. Utiliza la tecnología MVCC (Acceso Concurrente Multiversión, por sus siglas en inglés), que sirve para lograr un control de concurrencia tan eficiente que generalmente no se requiere de bloqueos.
- Características para la integridad de los datos, tales como: claves primarias, llaves foráneas con capacidad de actualizar en cascada o restringir la acción en caso que existan hijos, restricción check, restricción de unicidad y restricción not null; todas las restricciones se pueden postergar hasta el momento de terminar la transacción.
- Resistencia a fallas. Escritura adelantada de registros (WAL, sus siglas en inglés) para evitar pérdidas de datos en caso de fallos por: energía, sistema operativo, hardware.
- Cumple la prueba ACID (*Atomicity* (atomicidad), *Consistency* (consistencia), *Integrity* (integridad), *Durability* (durabilidad)) y tiene soporte completo para llaves foráneas, joins, vistas, subconsultas (incluyendo subconsultas en la cláusula FROM), *triggers*, y procedimientos almacenados en varios lenguajes.

[3] La **licencia BSD** es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

- Un sofisticado optimizador de consultas, que es capaz de resolver consultas complejas en tiempos comparables a los de los mejores SGBD (una última adición en este campo es la habilidad de usar varios índices para resolver consultas sobre una misma tabla).
- Posee soporte para utilizar *Tablespace*s (ubicaciones alternativas para los datos).
- Al igual que con los tipos de datos, PostgreSQL permite la declaración de funciones propias.
- Posee un buen soporte para *triggers* y procedimientos en el servidor.
- Posee soporte para el uso de índices, reglas y vistas.
- Además corre en la mayoría de los sistemas operativos más utilizados, Linux, varias versiones de UNIX, BeOS y Windows.

Posee productos que lo complementan, y que ayudan a mejorar su desempeño, eficiencia y manipulación:

- PgCluster y Slony-I: Utilizados en la replicación de datos, el primero en réplicas multi-maestro y el segundo en réplicas maestro - esclavo.
- PgAdmin3, PgAccess, PhpPgAdmin, Psql: Herramientas de administración.

El principal soporte de PostgreSQL lo provee la gran comunidad de usuarios que existe en el mundo, quienes aportan experiencias y soluciones obtenidas del trabajo con este gestor. Posee además soporte comercial, destacándose la aportada por su equipo de desarrollo.

De manera general las características de PostgreSQL que se ajustan a las principales demandas del proyecto en cuanto al gestor de base de datos, son:

- Debe poseer altos niveles de confiabilidad y escalabilidad.
- Ser estable durante el manejo de grandes volúmenes de información.
- Soporte de altos volúmenes de concurrencia.
- Soporte de integridad referencial.
- Soporte de procedimientos almacenados o funciones y vistas. [8].

La versión del gestor propuesto definida por el proyecto para ser utilizada es PostgreSQL 8.3.5, pues introduce mejoras considerables en cuanto al rendimiento.

1.5.5 Metodología

La selección de qué herramienta utilizar durante todo el ciclo de desarrollo del software constituye la principal disyuntiva a la que se enfrentan los desarrolladores de hoy en día. Una buena selección tanto de las herramientas a utilizar como de la metodología que se desea emplear favorece la calidad del software deseado, pues de ahí se deriva el papel que debe desempeñar cada miembro dentro del equipo de desarrollo y qué actividades tiene que cumplir cada uno de ellos, se definen qué artefactos deben ser creados y se pormenoriza cada detalle de la información del producto que se alcance como resultado de toda la actividad realizada.

Actualmente, existen diferentes metodologías y técnicas para el desarrollo de productos las que son analizadas y definidas en la arquitectura del proyecto, en Akademos quedó definido que serían utilizadas las metodologías ágiles que constituyen un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto, la utilización de estas metodologías minimiza riesgos desarrollando software en cortos lapsos de tiempo, se genera menos documentación y promueve las comunicaciones cara a cara con el cliente.

1.5.5.1 Metodologías Ágiles

“Las Metodologías Ágiles o “ligeras” constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos (*e-projects*) que las metodologías convencionales (ISO-9000, CMM, etc.) debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración.” [9].

Las metodologías ágiles se entienden como un conjunto de metodologías para desarrollo de software surgidas en la década de los 90.

Lo ágil se define, por los mismos agilistas, como la habilidad de responder de forma versátil al cambio para maximizar los beneficios.

Características de las metodologías ágiles

De manera general, las metodologías ágiles pueden explicarse a través de los siguientes principios fundamentales que son a su vez los propósitos del Manifiesto Ágil, documento que resume la filosofía ágil:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Dado que el proceso de desarrollo es creativo, no es posible pensar que las personas funcionen respondiendo a órdenes o procesos rígidos.
- Desarrollar software que funciona más que conseguir una buena documentación. Puesto que si el software no funciona la documentación no vale de nada. A nivel interno puede haber documentación, pero solo la necesaria y a nivel externo lo que el cliente requiera.
- La colaboración con el cliente más que la negociación de un contrato. Supone que la satisfacción del cliente con el producto será mayor, mientras exista una conversación y retroalimentación continua entre este y la empresa.
- Responder a los cambios más que seguir estrictamente un plan. Puesto que si un proyecto de software no es capaz de adaptarse a los cambios fracasará, especialmente en productos de gran envergadura. La estrategia de planificación se basa en: planes detallados para las próximas semanas, planes aproximados para los próximos meses y muy generales para plazos mayores. [10].

A nivel mundial existen varias metodologías para el desarrollo de software que encajan bajo el estandarte de ágil. Mientras todas ellas comparten muchas características, también hay algunas diferencias significativas. Dentro de las metodologías ágiles más conocidas podemos citar:

- XP (*Extreme Programming* – Kent Beck).

- SCRUM (Jeff Sutherland, Ken Schwaber).

- Crystal (Alistair Cockburn).

- DSDM (DSDM Consortium).

1.5.5.2 XP - Programación Extrema (*Extreme Programming*)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo, se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las

soluciones implementadas y coraje para enfrentar los cambios. Es adecuada para proyectos con requisitos imprecisos y muy cambiantes.

Como ventajas fundamentales de esta metodología se puede señalar que:

- Es un proceso ligero, ágil, de bajo riesgo, flexible, predecible y científico.
- Está orientada hacia quien produce y usa el software (retroalimentación continua cliente y desarrollador).
- Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema.
- Combina las que han demostrado ser las mejores prácticas para desarrollar software, y las lleva al extremo.

1.5.5.3 Scrum

SCRUM define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Sus principales características se pueden resumir en dos: el desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

Características fundamentales:

- Equipos auto dirigidos.
- Utiliza reglas para crear un entorno ágil de administración de proyectos.
- No prescribe prácticas específicas de ingeniería.
- Los requerimientos se capturan como ítems de la lista reserva del producto.
- El producto se construye en una serie de *sprints* de un mes de duración.

Scrum tiene como objetivo primordial elevar al máximo la productividad de un equipo. Reduce al máximo la burocracia y actividades no orientadas a producir software que funcione y produce resultados en períodos muy breves de tiempo (cada 30 días), por medio de iteraciones o *sprints*.

1.5.5.4 Metodología SXP

Scrum y XP (Programación eXtrema) pueden combinarse de forma fructífera pues sus características son ideales para agilizar el proceso de documentación del software. Scrum se enfoca en las prácticas de organización y gestión, mientras que XP se centra más en las prácticas de programación. Esa es la razón de que funcionen tan bien juntas: tratan áreas diferentes y se complementan entre ellas. En la nueva versión del sistema Akademos se utilizará la metodología SXP o MA-GMPR-UR2, que es la unión de Scrum y XP, por decisión de la Dirección de Informatización de la UCI y la dirección del proyecto. La figura 1 fue tomada del Release_0.2_SXP, documento donde se explica todo lo referente a la metodología SXP.

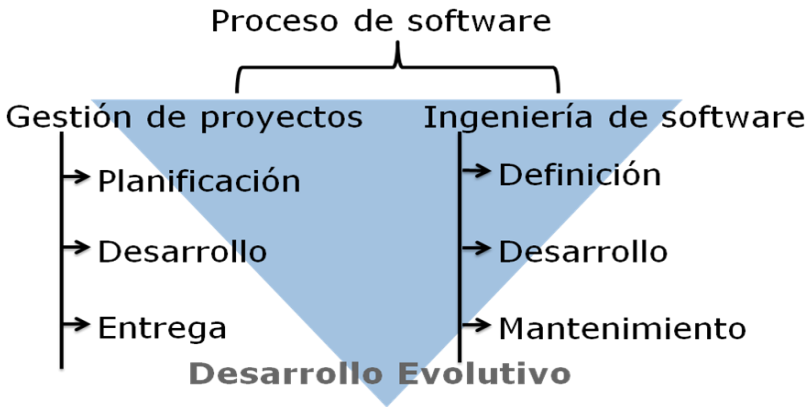

Figura 1 Esquema de la metodología SXP (MA-GMPR-UR2)

La aplicación de esta metodología creada en la Universidad por la facultad 10 presenta ventajas considerables en el desarrollo exitoso del proyecto, dentro de las más significativas se pueden citar:

- Con la utilización de SCRUM para la gestión, se logra una planificación y organización inigualable; mientras que XP respalda con sus prácticas todo el proceso de desarrollo, obteniéndose de esta forma un proceso de software completo.
- La generación de los artefactos necesarios e imprescindibles respalda la documentación de cada uno de los sistemas, de esta forma se logra que no queden sistemas sin ser analizados y documentados.
- Se logra que la organización de la documentación de cada uno de los sistemas sea eficiente, además, que cada uno de los miembros del equipo se muestre interesado y motivado para el

desarrollo. El trabajo es más ágil y se mantiene al cliente dentro del equipo de desarrollo lo que proporciona mejores resultados y una mayor satisfacción por parte de los interesados finales del producto. [11].

1.5.6 Herramientas CASE

Debido a las exigencias y el esfuerzo adicional que requiere la elaboración de los modelos y la gran cantidad de documentación que es generada, se hace imprescindible la utilización de las herramientas CASE.

La Ingeniería de Software Asistida por Computación (CASE, sus siglas en inglés), permite organizar y manejar la información de un proyecto informático, mejorar la comunicación entre los participantes y hace que un sistema, por muy complejo que parezca, se torne más flexible y más comprensible a sus desarrolladores.

Las herramientas CASE son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo del sistema de información.

En el diseño de la base de datos se utilizó, para el modelado de los artefactos a obtener: el Visual Paradigm.

1.5.6.1 Visual Paradigm

Visual Paradigm para UML (VP-UML), es una potente herramienta visual UML CASE. Esta herramienta además de soportar el modelado UML, provee el modelado de procesos de negocio *Business Process Modeling Notation* (BPMN) y el mapeo relacional de objetos para Java, .Net y PHP.

Para el modelado de base de datos utiliza dos tipos de diagramas fundamentales: el diagrama entidad-relación, que permite el modelado de base de datos relacional en el nivel físico y el diagrama de mapeo relacional de objetos que muestra el mapeo entre clases y las entidades, o sea muestra la correspondencia entre clases del mundo orientado a objetos y las entidades en el mundo relacional de base de datos.

Visual Paradigm es una herramienta CASE de fácil uso. Tiene la ventaja de ser multiplataforma. No es de libre distribución, pero la UCI adquirió la licencia de Visual Paradigm para UML, tiene la ventaja de que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos,

construcción, pruebas y despliegue. Este software de modelado permite la construcción, más rápida, de aplicaciones con mayor calidad y a menor costo. Permite graficar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

1.6 Conclusiones

En este capítulo ha sido abordado el estudio realizado de varios sistemas de gestión de información, incluyendo sistemas de gestión académica tanto en el ámbito nacional e internacional. De estos sistemas investigados se ha obtenido un conjunto de experiencias que han servido de apoyo a la investigación, teniendo en cuenta que este trabajo es para la gestión académica como parte de Akademos v_2.0, se realizó la descripción de las principales funcionalidades y los problemas que la primera versión del software presenta.

Como parte de la investigación realizada se describió la tecnología utilizada durante el trabajo. En cuanto al tipo de base de datos a utilizar se determinó que no era factible realizar un almacén de datos porque no se ajusta a las características específicas de los tipos de datos que se necesitan guardar pues sería muy complicado modelar un diseño que se correspondiera con los datos específicos de cada estudiante que deben almacenarse; por lo tanto la solución será realizada con una base de datos relacional.

Es necesario aclarar que no se realiza un estudio comparativo entre los distintos tipos de software utilizados en la actualidad en el mundo pues la investigación se ajusta a las políticas de la Dirección de Informatización de la UCI y a lo definido por la dirección del proyecto, es por esta razón que el gestor de base de datos es el PostgreSQL, y como herramienta de modelado se ha decidido utilizar Visual Paradigm. La metodología que se usará es la SXP que también ha sido definida por el equipo de desarrollo del proyecto.

Capítulo # 2

Análisis y descripción de la solución propuesta

2.1 Introducción

En el presente capítulo se describen actividades importantes para la construcción de la base de datos, entre ellas se encuentran: la selección y argumentación de los requisitos del sistema, la descripción de la arquitectura propuesta y la estrategia de integración de la Base de Datos Histórica con los módulos que componen el sistema.

Se construye, además, la propuesta de diseño de la base de datos, para lograr este objetivo se desarrolla el modelo lógico y físico de los datos y se describen las clases obtenidas en ambos modelos.

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

En la nueva versión, el sistema Akademos posee varios módulos dentro de los que se encuentran: Gestión de Carrera, Gestión de Personal, Registro y Control Docente y, Administración y Seguridad, para gestionar toda la información de pre-grado de un centro de estudios. La base de datos del sistema garantiza que la información generada en cada uno de ellos persista durante el transcurso de los estudiantes por la Universidad.

Al estudiante graduarse, la información que el sistema posee sobre el mismo será almacenada en la Base de Datos Histórica, cada módulo del sistema contará con un adaptador, que no es más que un segmento de código que es el encargado de pasar la información de una base de datos a otra de manera correcta.

Además, cada módulo deberá poseer un conjunto de interfaces en las cuales se muestre la información histórica almacenada en el momento y formato requerido.

La Base de Datos Histórica no sólo está integrada por los distintos módulos que componen el sistema, sino que brinda información a otras aplicaciones, que requieren datos personales y académicos

contenidos en la misma. El intercambio de información con la base de datos se realizará a través de clases de acceso a datos, generadas por los frameworks utilizados en el desarrollo, a su vez serán implementadas una serie de funciones que estarán encargadas de chequear la integridad y seguridad de los datos.

2.3 Descripción de la arquitectura propuesta y su fundamentación

El sistema Akademos en su versión 2.0 utiliza como lenguaje de programación PHP 5.2.8 y como marco de trabajo el Zend Framework (ZF) que es compatible con el gestor de base de datos definido. El ZF es uno de los frameworks más utilizados para PHP y utiliza el estilo arquitectónico MVC (Modelo-Vista-Controlador) como base de su funcionamiento, es fácilmente integrable a las aplicaciones debido a su composición y a que contiene diferentes clases de gran utilidad.

Será utilizado, además del ZF, el framework Doctrine que es un potente y completo sistema ORM[4] para PHP 5.2+ con un DBAL[5] incorporado. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente. Es fácilmente integrado a los principales frameworks de desarrollo utilizados actualmente dentro de los que se encuentra el Zend Framework. Permite exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos.

De manera general el software que es utilizado en el área de base de datos en el proyecto es el PostgreSQL como gestor de base de datos y como herramienta de modelado el Visual Paradigm.

A continuación se puntualizan de forma teórica algunos aspectos, de la Base de Datos Histórica, que permitirán que su confección sea entendible por todos siguiendo un orden lógico y así, facilitar el trabajo del diseñador de la misma.

Nomenclatura utilizada en la Base de Datos Histórica:

Tablas:

Tb_h<*prefijo*><*nombre1_nombre2*>

[4] ORM: Acrónimo de *Object-Relational Mapping* (Mapeo Objeto-Relacional).

[5] DBAL: Acrónimo de *Database Abstraction Layer* (Capa de Abstracción de la Base de Datos).

El nombre de las tablas será escrito en minúsculas y en singular preferentemente. Comenzará por "*Tb* _ + <*h*> haciendo referencia a las tablas históricas + *<prefijo>"* seguido de los nombres <*nombre1_nombre2*> que serán lo más descriptivos posible y separados por "_". Para poner el prefijo se seguirán las siguientes reglas:

- "*n*", para las tablas que representan nomencladores.
- "*d*", para las tablas que representan datos de entidades.
- "*r*", para las tablas que son formadas por relaciones entre otras tablas. En este caso, para nombrar la tabla se buscará un nombre descriptivo si se crea una nueva entidad, de lo contrario la tabla tomará los nombres de las tablas que relaciona preferentemente si no se encuentra alguno más significativo.

Ejemplos:

Tb_hdPersona (una tabla entidad, donde se guardan los datos de las personas)

Tb_hnNomenclador (una tabla para la generación de nomencladores)

Tb_hrPersona_Plantilla (tabla formada por la relación entre Tb_hdPersona y Tb_hdPlantilla)

Campos de las Tablas:

Los nombres de los campos serán en minúsculas, cortos y descriptivos, si el nombre fuera compuesto estos serían separados por "_". Los campos identificadores (Id) comenzarán con "*Id_*" seguido del nombre, el nombre coincidirá preferentemente con el nombre de la tabla de la que el campo es llave.

Ejemplos:

Id_Persona (campo llave la tabla persona)

Id_Asignatura (campo llave de la tabla Asignatura)

Procedimientos Almacenados o Funciones:

<pa|fn>_<<entidad>_<operación>|<funcionalidad>>

Los nombres de los procedimientos almacenados y funciones serán en minúscula, comenzarán con "*pa" o "fn"* respectivamente seguido de "_". Si el procedimiento o la función ejecuta una operación sobre una entidad, irá seguido de <*entidad*> + _ + <*operación*>, de lo contrario irá seguido de un nombre, <*funcionalidad*>, que describirá su funcionalidad, si el nombre fuera compuesto estos serían

separados por "_". Para nombrar los procedimientos y funciones de operaciones sobre entidades, la <*operación*> sería:

- "*ins*" para los procedimientos de insertar.
- "*act*" para los procedimientos de actualizar.
- "*sel*" para los procedimientos de seleccionar.

Ejemplo:

pa_persona_ins (procedimiento para insertar una persona)

Vistas:

vs_<*nombre*>

Los nombres de las vistas en minúsculas y descriptivos. Comienzan con *"vs_"* seguido del nombre, si el nombre fuera compuesto estos serían separados por "_".

Ejemplo:

vs_personas (vista que devuelve los datos de todas las personas).

2.4 Reglas del Negocio

1- Los datos de los egresados pasarán a la Base de Datos Histórica luego de tres años de haberse efectuado la graduación.

Se establece que los datos de los egresados pueden considerarse como activos en los cursos en los que esté cumpliendo el servicio social.

2- Los datos de los estudiantes que provoquen baja de la Universidad no podrán pasar a la Base de Datos Histórica hasta que no haya culminado su graduación.

Se espera a que finalice la graduación del año en que el estudiante entró en la Universidad para pasarlo al histórico con su expediente cerrado.

3- Una vez que los datos pasen a formar parte de la Base de Datos Histórica no volverán a la base de datos central.

Se establece que los datos no pueden volver a la base de datos central una vez que sean transferidos al histórico pues estos datos se consideran pasivos después de haberse efectuado la graduación.

4- Para diferenciar los egresados de las facultades regionales se utilizará en la tabla Tb_hdEstructura la siguiente nomenclatura: B – Artemisa, A – Ciego de Ávila, G – Granma.

Se establecen estas iniciales para una mejor comprensión de la notación que debe ser empleada para definir la estructura de los estudiantes de las facultades regionales.

2.5 Selección y argumentación de los requisitos funcionales y no funcionales del sistema propuesto

Los requisitos son condiciones o capacidades que tienen que ser alcanzadas o poseídas por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Los requisitos se clasifican en funcionales y no funcionales. Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir y los no funcionales son propiedades o cualidades que el producto debe tener.

2.5.1 Requisitos Funcionales

La investigación responde a un número significativo de requisitos funcionales de los distintos módulos que componen el Sistema de Gestión Académica de Akademos v_2.0, de todos los requisitos funcionales de la base de datos central del sistema se han seleccionado para formar parte de la Base de Datos Histórica sólo aquellos que, por su relevancia, es necesario que perduren de 25 a 30 años dentro de los archivos centrales de la Universidad. A continuación aparece una descripción de los principales requerimientos propuestos:

Requisitos Funcionales del Módulo Registro y Control Docente

RF- 6 Consultar grupo docente.

6.1- Buscar grupo docente por criterios.

6.2- Mostrar listado de grupos docentes ordenados por un criterio.

6.3- Ver datos del grupo docente seleccionado.

RF- 13 Ver Registro de Asistencia.

RF- 16 Ver Registro de Evaluación.

RF- 24 Ver Expediente del Estudiante.

RF- 26 Ver Resumen Histórico.

RF- 36 Ver Notas Históricas Básicas.

RF- 37 Ver Notas Históricas Optativas.

RF- 38 Ver Notas Históricas Finales.

RF- 39 Ver Hoja de Resultados Académicos.

RF- 40 Ver Certificación de Notas.

Requisitos Funcionales del Módulo Gestión de Personal

RF- 2 Visualizar estado.

RF- 15 Buscar persona.

15.1 Mostrar el listado de personas.

15.2 Mostrar detalles de la persona.

RF- 37 Ver lista de bajas por período.

RF- 38 Ver listado de traslados por período.

Requisitos Funcionales del Módulo Gestión de Carreras

RF- 2 Visualizar Carrera.

RF- 14 Visualizar Tipos de Asignaturas.

RF- 22 Visualizar Plan de Estudio.

RF- 27 Visualizar Bonificación.

RF- 30 Visualizar Asignatura.

RF- 34 Visualizar Actividad Extracurricular.

Otros Requisitos en cuanto a la Seguridad

RF- 1 Autenticar.

RF- 2 Comprobar permisos.

RF- 3 Registrar incidencia.

2.5.2 Requisitos no funcionales

Los requerimientos no funcionales forman una parte significativa de la especificación de requisitos de un proyecto pues, en muchos casos, son fundamentales en el éxito del producto. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable, casi siempre son estas propiedades las que pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. A continuación se describen los principales requerimientos relacionados con la base de datos.

Seguridad

Seguridad de la Base de Datos

- La base de datos deberá estar fraccionada en esquemas que permitan un mejor uso de la información y la división de forma lógica de las funcionalidades del sistema, trayendo consigo además la protección de la información al ocurrir un incidente sobre una parte de la base de datos.
- El Sistema Gestor de Base de Datos escogido debe presentar facilidades de administración de roles y usuarios restringiendo el acceso a los datos.

Registro sistemático de incidencias

- El sistema debe ser capaz de registrar el accionar del usuario sobre el mismo, así como permitir auditorías y exámenes de las trazas tanto en tiempo real como en históricos.
- Se precisa un monitor de incidencia para la visualización y tratamiento de las mismas.

Alta protección de los datos

- Al estar trabajando con información sensible, se hace necesario una alta protección de los datos a nivel de aplicación y de tráfico por la red, para tal fin se ha definido además la seguridad en varios

niveles dentro de la aplicación (Nivel de Interfaz, Nivel de Acceso a Datos y Nivel de Base de Datos), así como el uso del protocolo seguro HTTPS para el tráfico de la información por la red.

Políticas de seguridad por usuarios y roles

- El sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.

Disponibilidad

- La base de datos deberá estar disponible las 24 horas del día.

Rendimiento

Escalabilidad

- El sistema debe ser capaz de mantener un rendimiento y una estabilidad adecuada al gestionar amplios volúmenes de datos, así como de permitir la incorporación de nuevas funcionalidades adaptándose de manera natural a los procesos de gestión que se requieran.

Hardware

Para el desarrollo

- PC con las siguientes características: Intel Pentium 4 o superior, CPU 3GHZ o superior, 512 MB RAM o superior, 160 GB HDD o superior.

Para explotación del cliente

- PC con las siguientes características: Pentium 3 o superior, CPU 133 MHZ o superior, 128 RAM mínimo (512 RAM recomendada o superior).

Para explotación del servidor

- PC con las siguientes características: CPU: Dual Core 2.0 GHZ o superior, RAM: 4 GB (Recomendado 6 GB), 250 GB HDD.

Software

Para el cliente

- Sistema operativo con interfaz gráfica y conexión a red.

- Navegador Web (Mozilla FireFox Recomendado).

Para el Servidor

- Sistema operativo Linux del servidor: Ubuntu Server 7.10 ó superior.
- Framework: Zend Framework 1.7.6.
- Servidor Web: Apache 2.2.9.
- Gestor de Base de Datos: PostgreSQL 8.3.5.
- Software controlador de versiones: Subversion v_1.4.4 (r25188).

Soporte

Manual de Usuario

- El sistema deberá presentar un manual de usuario, permitiendo con ello un correcto uso de sus funcionalidades y brindarle al usuario una mayor experiencia del trabajo con el mismo.

Usabilidad

Facilidad de uso por parte de los usuarios

- El sistema debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Debe, además, ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencia una rápida adaptación al mismo.

Emplear perfiles de usuario

- Diferenciar las interfaces y opciones para los usuarios que accedan al sistema con diferentes roles (secretaria general, secretaria, administrador, estudiantes, etc.).

Portabilidad

- El sistema está basado en un diseño comercial, entiéndase en un producto propiamente dicho y no en una solución abierta. Al ser conceptualizado como producto deberá poderse instalar, así como actualizar y desinstalar, sobre Linux o Windows respectivamente.
- La actualización a la que se hace referencia vela por un conjunto de funcionalidades que se le pueden incorporar a los módulos ya instalados del sistema así como otros que se le quieran

agregar. Todo este diseño modular del sistema determina un bajo acoplamiento entre las distintas entidades del mismo.

Restricciones en el diseño y la implementación

- Lenguaje de programación será: PHP 5.2.8.
- El framework base de desarrollo que se utilizará es: Zend Framework 1.7.6.
- Como IDE se empleará NetBeans 6.5.
- Como servidor Web se explotará Apache 2.2.9.
- El SGDB deberá ser PostgreSQL 8.3.5.
- El modelado UML se realizará con Visual Paradigm 3.0.
- El sistema operativo a utilizar en el entorno de desarrollo deberá ser: Windows XP SP 2 o Ubuntu 7.10 (ó superior).
- El repositorio principal, el entorno de prueba y el servidor de base de datos estarán montados sobre Ubuntu Server 7.10 (ó superior).

Apariencia o interfaz externa

Interfaz Web

- La interfaz deberá ser sencilla con colores suaves a la vista y sin cúmulo de imágenes u objetos que distraigan al cliente del objetivo de su empleo.

Interfaz Interna

- La interfaz interna estará determinada por los desarrolladores, construyendo así una vista escalable de las clases o agrupaciones de clases que permitirán un mejor encapsulamiento de las funcionalidades y una mayor abstracción modular del sistema.

2.6 Diseño de la Base de Datos Histórica

Patrones utilizados

Durante la realización del diagrama de clases persistentes y el diagrama entidad – relación de la Base de Datos Histórica fueron utilizados varios patrones de diseño dentro de los que se encuentran los patrones de Brown, a continuación se describe la utilización de los mismos.

Nombre: **Representación de objetos como tablas**

Este patrón es utilizado cuando se quiere representar un objeto en un esquema de base de datos relacional. Como solución propone definir una tabla para cada clase de objetos persistente. Los atributos de la clase que son tipos primitivos serán las columnas de la tabla.

Nombre: **Intermediario de base de datos**

Este patrón es utilizado cuando es necesario conocer quién es el responsable de la persistencia de los datos. Propone como solución construir una clase que es responsable de la materialización y desmaterialización de cada uno de los objetos persistentes.

Nombre: **Representación de relaciones como tablas**

Al diseñar una base de datos se debe tener en cuenta cómo representar una relación en un esquema de base de datos relacional, este patrón de diseño propone soluciones para los distintos tipos de relaciones que posee este modelo de datos.

Para las relaciones uno a uno o uno a muchos:

- Colocar una clave ajena en la tabla de cardinalidad uno, para representar la relación entre los objetos.
- O, crear una tabla asociativa para registrar los identificadores de cada uno de los objetos de la relación.

Para las relaciones muchos a muchos:

- Crear una tabla asociativa para registrar los identificadores de cada uno de los objetos de la relación.

Descripción de la propuesta de solución:

Para la realización de este trabajo fue creado un nuevo esquema dentro de la base de datos central con la información histórica, para evitar las inconsistencias en los datos que podrían ocurrir cuando se efectuara la migración de una base de datos a otra. Una ventaja de la propuesta es con respecto a las políticas de seguridad, pues al ser parte de la base de datos central no hay que definir políticas apartes sino que asume las que le sean aplicadas a la misma. No obstante, al ser un esquema, pueden realizarse copias de seguridad del mismo sin necesidad de realizárselas a la base de datos completa.

Para obtener la propuesta de solución se partió del diagrama de clases persistentes de la base de datos central del sistema Akademos v_2.0 y fueron revisados los conceptos y atributos definidos para cada uno de estos. Se realizó la selección de aquellos conceptos y atributos que fueron identificados para pasar al modelo histórico. Durante este proceso se unieron algunos conceptos que para la base de datos central poseen gran importancia, pero no así para la Base de Datos Histórica.

Partiendo del modelo de clases persistentes de la Base de Datos Histórica, se realizó el diagrama entidad-relación de dicha base de datos. En el capítulo 3 del presente trabajo de tesis se aborda con más amplitud el proceso de normalización realizado para la obtención de este diseño.

Como parte de la solución se realizaron algunos *Tablepaces* que fueron aplicados a las tablas del diseño. La ventaja fundamental que se le atribuye al uso de los mismos radica en la mejora considerable que proporcionan en cuanto a rendimiento de la base de datos. El uso del *Tablespace* permite darle un espacio lógico a las tablas de la base de datos que son mayormente consultadas, las tablas que son de sólo lectura, las vistas y procedimientos almacenados que son comúnmente utilizados, los índices, etc.; cada uno de estos elementos son ubicados en *Tablespaces* diferentes que a su vez tienen un espacio físico diferente para almacenar los datos, esto permite que se puedan realizar varias querys al mismo tiempo y que aumente la velocidad de respuesta del servidor pues las tablas están distribuidas en varios espacios físicos diferentes.

Los *Tablespaces* en PostgreSQL permiten a los administradores de la base de datos definir lugares donde pueden ser almacenados los objetos de la base de datos en carpetas. Los *Tablespaces* especifican solamente los lugares de almacenamiento de la base de datos, no la estructura lógica o el esquema de la base de datos. Por ejemplo, diferentes objetos en el mismo esquema, pueden tener asociados diferentes *Tablespaces*.

Existen dos ventajas fundamentales que proporciona el uso del *Tablespace*: en primer lugar si a la partición o volumen en la cual el clúster fue inicializado, se le agota el espacio y no puede aumentársele, se puede crear un *Tablespace* en otra partición y usarlo hasta que el sistema pueda reconfigurarse. Un uso común de los *Tablespaces* es optimizar la ejecución. Por ejemplo, un índice muy usado puede ser situado en un disco rápido SCSI. Por otro lado, una tabla de base de datos que contenga datos archivados a los que se accede con muy poca frecuencia puede ser almacenada en un disco IDE más barato pero más lento.

2.6.1 Diagrama de clases persistentes

El diagrama de clases persistentes es obtenido a partir del diagrama de clases del diseño. La persistencia es la propiedad de los objetos de trascender su estado en el tiempo y el espacio. Estos datos existen durante las ejecuciones de un programa, entre distintas versiones del programa o sobreviven a la eliminación del mismo. Debido al tamaño del diagrama este se ha dividido en 2 secciones para su mejor visualización.

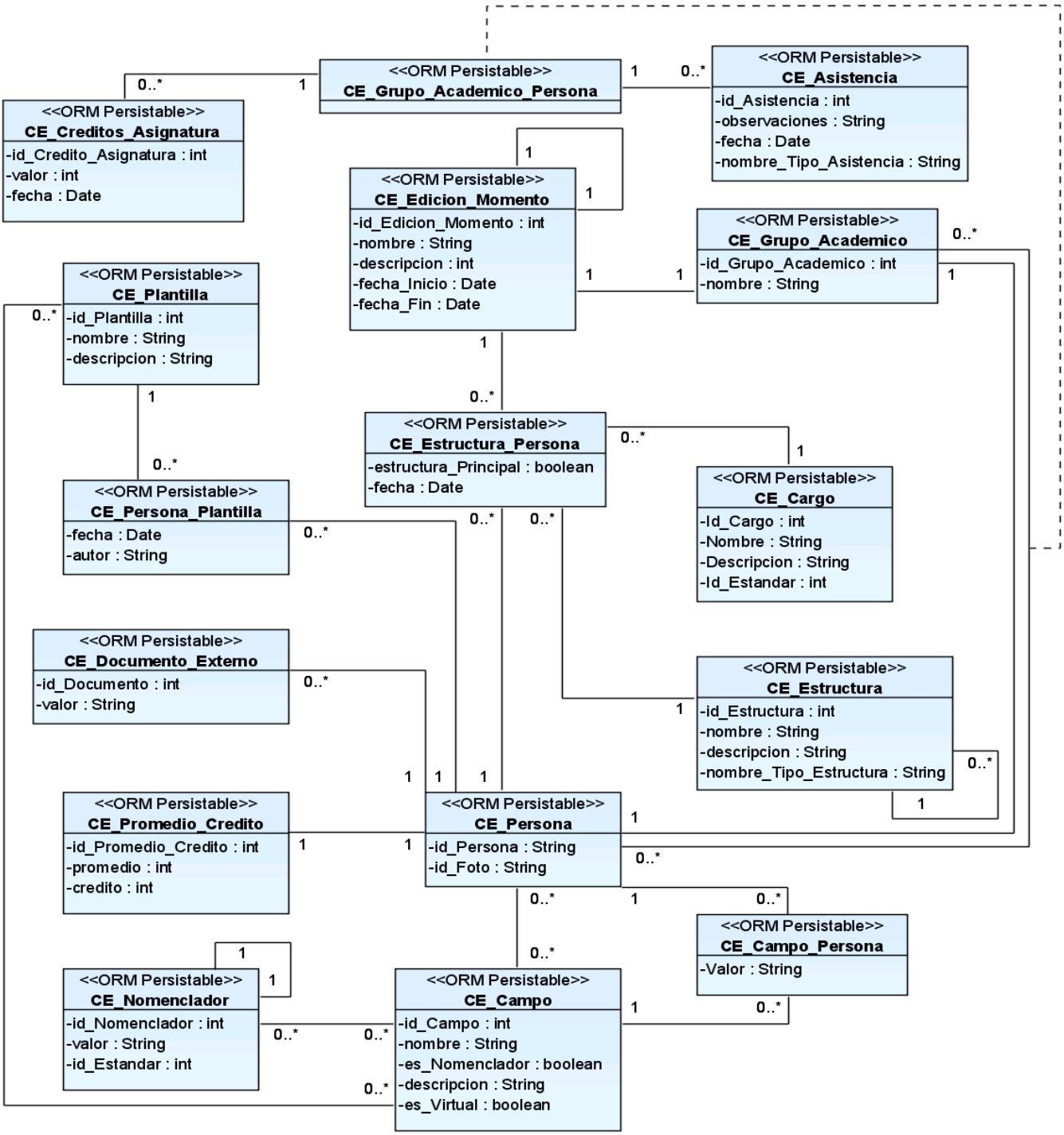

Figura 2 Diagrama de Clases Persistentes

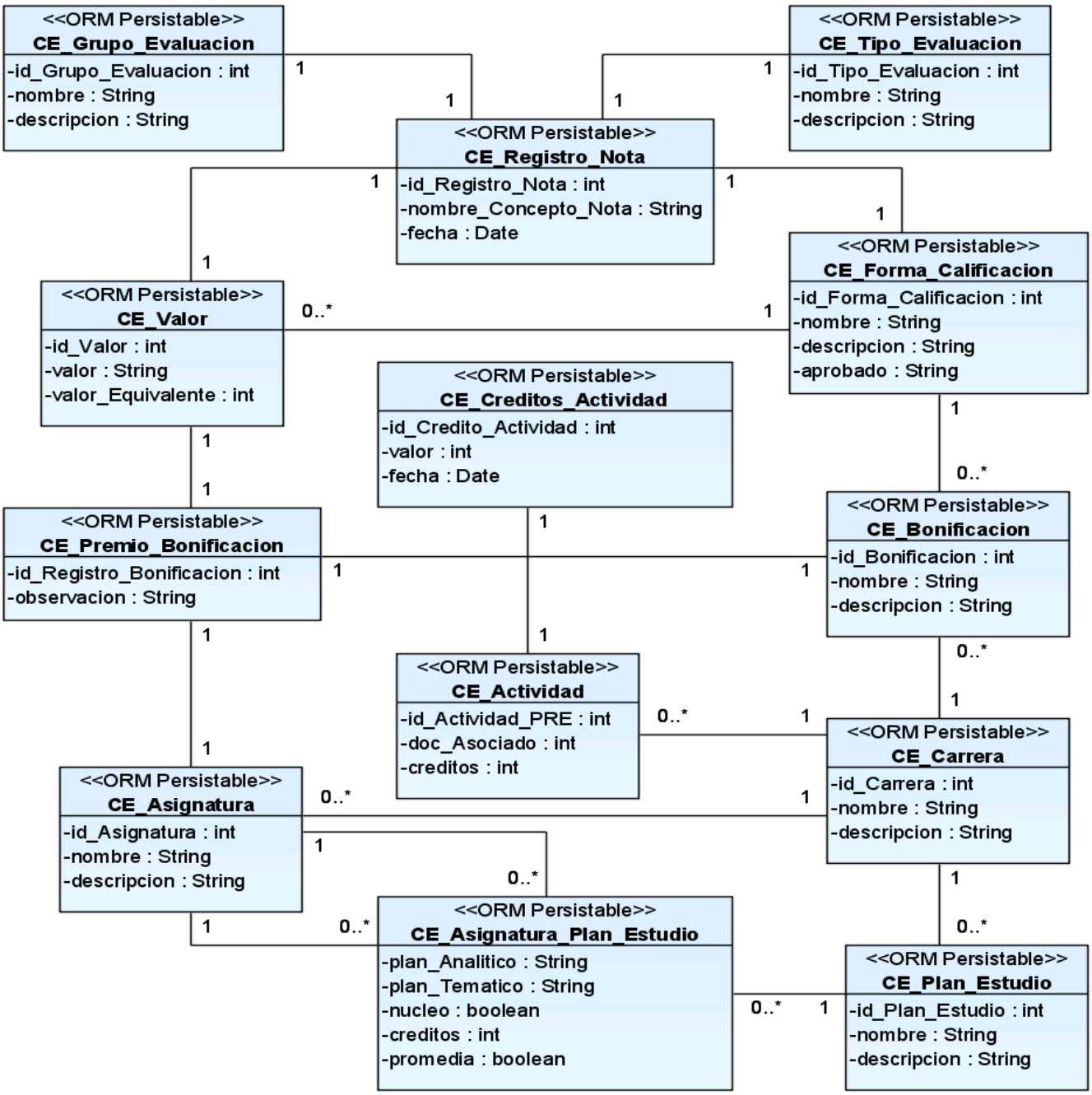

Figura 3 Diagrama de Clases Persistentes (Continuación)

2.6.2 Descripción de las clases persistentes

A continuación se representa la descripción de las clases persistentes modeladas anteriormente, de cada una de ellas se especifica el nombre, el tipo de clase, atributos y el tipo de cada uno de estos.

Tabla 2.1 Descripción de la clase: CE_Promedio_Credito

Nombre: CE_Promedio_Credito	
Tipo de clase Entidad	
Atributo	**Tipo**
id_Promedio_Credito	Int
Promedio	Int
Credito	Int

Tabla 2.2 Descripción de la clase: CE_Nomenclador

Nombre: CE_Nomenclador	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Nomenclador	Int
Valor	String
Id_Estandar	Int

Tabla 2.3 Descripción de la clase: CE_Campo

Nombre: CE_Campo	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Campo	Int
Nombre	String
Es_Nomenclador	Boolean
Descripcion	String
Es_Virtual	Boolean

Tabla 2.4 Descripción de la clase: CE_Registro_Nota

Nombre: CE_Registro_Nota	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Registro_Nota	Int
Nombre_Concepto_Nota	String
Fecha	String

Tabla 2.5 Descripción de la clase: CE_Estructura

Nombre: CE_Estructura	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Estructura	Int
Nombre	String
Descripcion	String
Nombre_Tipo_Estructura	String

Tabla 2.6 Descripción de la clase: CE_Cargo

Nombre: CE_Cargo	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Cargo	Int
Nombre	String
Descripcion	String
Id_Estandar	Int

Tabla 2.7 Descripción de la clase: CE_Estructura_Persona

Nombre: CE_Estructura_Persona	
Tipo de clase Entidad	
Atributo	**Tipo**
Estructura_Principal	Boolean
Fecha	Date

Tabla 2.8 Descripción de la clase: CE_Grupo_Academico

Nombre: CE_Grupo_Academico	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Grupo_Academico	Int
Nombre	String

Tabla 2.9 Descripción de la clase: CE_Documento_Externo

Nombre: CE_Documento_Externo	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Documento	Int
Valor	String

Tabla 2.10 Descripción de la clase: CE_Edicion_Momento

Nombre: CE_Edicion_Momento	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Edicion_Momento	Int
Nombre	String
Descripcion	String
Fecha_Inicio	Date
Fecha_Fin	Date

Tabla 2.11 Descripción de la clase: CE_Asignatura_Plan_Estudio

Nombre: CE_Asignatura_Plan_Estudio	
Tipo de clase Entidad	
Atributo	**Tipo**
Plan_Analitico	String
Plan_Tematico	String
Nucleo	Boolean
Creditos	Int
Promedia	Boolean

Tabla 2.12 Descripción de la clase: CE_Plan_Estudio

Nombre: CE_Plan_Estudio	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Plan_Estudio	Int
Nombre	String
Descripcion	String

Tabla 2.13 Descripción de la clase: CE_Asignatura

Nombre: CE_Asignatura	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Asignatura	Int
Nombre	String
Descripcion	String

Tabla 2.14 Descripción de la clase: CE_Carrera

Nombre: CE_Carrera	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Carrera	Int
Nombre	String
Descripcion	String

Tabla 2.15 Descripción de la clase: CE_Actividad

Nombre: CE_Actividad	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Actividad	Int
Doc_Asociado	Int
Creditos	Int

Tabla 2.16 Descripción de la clase: CE_Bonificacion

Nombre: CE_Bonificacion	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Bonificacion	Int
Nombre	String
Descripcion	String

Tabla 2.17 Descripción de la clase: CE_Premio_Bonificacion

Nombre: CE_Premio_Bonificacion	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Registro_Bonificacion	Int
Observacion	String

Tabla 2.18 Descripción de la clase: CE_Campo_Persona

Nombre: CE_Campo_Persona	
Tipo de clase Entidad	
Atributo	**Tipo**
Valor	String

Tabla 2.19 Descripción de la clase: CE_Valor

Nombre: CE_Valor	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Valor	Int
Valor	String
Valor_Equivalente	String

Tabla 2.20 Descripción de la clase: CE_Plantilla

Nombre: CE_Plantilla	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Plantilla	Int
Nombre	String
Descripcion	String

Tabla 2.21 Descripción de la clase: CE_Forma_Calificacion

Nombre: CE_Forma_Calificacion	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Forma_Calificacion	Int
Nombre	String
Descripcion	String
Aprobado	String

Tabla 2.22 Descripción de la clase: CE_Grupo_Evaluacion

Nombre: CE_Grupo_Evaluacion	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Grupo_Evaluacion	Int
Nombre	String
Descripcion	String

Tabla 2.23 Descripción de la clase: CE_Persona_Plantilla

Nombre: CE_Persona_Plantilla	
Tipo de clase Entidad	
Atributo	**Tipo**
Fecha	String
Autor	String

Tabla 2.24 Descripción de la clase: CE_Asistencia

Nombre: CE_Asistencia	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Asistencia	Int
Observaciones	String
Fecha	String
Nombre_Tipo_Asistencia	String

Tabla 2.25 Descripción de la clase: CE_Creditos_Asignatura

Nombre: CE_Creditos_Asignatura	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Creditos_Asignatura	Int
Valor	Int
Fecha	String

Tabla 2.26 Descripción de la clase: CE_Tipo_Evaluacion

Nombre: CE_Tipo_Evaluacion	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Tipo_Evaluacion	Int
Nombre	String
Descripcion	String

Tabla 2.27 Descripción de la clase: CE_Creditos_Actividad

Nombre: CE_Creditos_Actividad	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Creditos_Actividad	Int
Valor	Int
Fecha	String

Tabla 2.28 Descripción de la clase: CE_Persona

Nombre: CE_Persona	
Tipo de clase Entidad	
Atributo	**Tipo**
Id_Persona	String
Id_Foto	String

2.6.3 Diagrama Entidad – Relación de la Base de Datos Histórica

El diagrama entidad-relación representa el estado final de las tablas de la base de datos. Los diagramas por módulos se encuentran más detallados en los anexos. [Ver anexo 1].

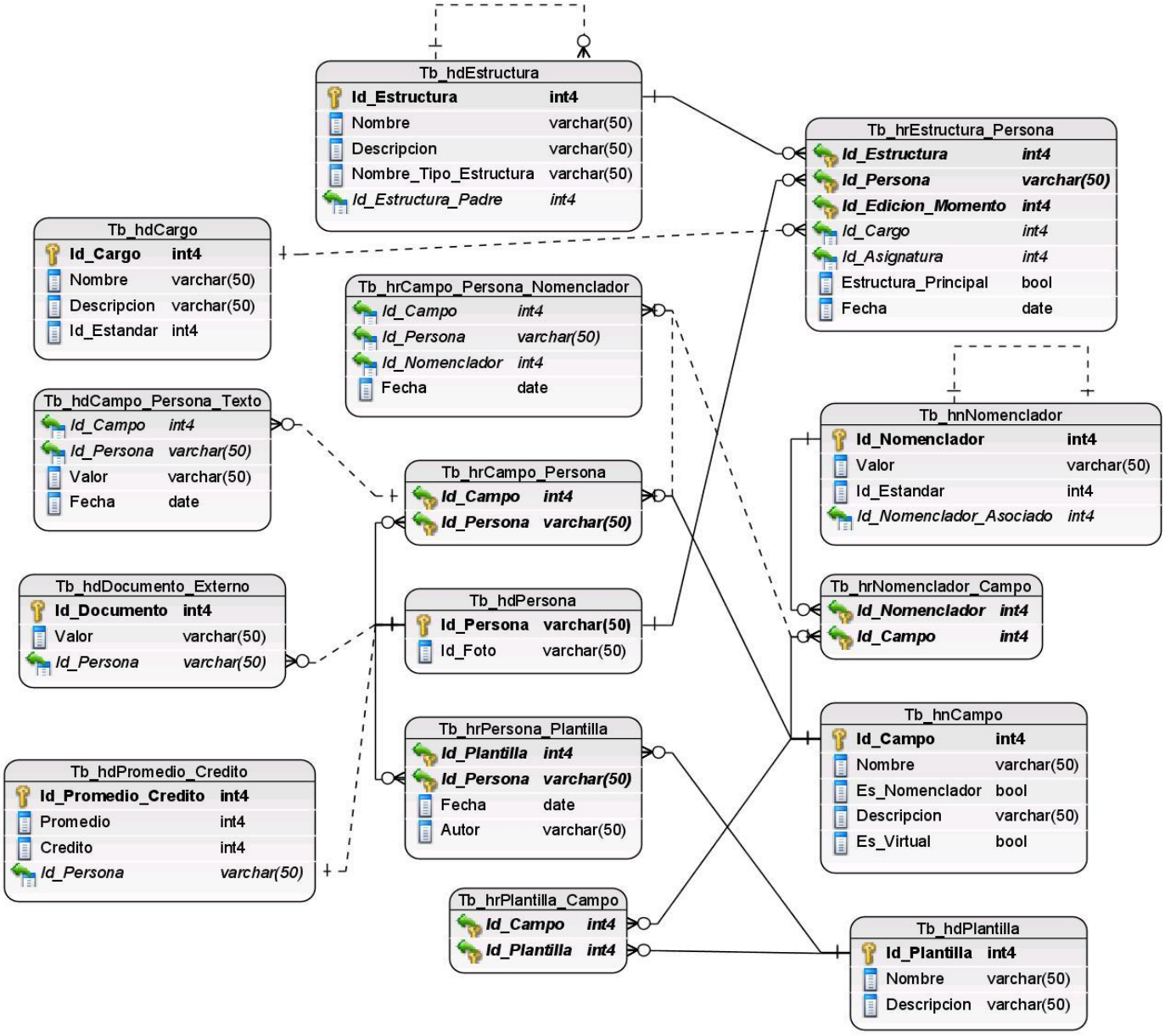

Figura 4 Diagrama Entidad-Relación

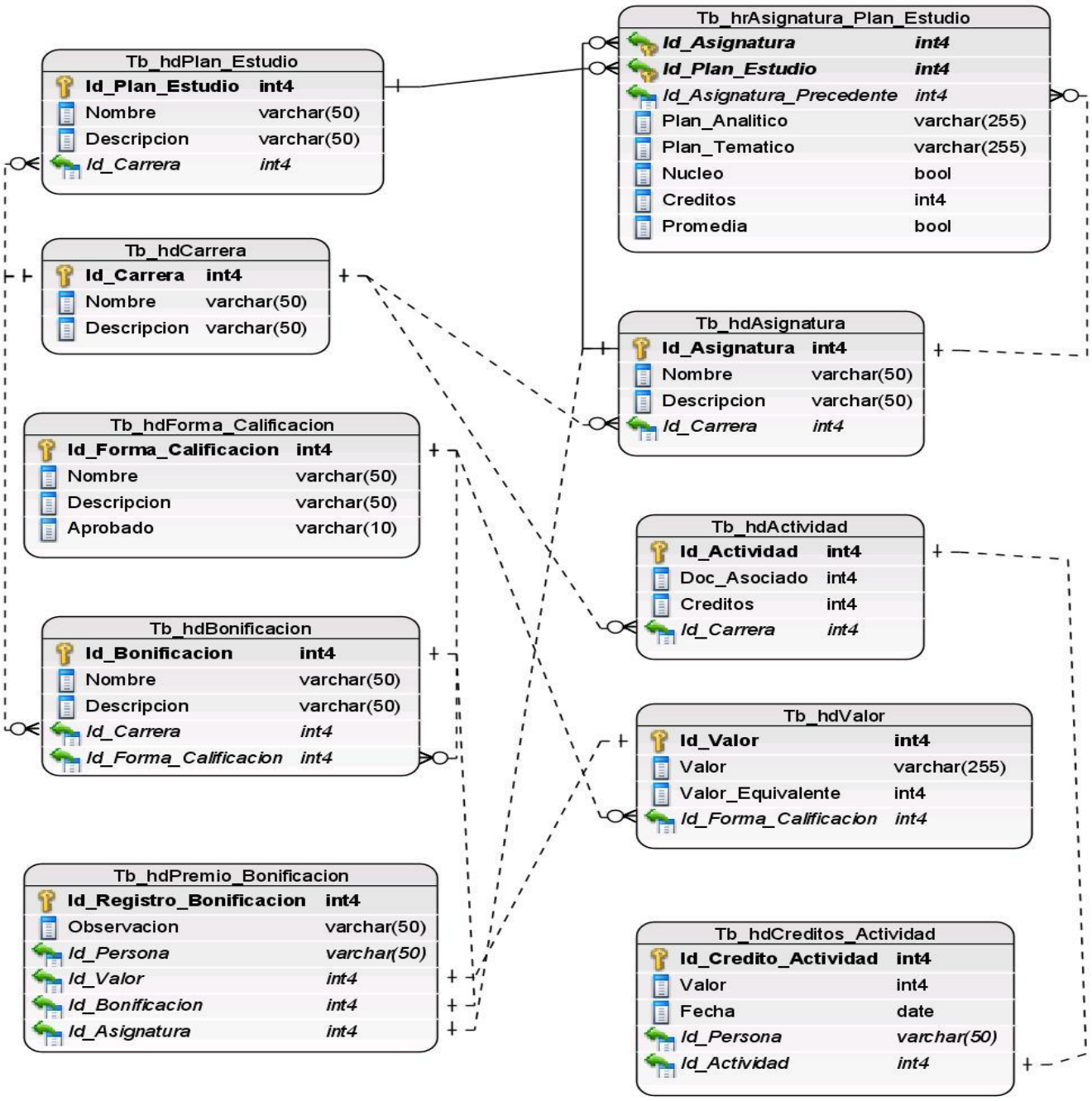

Figura 5 Diagrama Entidad-Relación (Continuación)

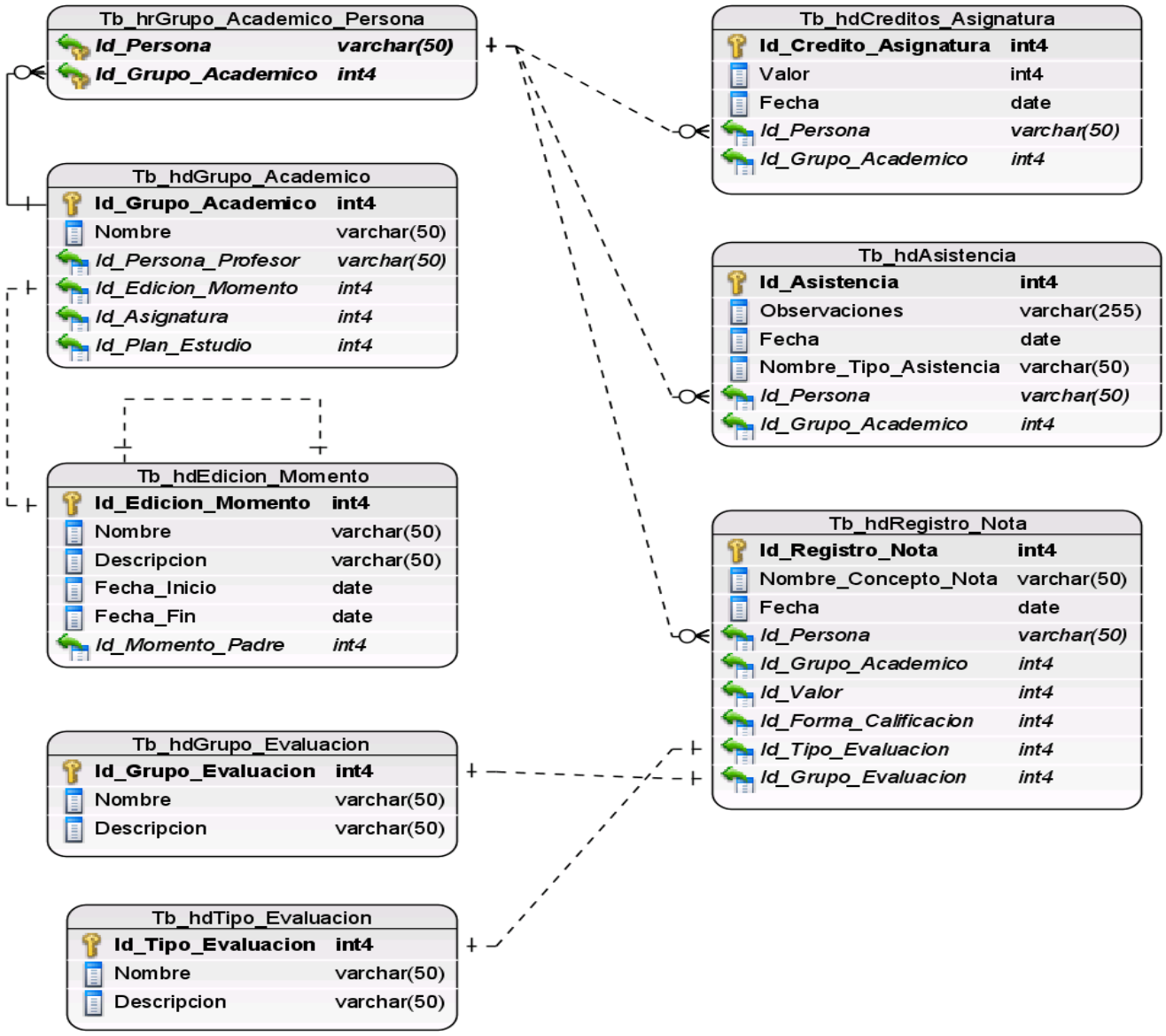

Figura 6 Diagrama Entidad-Relación (Continuación)

2.6.4 Descripción de las tablas del diagrama entidad – relación

Tabla 2.29 Descripción de la tabla: Tb_hdGrupo_Academico

Nombre: Tb_hdGrupo_Academico		
Descripción: Almacena los datos de los grupos académicos creados en el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Grupo_Academico	Entero	Identificador del grupo académico.
Nombre	Cadena	Nombre del grupo académico.
Id_Persona_Profesor	Cadena	Identificador del profesor.
Id_Edicion_Momento	Entero	Identificador de la edición del momento.
Id_Asignatura	Entero	Identificador de la asignatura.
Id_Plan_Estudio	Entero	Identificador del plan de estudio.

Tabla 2.30 Descripción de la tabla: Tb_hdAsistencia

Nombre: Tb_hdAsistencia		
Descripción: Almacena los datos de la asistencia de los estudiantes.		
Atributo	**Tipo**	**Descripción**
Id_Asistencia	Entero	Identificador de la asistencia.
Observaciones	Cadena	Descripción sobre la asistencia.
Fecha	Cadena	Fecha en que se registra la asistencia.
Nombre_Tipo_Asistencia	Cadena	Nombre del tipo de asistencia.
Id_Persona	Cadena	Identificador de la persona.
Id_Grupo_Academico	Entero	Identificador del grupo académico.

Tabla 2.31 Descripción de la tabla: Tb_hdRegistro_Nota

Nombre: Tb_hdRegistro_Nota		
Descripción: Almacena las notas de los estudiantes.		
Atributo	**Tipo**	**Descripción**
Id_Registro_Nota	Entero	Identificador del registro de nota.
Nombre_Concepto_Nota	Cadena	Nombre del concepto por el cual se emite la nota
Fecha	Cadena	Fecha en que se registra la nota.
Id_Persona	Cadena	Identificador de la persona.
Id_Grupo_Academico	Entero	Identificador del grupo académico.
Id_Valor	Entero	Identificador del valor.
Id_Forma_Calificacion	Entero	Identificador de la forma de calificación.
Id_Tipo_Evaluacion	Entero	Identificador del tipo de evaluación.
Id_Grupo_Evaluacion	Entero	Identificador del grupo de evaluación.

Tabla 2.32 Descripción de la tabla: Tb_hdEdicion_Momento

Nombre: Tb_hdEdicion_Momento		
Descripción: Almacena las ediciones de momento definidas en el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Edicion_Momento	Entero	Identificador de la edición de momento.
Nombre	Cadena	Nombre de la edición de momento.
Descripcion	Cadena	Descripción de la edición de momento.
Fecha_Inicio	Cadena	Fecha inicial de la edición de momento.
Fecha_Fin	Cadena	Fecha final de la edición de momento.
Id_Momento_Padre	Entero	Identificador de la edición de momento a la cual está asociada.

Tabla 2.33 Descripción de la tabla: Tb_hdBonificacion

Nombre: Tb_hdBonificacion		
Descripción: Almacena las bonificaciones y premios asignados a estudiantes.		
Atributo	**Tipo**	**Descripción**
Id_Bonificacion	Entero	Identificador de la bonificación.
Nombre	Cadena	Nombre de la bonificación.
Descripcion	Cadena	Descripción de la bonificación.
Id_Carrera	Entero	Identificador de la carrera.
Id_Forma_Calificacion	Entero	Identificador de la forma de calificación.

Tabla 2.34 Descripción de la tabla: Tb_hdPlan_Estudio

Nombre: Tb_hdPlan_Estudio		
Descripción: Almacena los datos correspondientes a los planes de estudios creados en el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Plan_Estudio	Entero	Identificador del plan estudio.
Nombre	Cadena	Nombre del plan de estudio.
Descripcion	Cadena	Descripción del plan de estudio.
Id_Carrera	Entero	Identificador de la carrera.

Tabla 2.35 Descripción de la tabla: Tb_hdPersona

Nombre: Tb_hdPersona		
Descripción: Almacena las personas que posee el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Persona	Cadena	Identificador de la persona.
Id_Foto	Cadena	Identificador de la foto de la persona.

Tabla 2.36 Descripción de la tabla: Tb_hdAsignatura

Nombre: Tb_hdAsignatura		
Descripción: Almacena los datos generales de las asignaturas.		
Atributo	**Tipo**	**Descripción**
Id_Asignatura	Entero	Identificador de la asignatura.
Nombre	Cadena	Nombre de la asignatura.
Descripcion	Cadena	Descripción de la asignatura.
Id_Carrera	Entero	Identificador de la carrera.

Tabla 2.37 Descripción de la tabla: Tb_hrAsignatura_Plan_Estudio

Nombre: Tb_hrAsignatura_Plan_Estudio		
Descripción: Almacena los datos de las asignaturas específicas de un plan de estudio.		
Atributo	**Tipo**	**Descripción**
Id_Asignatura	Entero	Identificador de la asignatura.
Id_Plan_Estudio	Entero	Identificador del plan estudio.
Plan_Analitico	Cadena	Documento donde se describen los datos generales de una asignatura como distribución de horas clases, evaluaciones, etc., de un plan de estudio.
Plan_Tematico	Cadena	Documento donde se describen varios temas relacionados con la asignatura como sus objetivos, indicaciones metodológicas, etc., en un determinado plan de estudio.
Nucleo	Boolean	Indica si la asignatura es del núcleo.
Creditos	Entero	Cantidad de créditos que aporta la asignatura.
Promedia	Boolean	Indica si la asignatura promedia.
Id_Asignatura_Precedente	Entero	Identificador de la asignatura que la precede en el plan de estudio.

Tabla 2.38 Descripción de la tabla: Tb_hrNomenclador_Campo

Nombre: Tb_hrNomenclador_Campo		
Descripción: Almacena los datos de las asociaciones entre los nomencladores y los campos.		
Atributo	**Tipo**	**Descripción**
Id_Nomenclador	Entero	Identificador del nomenclador.
Id_Campo	Entero	Identificador del campo.

Tabla 2.39 Descripción de la tabla: Tb_hdGrupo_Evaluacion

Nombre: Tb_hdGrupo_Evaluacion		
Descripción: Almacena los datos de los grupos de evaluaciones pertenecientes a los distintos Planes de Estudios.		
Atributo	**Tipo**	**Descripción**
Id_Grupo_Evaluacion	Entero	Identificador del grupo de evaluación.
Nombre	Cadena	Nombre del grupo de evaluación.
Descripcion	Cadena	Descripción del grupo de evaluación.

Tabla 2.40 Descripción de la tabla: Tb_hdForma_Calificacion

Nombre: Tb_hdForma_Calificacion		
Descripción: Almacena los datos de las formas de calificación pertenecientes a los distintos Planes de Estudios.		
Atributo	**Tipo**	**Descripción**
Id_Forma_Calificacion	Entero	Identificador de la forma de calificación.
Nombre	Cadena	Nombre de la forma de calificación.
Descripcion	Cadena	Descripción de la forma de calificación.
Aprobado	Cadena	Valor que indica el aprobado de la forma de calificación.

Tabla 2.41 Descripción de la tabla: Tb_hdTipo_Evaluacion

Nombre: Tb_hdTipo_Evaluacion		
Descripción: Almacena los datos de los Tipos de Evaluaciones.		
Atributo	**Tipo**	**Descripción**
Id_Tipo_Evaluacion	Entero	Identificador del tipo de evaluación.
Nombre	Cadena	Nombre del tipo de evaluación.
Descripcion	Cadena	Descripción del tipo de evaluación.

Tabla 2.42 Descripción de la tabla: Tb_hdDocumento_Externo

Nombre: Tb_hdCargo		
Descripción: Almacena la información correspondiente a los cargos que puede ocupar una persona.		
Atributo	**Tipo**	**Descripción**
Id_Cargo	Entero	Identificador del cargo.
Nombre	Cadena	Nombre del cargo.
Descripcion	Cadena	Descripción del cargo.
Id_Estandar	Entero	Identificador que indica el valor numérico que puede tomar este atributo para el negocio.

Tabla 2.43 Descripción de la tabla: Tb_hdCargo

Nombre: Tb_hdDocumento_Externo		
Descripción: Almacena los datos de los documentos externos de los estudiantes.		
Atributo	**Tipo**	**Descripción**
Id_Documento	Entero	Identificador del documento.
Valor	Cadena	Valor del documento.
Id_Persona	Cadena	Identificador de la persona.

Tabla 2.44 Descripción de la tabla: Tb_hnNomenclador

Nombre: Tb_hnNomenclador		
Descripción: Almacena los nomencladores que serán utilizados en el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Nomenclador	Entero	Identificador del nomenclador.
Valor	Cadena	Valor del nomenclador.
Id_Estandar	Entero	Identificador que indica el valor numérico que puede tomar este atributo para el negocio.
Id_Nomenclador_Asociado	Entero	Identificador del nomenclador al cual está asociado este.

Tabla 2.45 Descripción de la tabla: Tb_hdActividad

Nombre: Tb_hdActividad		
Descripción: Almacena las actividades que poseen las carreras del centro de estudios.		
Atributo	**Tipo**	**Descripción**
Id_Actividad	Entero	Identificador de la actividad.
Doc_Asociado	Entero	Documento en el cual están descritos los lineamientos de la actividad.
Creditos	Entero	Créditos que aporta la actividad a las personas que la cursen.
Id_Carrera	Entero	Identificador de la carrera.

Tabla 2.46 Descripción de la tabla: Tb_hdValor

Nombre: Tb_hdValor		
Descripción: Almacena los valores que pueden tomar las formas de calificaciones definidas en el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Valor	Entero	Identificador del valor.
Valor	Cadena	Valor del valor.
Valor_Equivalente	Entero	Valor Equivalente del valor.
Id_Forma_Calificacion	Entero	Identificador de la forma de calificación.

Tabla 2.47 Descripción de la tabla: Tb_hrPlantilla_Campo

Nombre: Tb_hrPlantilla_Campo		
Descripción: Almacena los datos de las campos asociados a plantillas.		
Atributo	**Tipo**	**Descripción**
Id_Campo	Entero	Identificador del campo.
Id_Plantilla	Entero	Identificador de la plantilla.

Tabla 2.48 Descripción de la tabla:Tb_hnCampo

Nombre: Tb_hnCampo		
Descripción: Almacena los campos que serán utilizados en el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Campo	Entero	Identificador del campo.
Nombre	Cadena	Nombre del campo.
Es_Nomenclador	Boolean	Indica si un campo es nomenclador.
Descripcion	Cadena	Descripción del campo.

Tabla 2.49 Descripción de la tabla: Tb_hrPlantilla

Nombre: Tb_hrPlantilla		
Descripción: Almacena los datos de las plantillas.		
Atributo	**Tipo**	**Descripción**
Id_Plantilla	Entero	Identificador de la plantilla.
Nombre	Cadena	Nombre de la plantilla
Descripcion	Cadena	Descripción de la plantilla

Tabla 2.50 Descripción de la tabla: Tb_hdCarrera

Nombre: Tb_hdCarrera		
Descripción: Almacena los datos de las carreras de un centro de estudios.		
Atributo	**Tipo**	**Descripción**
Id_Carrera	Entero	Identificador de la carrera.
Nombre	Cadena	Nombre de la carrera.
Descripcion	Cadena	Descripción de la carrera.

Tabla 2.51 Descripción de la tabla: Tb_hrGrupo_Academico_Persona

Nombre: Tb_hrGrupo_Academico_Persona		
Descripción: Guarda la relación de los grupos académicos que han cursado los estudiantes.		
Atributo	**Tipo**	**Descripción**
Id_Persona	Cadena	Identificador de la persona.
Id_Grupo_Academico	Entero	Identificador del grupo académico.

Tabla 2.52 Descripción de la tabla: Tb_hdCreditos_Asignatura

Nombre: Tb_hdCreditos_Asignatura		
Descripción: Almacena los créditos asignados a los estudiantes en las distintas asignaturas de la carrera.		
Atributo	**Tipo**	**Descripción**
Id_Credito_Asignatura	Entero	Identificador del crédito de la asignatura.
Valor	Entero	Valor del crédito de la asignatura.
Fecha	Cadena	Fecha en que se registró el crédito.
Id_Persona	Cadena	Identificador de la persona.
Id_Grupo_Academico	Entero	Identificador del grupo académico.

Tabla 2.53 Descripción de la tabla: Tb_hdCreditos_Actividad

Nombre: Tb_hdCreditos_Actividad		
Descripción: Almacena los créditos asignados a los estudiantes en las distintas actividades de la carrera.		
Atributo	**Tipo**	**Descripción**
Id_Credito_Actividad	Entero	Identificador del crédito de la actividad.
Valor	Entero	Valor del crédito de la actividad.
Fecha	Cadena	Fecha en que se registra el crédito.
Id_Persona	Cadena	Identificador de la persona.
Id_Actividad	Entero	Identificador de la actividad.

Tabla 2.54 Descripción de la tabla: Tb_ hdPromedio_Credito

Nombre: Tb_hdPromedio_Credito		
Descripción: Almacena el promedio y los créditos de cada estudiante.		
Atributo	**Tipo**	**Descripción**
Id_Promedio_Credito	Entero	Identificador del promedio crédito.
Promedio	Entero	Promedio de cada estudiante.
Credito	Entero	Suma de los créditos de cada estudiante.
Id_Persona	Cadena	Identificador de la persona.

Tabla 2.55 Descripción de la tabla: Tb_hdPremio_Bonificacion

Nombre: Tb_hdPremio_Bonificacion		
Descripción: Almacena los premios y bonificaciones que han recibido los estudiantes en su carrera.		
Atributo	**Tipo**	**Descripción**
Id_Registro_Bonificacion	Entero	Identificador del registro de bonificación.
Observacion	Cadena	Descripción del motivo por el cual se registra la nota.
Id_Persona	Cadena	Identificador de la persona.
Id_Valor	Entero	Identificador del valor.
Id_Bonificacion	Entero	Identificador de la bonificación.
Id_Asignatura	Entero	Identificador de la asignatura.

Tabla 2.56 Descripción de la tabla: Tb_ hrCampo_Persona

Nombre: Tb_hrCampo_Persona		
Descripción: Almacena los campos que se le han llenado a las personas.		
Atributo	**Tipo**	**Descripción**
Id_Persona	Cadena	Identificador de la persona.
Id_Campo	Entero	Identificador del campo.

Tabla 2.57 Descripción de la tabla: Tb_hrPlantilla_Persona

Nombre: Tb_hrPlantilla_Persona		
Descripción: Almacena las plantillas que se le han llenado a cada persona.		
Atributo	**Tipo**	**Descripción**
Id_Plantilla	Entero	Identificador de la plantilla.
Id_Persona	Cadena	Identificador de la persona.
Fecha	Cadena	Fecha en que se le llena la plantilla a la persona.
Autor	Cadena	Persona que llena la plantilla.

Tabla 2.58 Descripción de la tabla: Tb_hrPlantilla_Persona

Nombre: Tb_hdCampo_Persona_Texto		
Descripción: Almacena los valores de tipo texto de los campos llenados a las personas.		
Atributo	**Tipo**	**Descripción**
Id_Campo	Entero	Identificador del campo.
Id_Persona	Cadena	Identificador de la persona.
Valor	Cadena	Valor que toma el campo.
Fecha	Cadena	Fecha en que se llena el campo.

Tabla 2.59 Descripción de la tabla: Tb_hdCampo_Persona_Nomenclador

Nombre: Tb_hdCampo_Persona_Nomenclador		
Descripción: Almacena los valores de tipo nomenclador de los campos llenados a las personas.		
Atributo	**Tipo**	**Descripción**
Id_Campo	Entero	Identificador del campo.
Id_Persona	Cadena	Identificador de la persona.
Id_Nomenclador	Entero	Identificador del nomenclador.
Fecha	Cadena	Fecha en que se llena el campo.

Tabla 2.60 Descripción de la tabla: Tb_hrEstructura_Persona

Nombre: Tb_hrEstructura_Persona		
Descripción: Almacena la relación existente entre las personas y las estructuras.		
Atributo	**Tipo**	**Descripción**
Id_Estructura	Entero	Identificador de la estructura.
Id_Persona	Cadena	Identificador de la persona.
Id_Edicion_Momento	Entero	Identificador de la edición de momento.
Id_Cargo	Entero	Identificador del cargo.
Id_Asignatura	Entero	Identificador de la asignatura.
Estructura_Principal	Boolean	Indica si es la estructura principal de la persona.
Fecha	Cadena	Fecha en que se le asigna la estructura a la persona.

Tabla 2.61 Descripción de la tabla: Tb_hdEstructura

Nombre: Tb_hdEstructura		
Descripción: Almacena las estructuras de un centro de estudios.		
Atributo	**Tipo**	**Descripción**
Id_Estructura	Entero	Identificador de la estructura.
Nombre	Cadena	Nombre de la estructura.
Descripción	Cadena	Descripción de la estructura.
Nombre_Tipo_Estructura	Cadena	Nombre del tipo de estructura.
Id_Estructura_Padre	Entero	Identificador de la estructura padre.

2.7 Análisis y optimización de querys

El análisis de optimización de consultas constituye un elemento importante en la implementación de una base de datos, de poco sirve tener un buen modelo si las consultas realizadas al mismo no lo aprovechan, lo cual influye negativamente en la eficiencia y rendimiento del sistema. El objetivo de la optimización es mejorar los tiempos de respuesta del gestor de base de datos, haciendo un uso correcto de sus recursos disponibles. Es importante tener en cuenta la complejidad de las consultas, pues de este aspecto depende el tiempo que se necesite para dar respuesta a la misma, y teniendo presente que no siempre se alcanza una respuesta óptima cuando estas son muy complejas.

Las peticiones a la base de datos o consultas se utilizan para almacenar, visualizar o actualizar información existente en la misma. Estas pueden expresarse de diferentes maneras, donde cada una sugiere una estrategia para encontrar la respuesta y, por lo tanto, algunas pueden ser más eficientes que otras.

Criterios a tener en cuenta durante la implementación de las consultas al modelo propuesto:

- Todas las tablas del modelo presentan un *PRIMARY KEY*, todos tienen presente además la propiedad *UNIQUE* como punto de validación. De ser posible, en todas las consultas serán utilizados estos índices.
- Las cláusulas *HAVING* son muy costosas para el servidor de base de datos, por lo que se propone utilizar la cláusula *WHERE* siempre que sea posible.
- Es importante hacer uso de los índices primarios en las condiciones JOIN, mediante lo cual se gana en eficiencia en términos de rendimiento, ya que se estaría evaluando una condición de igualdad numérica lo cual es más factible que una condición de igualdad de cadenas de caracteres.
- Los *triggers* y las funciones en lenguajes de procedimiento son indispensables en el desarrollo de complejas aplicaciones. Mientras más trabajo realice la base de datos, mejores son las aplicaciones, se hacen más portables.
- Es importante conocer las particularidades del manejador de consultas del gestor utilizado, a través del conocimiento de la implementación de las rutinas de procesamiento de consultas. Por ejemplo: al existir una condición disyuntiva en la consulta si se usan o no los índices existentes.

Pasos a seguir para la confección de las consultas, los ejemplos expuestos son consultas realizadas en lenguaje PostgreSQL:

1. Cuando se realiza una consulta que incluya varias tablas es bueno identificar a qué tabla pertenece cada campo, así el gestor de datos no tiene que encontrar a qué tabla pertenece cada uno de los campos:

```
SELECT
tb_hdpersona.id_persona AS persona,
tb_hdcampo_persona_texto.valor AS valor,
 tb_hncampo.nombre AS nombre_campo
FROM
tb_hdcampo_persona_texto INNER JOIN tb_hdpersona ON
(tb_hdcampo_persona_texto.id_persona = tb_hdpersona.id_persona)
```

INNER JOIN tb_hncampo **ON** (tb_hdcampo_persona_texto.id_campo=tb_hncampo.id_campo)

2. El orden de las tablas en la consulta es importante, un mal uso de estas puede implicar un mayor consumo de recursos.

Consulta 1:

```
SELECT
tb_hncampo.nombre AS campo,
tb_hdcampo_persona_texto.valor AS valor_campo,
tb_hdpersona.id_persona AS persona
FROM
tb_hdcampo_persona_texto
INNER JOIN tb_hncampo ON (tb_hdcampo_persona_texto.id_campo=tb_hncampo.id_campo)
INNER JOIN tb_hdpersona ON (tb_hdcampo_persona_texto.id_persona= tb_hdpersona.id_persona)
WHERE tb_hdpersona.id_persona = wefwehrghewg2
```

Consulta 2:

```
SELECT
tb_hdpersona.id_persona AS persona,
tb_hdcampo_persona_texto.valor AS valor,
tb_hncampo.nombre AS nombre_campo
FROM
tb_hdpersona  INNER JOIN tb_hdcampo_persona_texto ON
(tb_hdpersona.id_persona= tb_hdcampo_persona_texto.id_persona)
INNER JOIN tb_hncampo ON (tb_hdcampo_persona_texto.id_campo= tb_hncampo.id_campo)
WHERE tb_hdpersona.id_persona = wefwehrghewg2
```

Las dos consultas anteriores devuelven los mismos datos, pero la primera consume más tiempo que la segunda, esta última primero busca todos los campos que están asociados a un identificador de una persona y luego, dado estos identificadores, busca el nombre del campo asociado, por el contrario, la primera consulta busca los nombres de todos los campos y luego selecciona los que pertenecen a una persona determinada. Estas consultas realizadas en pequeñas bases de datos y donde se manejen pocos

datos, no significan mucho; sin embargo, para grandes bases de datos que manejen grandes volúmenes de información y que las consultas que se le realicen tengan un alto nivel de complejidad, es de gran importancia tener en cuenta la forma de elaborar las consultas para agilizar el proceso de respuesta del servidor y obtener un buen rendimiento del mismo.

2.8 Conclusiones

En este capítulo se han descrito un conjunto de requerimientos funcionales pertenecientes a los módulos, del área de pre-grado, que componen hoy la nueva versión de Akademos, es válido señalar que de todos los requerimientos se han seleccionado sólo los que por su relevancia deben pasar a la Base de Datos Histórica. Por otra parte los requerimientos no funcionales definidos son aquellos que están vinculados directamente al área de la base de datos. Como parte de la descripción de la arquitectura del sistema se han mencionado los frameworks que interactúan de manera directa con la base de datos y se ha explicado la interacción entre ellos, así como las características más importantes de cada uno. Se hace una descripción de la solución propuesta y como resultado se obtuvo el diagrama de clases persistentes y el diagrama entidad – relación de los datos, este último cumple con las restricciones de que todas sus tablas no tienen un mismo nivel de normalización para garantizar un acceso mucho más simple a la información, y además como ventajas fundamentales proporciona una mayor organización a la base de datos central y un mejor rendimiento del servidor de la misma mediante el uso de los *Tablespaces.*

Capítulo # 3

Validación del diseño realizado

3.1 Introducción

En el presente capítulo se realiza una validación de la propuesta de solución para el diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0, que se obtuvo en el capítulo anterior. Además se describe el tratamiento dado a importantes aspectos para el área de la base de datos como son la normalización, redundancia, integridad y seguridad de la información en el diseño propuesto, elementos donde se han centrado los esfuerzos de los diseñadores. El objetivo fundamental de este capítulo es validar el diseño realizado con aspectos teóricos y funcionales.

3.2 Validación teórica del diseño

Los aspectos que en este epígrafe se incluyen tienen que ver con la exactitud, consistencia y confiabilidad de la información y con la privacidad y confidencialidad de los datos. Las bases de datos tienen dentro de sus características elementos que pueden ser utilizados para garantizar la calidad de la información almacenada y procesada.

3.2.1 Integridad

3.2.1.1 Integridad de datos

Cuando se colocan los datos de una aplicación en un lugar común siempre se corren riesgos de que esta sea modificada o dañada por personas que no tienen la debida autoridad y responsabilidad sobre la misma, y en muchas ocasiones, esto sucede sin la menor intención. Precisamente con el objetivo de evitar este tipo de situaciones es que se tiene en cuenta la integridad de los datos que están almacenados en una base de datos cuando esta es implementada.

En las bases de datos relacionales es muy común el uso de sentencias *INSERT*, *DELETE* o *UPDATE,* las cuales pueden causar pérdida de la integridad de los datos almacenados. Por la importancia de esta base de datos y por las características de los datos que almacena no se permitirán realizar sentencias *DELETE* o *UPDATE,* solamente *INSERT,* para introducir nuevos datos a la misma, de esta forma:

- Se evita el cambio de información existente válida por una no válida. Por ejemplo el cambio de información académica o personal de un estudiante que se ha graduado con anterioridad.
- Los cambios pueden ser aplicados por partes, por ejemplo, si se cambia el plan de estudio donde se graduó un estudiante y no se cambian las evaluaciones y las asignaturas al nuevo plan de estudio, se generarían entonces inconsistencias en los datos almacenados.

3.2.1.2 Integridad de clave

Para definir la clave primaria se debe pasar por el proceso de selección de la misma, que incluye definir las claves candidatas. Estas pueden ser varias y estar compuestas por un número diferente de atributos, de ellas se elegirá una que será la clave primaria, el resto de las claves candidatas se definen como llaves alternativas.

Las reglas de integridad son restricciones que definen los estados de consistencia de la base de datos. En la Base de Batos Histórica fue aplicada la regla de integridad llamada Regla de la Entidad que parte del hecho que toda tabla posee una llave primaria y además, dicta que ningún atributo primo puede ser nulo.

En la propuesta, así como en la base de datos central, ningún atributo de una clave primaria toma valores nulos, a su vez estos valores son distintos para cada posible tupla; esto se garantiza a través del uso de restricciones *UNIQUE* o *PRIMARY KEY*, o el uso de propiedades *IDENTITY*. Esto permite garantizar que no se tendrá información repetida o discordante para un valor de clave y puede ser usada como control, para evitar la inclusión de información inconsistente en las tablas.

3.2.1.3 Integridad de dominio

La integridad de dominio garantiza el dominio de los datos que se guardarán de cada uno de los atributos de las tablas. Cada columna de la Base de Datos Histórica tiene definido su dominio, en correspondencia a los definidos en la base de datos central, de manera tal que en el paso de la información de una base de datos a otra no se generen errores. Durante el proceso de confección del modelo se ha tenido en cuenta la restricción del tipo de datos, el formato a través de las restricciones *CHECK* y de las reglas, o el rango

de valores posibles a través de restricciones *FOREIGN KEY*, definiciones *DEFAULT* y definiciones *NOT NULL*.

3.2.1.4 Integridad referencial

La integridad referencial se puede decir que es la más importante de las integridades dentro del área de la base de datos, la misma da la medida de qué tan exacta o correcta es la información que se maneja. Implica que en todo momento los datos sean correctos y sin repeticiones innecesarias evitando la pérdida de datos y relaciones incorrectas. Esto se refleja en el modelo propuesto con el siguiente ejemplo: los atributos *Id_Persona e Id_Grupo_Academico* de la tabla *Tb_hrGrupo_Academico_Persona* son llaves foráneas, debido a que son llaves primarias que provienen de las tablas *Tb_hdPersona y Tb_hdGrupo_Academico* respectivamente. Además en la propuesta se garantiza el cumplimiento de las siguientes restricciones:

- Los *Id_Persona e Id_Grupo_Academico* poseen el mismo dominio tanto en sus tablas de origen como en la *Tb_hrGrupo_Academico_Persona.* Esto también se garantiza en todas las tablas donde se utilicen llaves foráneas de otras tablas pues estas poseen el mismo dominio en ambas tablas.
- Ambos identificadores están presentes en sus tablas de origen una vez que hayan sido insertados en la *Tb_hrGrupo_Academico_Persona*. Todos los identificadores se encuentran presentes en las tablas de origen para ser utilizados en las tablas donde se referencian.

El gestor de datos utilizado, el PostgreSQL, maneja de manera muy positiva la integridad referencial, el mismo asegura en todo momento que, en un diseño elaborado correctamente, las referencias por llaves foráneas sean las correctas.

3.2.1.5 Integridad semántica

Durante el diseño de la Base de Datos Histórica se tuvieron en cuenta algunos elementos que garantizan la integridad semántica de los datos como son:

- La definición de reglas de integridad que se almacenan en el diccionario del gestor como parte integrante de la descripción de los datos con lo que son fáciles de entender y de cambiar, esto facilita su mantenimiento. Esto permite al gestor controlar las transacciones y detectar violaciones de integridad.

- El uso de restricciones estáticas o dinámicas que, con una adecuada declaración, permiten una gran flexibilidad. Entre las restricciones más utilizadas se encuentran las de unicidad, las de valores no nulos, etc.

De manera general estas son las consideraciones que se han tenido en cuenta durante el diseño de la propuesta, asegurando que el mismo cumpla con la integridad de la información requerida, debido fundamentalmente a la importancia que se le atribuye a este aspecto dentro de la base de datos.

3.2.2 Normalización de la base de datos

Una base de datos tiene que ser diseñada antes de que pueda ser creada y usada. El diseño debe ajustarse a estándares que permitan el ahorro de memoria, acceso rápido, fácil mantenimiento, portabilidad, facilidad de futuros mejoramientos, buen desempeño y eficiencia de costos, entre otros. El diseño lógico final de una base de datos debe ser tal que equilibre un desempeño óptimo junto con la integridad de la información. Esto puede ser logrado a través de un proceso conocido como normalización que consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional. Uno de los conceptos fundamentales que se maneja en la normalización es el de dependencia funcional.

El proceso de normalización permite evitar algunos elementos que no son deseados en las bases de datos como:

- Redundancia o repetición de los datos en el sistema.
- Inconsistencias de actualización de datos como resultado de las actualizaciones parciales y la redundancia de la información.
- Anomalías de borrado o pérdidas no intencionadas de datos.
- Anomalías de inserción o imposibilidad de adicionar datos en la base de datos debido a la ausencia de otros datos.

Como resultado de la normalización realizada se obtuvo una propuesta de diseño libre de anomalías en la actualización y mejora de la independencia de los datos.

El resultado del proceso de normalización es la obtención del modelo físico de la base de datos, en sus distintas formas normales, las utilizadas en el modelo se describen a continuación:

Definición de la clave: Antes de proceder a la normalización de las tablas lo primero que debe definirse son las claves, estas deberán contener un valor único para cada registro (no podrán existir dos valores iguales en toda la tabla) [13]. Una vez que las claves hayan sido definidas se puede comenzar a normalizar las tablas del modelo. En el caso de la Base de Datos Histórica, como se partió de un modelo ya existente no fue necesario definir las claves pues ya habían sido identificadas.

Primera Forma Normal: Se dice que una tabla se encuentra en primera forma normal (1FN) si y solo si cada uno de los campos contiene un único valor para un registro determinado [14]. Por lo tanto se considera que el conjunto de relaciones estará en primera forma normal cuando todos los atributos tengan dominios atómicos, para alcanzar esto se deben eliminar todos los atributos compuestos y multivaluados.

Segunda Forma Normal: Para que una relación esté en segunda forma normal (2FN) tiene necesariamente que estar en 1FN. La 2FN compara todos y cada uno de los campos de la tabla con la clave definida. Si todos los campos dependen directamente de la clave se dice que la tabla está en 2NF [15].

La segunda forma normal se basa en el concepto de dependencia funcional parcial. Las dependencias funcionales parciales ocurren cuando en una dependencia se elimina un subconjunto de la clave y la dependencia continúa siendo válida. Por lo tanto un conjunto de relaciones estará en segunda forma normal cuando no existan dependencias funcionales parciales.

Tercera Forma Normal: Se dice que una tabla está en tercera forma normal (3FN) si y solo si los campos de la tabla dependen únicamente de la clave, dicho en otras palabras los campos de las tablas no dependen unos de otros [16].

Una relación está en tercera forma normal si, y solo si, está en 2FN y, además, cada atributo que no está en la clave primaria no depende transitivamente de ella. La tercera forma normal se basa en el concepto de dependencia funcional transitiva. Por tanto para pasar una relación de 2FN a 3FN hay que eliminar las dependencias transitivas.

La base de datos que se ha obtenido no posee un nivel de normalización estándar para todas sus tablas, pues el objetivo de la investigación es lograr mayor velocidad de respuesta a las consultas que se le realicen a la misma y acceder con más facilidad y rapidez a la información contenida en ella, por lo que no es necesario que todas las tablas estén en una misma forma normal. El modelo oscila entre la segunda y

tercera forma normal, el uso de niveles más altos de normalización podría traer consigo una base de datos más relacional pero a la vez más compleja para trabajar y los tiempos de respuesta de varias de las consultas no serían los más óptimos para responder a las peticiones del sistema.

3.2.3 Análisis de redundancia de la información

Es de gran importancia para el sistema que la información que se maneje no se encuentre duplicada. En estudios realizados con anterioridad se han encontrado ejemplos que presentan este problema, que es la causa de inconsistencias en los sistemas lo cual afecta el buen rendimiento de los mismos y la confiabilidad de la información que se maneja.

En el estudio realizado a la primera versión del sistema Akademos fueron identificadas como las áreas que presentan estos problemas la de matrícula y registro de notas. En la propuesta de la base de datos central para el sistema Akademos v_2.0 se eliminan estas inconsistencias, el método utilizado en el diseño de esta última se tiene en cuenta en el diseño de la Base de Datos Histórica. A continuación se exponen algunas de las ideas utilizadas:

- El uso de una tabla de nomencladores para todos aquellos datos comunes de la información personal dentro de la base de datos.
- El uso de una tabla de campos la cual se encarga de almacenar los campos comunes que pueden tener los distintos formularios definidos en el sistema.

3.2.4 Análisis de la seguridad de la base de datos

En el capítulo 2 del presente trabajo de tesis se hace referencia a algunas de las políticas de seguridad definidas para el área de la base de datos que se deben tener en cuenta durante su desarrollo, implementación y despliegue. La base de datos central del sistema Akademos v_2.0, como se ha mencionado con anterioridad, estará dividida por esquemas, uno de estos esquemas es utilizado para el manejo de la seguridad del sistema. En el mismo se gestionan los permisos e incidencias de cada usuario que tiene acceso al sistema, mediante un conjunto de tablas que establecen qué acciones puede realizar cada usuario o grupos de estos sobre cada información almacenada en la base de datos.

Como parte del sistema, la Base de Datos Histórica, se ajusta a esta propuesta y agrega algunas medidas más dada la importancia de la protección contra accesos ilegales, alteraciones malintencionadas o destrucción de los datos almacenados en ella. Algunas de estas medidas son:

- La autenticación a través de un usuario y contraseña determinado para el esquema de la Base de Datos Histórica.
- PostgreSQL permite gestionar el acceso de los clientes basados en máquinas (host), estos no deben confundirse con los permisos de un usuario de PostgreSQL a los objetos dentro de la base de datos. De manera simple permite determinar quién está autorizado para conectarse a qué base de datos desde qué máquinas y además guardar cómo debe ser probada su autenticidad para obtener el acceso.

Es de gran importancia el uso de las potencialidades que brinda el gestor de base de datos en cuestiones de seguridad, el mismo puede mantener catálogos de los permisos de cada usuario independientemente de los definidos en el sistema, esto permite conocer quién puede realizar consultas, actualizaciones, etc., según el juego de privilegios de acceso y de restricciones aplicables a cada objeto de la base de datos (tablas, vistas y secuencias).

Las técnicas de copia de seguridad y restauración compatibles con los sistemas de bases de datos deben garantizar la preservación de los datos ante cualquier imprevisto o falla del sistema. Aún así, el administrador de base de datos debe definir procedimientos para recuperar la información perdida. La Base de Datos Histórica se ajusta a las políticas de gestión de copias de seguridad establecidas para la base de datos central del sistema, aunque al ser un esquema de ella, pueden realizarse copias del mismo una vez en el curso, pues la información que este contiene no se modificará, sólo será actualizada una vez al año.

3.2.5 Trazabilidad de las acciones

La trazabilidad es la capacidad que posee un sistema para rastrear, reconstruir o establecer relaciones entre objetos monitoreados, para identificar y analizar situaciones específicas o generales en los mismos. Al referirse a la trazabilidad se puede decir entonces que existe la posibilidad de conocer o establecer, entre la información que se maneja de un sistema, quién ha realizado qué, y poder llevar el sistema a un estado anterior.

Basado en esta idea se define en el esquema de seguridad de la base de datos central una bitácora o registro diario, lo cual permite conocer todas las operaciones realizadas en el sistema por los usuarios.

De manera general en la tabla de incidencias se registra el usuario, la dirección de IP, la hora y fecha de la operación, la acción realizada sobre qué objeto y los valores antes y después de la operación. De esta manera se tiene un control sobre todas las acciones realizadas sobre el sistema.

3.2.6 Tablespaces

Para la mejora del rendimiento del servidor se propone en esta investigación el uso de los *Tablespaces.* Después de haber analizado el diagrama realizado y las consultas que se le pueden realizar al mismo se definieron varios *Tablespaces*:

- Uno para las tablas críticas que no son más que aquellas que tienen el mayor volumen de información y que son las más consultadas en la base de datos.
- Ejemplo: en el TBS_Critico estarían las tablas Tb_hnNomenclador, Tb_hdAsignatura, Tb_hrCampo_Persona_Nomenclador, Tb_hdEdicion_Momento, Tb_hdRegistro_Nota, etc.
- Otro para almacenar los índices de todas las tablas con el objetivo de realizar búsquedas más rápidas.
- Ejemplo: en el TBS_IX estarían los índices de las tablas de la base de datos, algunos de ellos son: Id_Campo, Id_Nomenclador, Id_Persona, Id_Asignatura, etc.
- Otro para las tablas que se consultan con frecuencia pero no llegan a ser críticas.
- Ejemplo: en el TBS_Media estarían las tablas Tb_hrCampo_Persona, Tb_hrPlantilla_Campo, Tb_hrGrupo_Academico_Persona, etc.
- Por último, se debe crear uno para guardar las tablas que se consultan muy poco.
- Ejemplo: en el TBS_SL estarían las tablas Tb_hdCargo, Tb_hdPlantilla, Tb_hdBonificacion, Tb_hdAsistencia, etc.

Para crear un *Tablespace* en Postgre debe utilizarse el comando CREATE TABLESPACE de la siguiente forma:

```
CREATE TABLESPACE "TBS_Critico"
OWNER "postgres"
LOCATION 'D:/Tablespace';
```

Siempre debe especificarse el gestor y la localización de la carpeta donde se almacenarán los datos, especificando el disco donde esta se encuentra. La carpeta siempre debe estar vacía para poder crear el *Tablespace* y además, el usuario “postgres” debe tener todos los permisos sobre la misma.

Para transferir los datos de una tabla o los índices de las tablas para un *Tablespace* debe utilizarse el siguiente comando:

ALTER TABLE "Tb_hnNomenclador" SET TABLESPACE "TBS_Critico"

Para eliminar un *Tablespace* deben haberse eliminado todos los datos que este almacenaba, el *Tablespace* debe estar seleccionado y se utilizará el comando: DROP TABLESPACE.

3.3 Validación Funcional

Luego de realizar una valoración teórica de la propuesta de diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0, se proponen algunos resultados obtenidos como parte de la implementación de la base de datos. Para realizar esta prueba se ha montado en el gestor PostgreSQL la base de datos central del sistema con sus esquemas y la Base de Datos Histórica con las configuraciones adecuadas de seguridad y administración.

Durante el proceso de carga del esquema histórico se ha utilizado la herramienta EMS Data Generator For PostgreSQL 2005 que, pese a ser una herramienta propietaria, permite generar un gran volumen de datos de prueba a las tablas de bases de datos de manera simultánea, además ofrece un control sobre integridad referencial durante la generación de los mismos y ofrece soporte a todos los tipos de datos definidos en el gestor.

El uso de este software ha permitido chequear la correcta implementación de la integridad referencial y de dominio dentro cada tabla del diseño pues permite, para cada atributo, definir los valores que este admite en dependencia del tipo que se le ha definido, y en el caso del uso de claves foráneas, chequea que el valor al que se hace referencia exista en la tabla de origen.

Se ha generado un volumen de 8000000 de datos entre varias de las tablas del modelo, este valor está definido en correspondencia a la cantidad de datos que el esquema histórico deberá manejar en el momento que forme parte del sistema.

Luego de la carga de la base de datos se procede a realizar algunas de las consultas más frecuentes a las que se enfrentará el modelo para obtener los tiempos de respuesta y comprobar la validez de los datos obtenidos.

Ejemplo 1: La siguiente consulta devuelve todas las notas finales de un estudiante:

```
SELECT
tb_hdregistro_nota.id_persona AS ID_PERSONA,
tb_hdasignatura.nombre AS ASIGNATURA,
tb_hdregistro_nota.nombre_concepto_nota AS CONCEPTO,
tb_hdvalor.valor AS NOTA,
tb_hdedicion_momento.nombre AS EDICION
FROM
tb_hdasignatura INNER JOIN tb_hdgrupo_academico ON
(tb_hdasignatura.id_asignatura= tb_hdgrupo_academico.id_asignatura)
INNER JOIN tb_hdregistro_nota ON
(tb_hdgrupo_academico.id_grupo_academico=tb_hdregistro_nota.id_grupo_academico)
INNER JOIN tb_hdedicion_momento ON
(tb_hdgrupo_academico.id_edicion_momento=tb_hdedicion_momento.id_edicion_momento)
INNER JOIN tb_hdvalor ON (tb_hdregistro_nota.id_valor=tb_hdvalor.id_valor)
WHERE tb_hdregistro_nota.id_persona = 3rgg4gtd4444
```

El ejemplo anterior devolvió como resultado para un estudiante determinado entre 70 y 89 filas, las mismas se devolvieron en un tiempo de 0.15 segundos aproximadamente.

Ejemplo 2: Devuelve todos los estudiantes que se graduaron en un curso determinado.

```
SELECT
 tb_hrcampo_persona.id_persona AS ID_PERSONA,
 tb_hnnomenclador.valor AS Año de Graduación
FROM
tb_hrcampo_persona_nomenclador INNER JOIN tb_hnnomenclador ON
(tb_hrcampo_persona_nomenclador.id_nomenclador =tb_hnnomenclador.id_nomenclador)
```

INNER JOIN tb_hrcampo_persona **ON**

(tb_hrcampo_persona.id_persona=tb_hrcampo_persona_nomenclador.id_persona)

WHERE tb_hnnomenclador.id_nomenclador = 102

El ejemplo anterior devolvió como resultado para un curso determinado entre 700 y 750 filas, las mismas se devolvieron en un tiempo de 0.31 segundos aproximadamente.

Es importante recordar que estos datos no son los finales que manejará la Base de Datos Histórica, son valores aleatorios generados por el software utilizado en la carga de la base de datos, por lo que los resultados obtenidos pueden diferir en cuanto al número, a los que se obtendrán una vez que se utilice el modelo junto al sistema.

3.4 Resultados

Después de haber realizado la propuesta de diseño y las pruebas al mismo, se han obtenido un conjunto de resultados:

- La Base de Datos Histórica cumple con los requisitos seleccionados del sistema Akademos v_2.0.
- La base de datos no presenta un nivel estándar de normalización lo que permite un acceso mucho más rápido y simple a la información histórica contenida en la misma.
- Con la utilización de los *Tablespaces* se garantiza el mejoramiento del rendimiento del servidor lo que conlleva a la mejora de la velocidad de respuesta del mismo.
- Han sido implementadas correctamente las restricciones de integridad y seguridad lo que garantiza la validez de los datos históricos almacenados.

3.5 Conclusiones

El diseño propuesto ha sido validado teórica y funcionalmente para demostrar que el mismo ha sido construido correctamente y que satisface las necesidades del proyecto.

En este capítulo se han analizado un conjunto de criterios que se han tenido en cuenta para validar la propuesta de diseño de la Base de Datos Histórica, como son: la integridad, la redundancia de información y la normalización.

Ha sido analizada además la seguridad, un aspecto de gran importancia para una base de datos, en este sentido se proponen varios niveles de seguridad, que incluye el gestor utilizado y la autenticación al esquema histórico mediante un usuario y contraseña precisamente por la información que se maneja, pues la misma no debe ser alterada ni dañada.

Se describieron, además, los resultados obtenidos luego de haber culminado las pruebas de carga y estrés a la propuesta de solución.

Conclusiones Generales

Dentro de la nueva propuesta de diseño de base de datos para la versión 2.0 del sistema Akademos, la Base de Datos Histórica tiene gran importancia, pues constituye una mejora en cuanto a organización de la información histórica y rendimiento del servidor de la base de datos central del sistema, lo que ha quedado demostrado durante la realización de este trabajo.

Para la realización de la propuesta de solución se partió del diseño de la base de datos de la versión 2.0 que ha dado solución a la deficiencias, en el área de base de datos, de la versión anterior del software; se tuvieron en cuenta, además, las experiencias de otros sistemas de gestión de información en cuanto al manejo de datos históricos y el estudio de varios temas vinculados al tratamiento de los datos históricos que, sin duda alguna, han sido de vital importancia para la investigación.

El diseño de la Base de Datos Histórica de la gestión académica de Akademos v_2.0, ha sido validado teórica y funcionalmente para garantizar que el mismo presenta la calidad necesaria y esperada. En su validación teórica se han teniendo en cuenta importantes criterios como son la normalización, integridad y seguridad de los datos, con los que ha cumplido satisfactoriamente según las restricciones y características del mismo. Como parte de la validación funcional, este diseño ha sido implementado utilizando las herramientas propuestas en la investigación, al realizarse las pruebas de su funcionamiento se obtuvieron buenos resultados en las mismas, aunque estos pudieran variar con su posterior despliegue e interacción con el usuario final; podemos concluir entonces que los objetivos planteados en este trabajo de tesis se han cumplido en su totalidad.

Recomendaciones

La Base de Datos Histórica que ha sido diseñada para la gestión académica de Akademos v_2.0 permite el soporte de varios reportes que se pueden realizar en alguna circunstancia necesaria sin afectar el rendimiento de la base de datos central del sistema, pues se creó teniendo en cuenta las necesidades de este tipo de información que requieren, en ocasiones, los directivos de la Universidad.

Las metas planteadas con este trabajo se han cumplido y en base a los resultados obtenidos se recomienda que:

- Se utilice la propuesta de diseño de la Base de Datos Histórica presentada en este trabajo y que sea implementada en el sistema Akademos v_2.0 por las ventajas que proporciona.
- Se le de seguimiento al trabajo realizado hasta el momento, pues el sistema Akademos v_2.0 aún está siendo desarrollado y pueden realizarse cambios en la base de datos central que afecten este diseño.
- El uso de los *Tablespaces* sea aplicado a toda el área de base de datos para garantizar una mejora considerable en el rendimiento del servidor como ha quedado demostrado en este trabajo de tesis.

Bibliografía Referenciada

1. **Orallo, José Hernández.** *La disciplina de los sistemas de Bases de Datos. Historia, Situación Actual y Perspectivas.* Valencia : s.n., 2002. Disponible en: http://palomo.usach.cl/Docs/BD/JHernandezO-La_disciplinade_los_sistemas.pdf

2. **Llerena, Roberto y Castro, Yissell María.** *Trabajo de Diploma Propuesta de diseño de la Base de Datos de Akademos v2.0.* La Habana, Cuba : s.n., 2008.

3. **Hernández, Andry Suárez.** *Trabajo de Diploma Arquitectura para Akademos 2.0.* La Habana, Cuba : s.n., 2008.

4. **Machado, Yenier Figueroa.** *Trabajo de Diploma Diseño de la capa de datos del Sistema de Gestión de Inventarios y Almacenes.* La Habana, Cuba : s.n., 2007.

5. Base de datos en castellano Data Warehousing. *Base de datos en castellano Data Warehousing.* [En línea] [Citado el: 8 de diciembre de 2008.] Disponible en: http://www.programacion.com/bbdd/tutorial/warehouse/4/

6. **Kronos y Tramullas, Jesús.** Introducción a la Documática Sección 2: Los sistemas de bases de datos y los SGBD. [En línea] 1997-2007. [Citado el: 17 de enero de 2009.] Disponible en: http://tramullas.com/documatica/2-4.html

7. **Soto, Lauro.** MiTecnológico Funciones del Gestor de Base da Datos. [En línea] [Citado el: 20 de enero de 2009.] Disponible en: http://www.mitecnologico.com/Main/FuncionesGestorBaseDatos

8. **Llerena, Roberto y Castro, Yissell María.** *Trabajo de Diploma Propuesta de diseño de la Base de Datos de Akademos v2.0.* La Habana, Cuba : s.n., 2008.

9. **Autores, C. D.** *Método de Desarrollo Adaptable.* Colombia : s.n., 2009. Disponible en: http://pisis.unalmed.edu.co/cursos/material/3004582/1/parte%201.ppt

10. **Villar, Malay Rodríguez.** *Trabajo de Diploma Introducción de procedimientos ágiles en la producción de software en la facultad 7 de la Universidad de las Ciencias Informáticas.* La Habana, Cuba : s.n., 2007.

11. **Peñalver Romero, Gladys Marsi.** *MA-GMPR-UR2 Metodología ágil para proyectos de software libre.* UCI : s.n., 2008.

12. **Llerena, Roberto y Castro, Yissell María.** *Trabajo de Diploma Propuesta de diseño de la Base de Datos de Akademos v2.0.* La Habana, Cuba : s.n., 2008.

13. Base de Datos en castellano. *Base de Datos en castellano Tutoriales: Modelo Relacional*. [En línea] [Citado el: 15 de abril de 2009.] Disponible en: http://www.programacion.com/bbdd/tutorial/modrel/2/

14. Base de Datos en castellano. *Base de Datos en castellano Tutoriales: Modelo Relacional*. [En línea] [Citado el: 15 de abril de 2009.] Dsiponible en: http://www.programacion.com/bbdd/tutorial/modrel/2/

15. Base de Datos en castellano. *Base de Datos en castellano Tutoriales: Modelo Relacional*. [En línea] [Citado el: 15 de abril de 2009.] Dsiponible en: http://www.programacion.com/bbdd/tutorial/modrel/2/

16. Base de Datos en castellano. *Base de Datos en castellano Tutoriales: Modelo Relacional*. [En línea] [Citado el: 15 de abril de 2009.] Dsiponible en: http://www.programacion.com/bbdd/tutorial/modrel/2/

Bibliografía Consultada

1. **Andrés, María Mercedes Marqués.** Base de datos. [En línea] 12 de febrero de 2001. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node1.html

2. **Autores, C. D.** *Data Warehouse.* 20/10/2008. Disponible en: http://www.uaem.mx/posgrado/mcruz/cursos/miic/warehouse.pdf

3. **Autores, C. D.** Estudiando BI. [En línea] 29 de 11 de 2007. [Citado el: 15 de octubre de 2008.] Disponible en: http://estudiandobi.blogspot.com/2007/11/data-warehouse-en-profundidad.html

4. **Autores, C. D.** *Metodologías Ágiles en el Desarrollo de Software.* Valencia : s.n., 21/02/2009.

5. **Autores, C. D.** Programación web en SIU-Toba. [En línea] [Citado el: 25 de febrero de 2009.] http://www.7c0h.com.ar/blog/?p=145

6. **Cabanes, José Ignacio.** Diccionario de Informática. [En línea] 4 de febrero de 2007. [Citado el: 16 de abril de 2009.] Disponible en: http://www.nachocabanes.com/diccio/ndic.php

7. **Casanova, Jaime.** Ecualug.org. PostgreSQL, robusto como un elefante. [En línea] [Citado el: 3 de marzo de 2009.] Disponible en: http://www.ecualug.org/?q=2005/11/20/postgresql_robusto_como_un_elefante

8. **Celma, Matilde.** *Almacenes de Datos.* Valencia : s.n., 12/11/2008.

9. **Henrik, Kniberg.** *Scrum y XP desde las trincheras.* 28/02/09. Disponible en: http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf

10. **León, Eduardo.** Visual Paradigm, una herramienta de lo más útil. [En línea] [Citado el: 28 de febrero de 2009.] Disponible en: http://slion2000.blogspot.com/2007/04/visual-paradigm-una-herramienta-de-lo.html

11. **León, Mireydis Orellana.** *LRP Gestión Académica Gestión de Personal 1.0.* UCI : s.n., 2009.

12. **Moraga, Hernán y Guerrero, Aline.** *Programación Ágil: Scrum y XP.* Santiago de Chile : s.n., 2005.

13. **Muñoz, Catherine.** *Requerimientos No Funcionales Akademos 2.0.* UCI : s.n., 2009.

14. **Nader, Javier.** *Tesis de Magister Sistema de Apoyo Gerencial Universitario.* 12/11/2009.

15. **Ortiz, Acralys Ferriol.** *LRP Akademos Resgistro y Control Docente 1.0.* UCI : s.n., 2009.

16. **Palomar, Manuel y Trujillo, Juan Carlos.** *Doctorado Uso y Diseño de Base de Datos Mulitidimensionales y Almacenes de Datos.* España : s.n., 2001-2002.

17. **Portales, Yisel Ávila.** *LRP Akademos Gestión de Carreras 1.0.* UCI : s.n., 2009.

18. **Rob, Peter y Coronel, Carlos.** *Sistemas de bases de datos.* s.l. : Cengage Learning Editores, 2004. Disponible en: http://books.google.com.cu/books?id=B_UVi51RDY4C&printsec=frontcover&dq=base+de+datos+relacionales%2Bventajas#PPP1,M1

19. **Serrano, Manuel.** *Almacenamiento y Recuperación de la Información.* 12/11/2008.

20. **UNNE, Instituto de Ciencias Criminalísticas y Criminología-.** *Sistema de Información Universitaria.* Argentina : s.n., 31/01/09. Disponible en: http://criminalisticas.unne.edu.ar/documentos/siu_introduccion.pdf

21. *De Diseño a Construcción. Patrones.* 20/03/2009. Disponible en: http://profesores.is.escuelaing.edu.co/asignaturas/pob2/cla/sem14aPat.pdf

22. PostgreSQL. *Documentation Chapter 19. Managing Databases 1.9.6 Tablespace.* [En línea] [Citado el: 15 de abril de 2009.] Disponible en:

http://www.postgresql.org/docs/8.2/interactive/manage-ag-tablespaces.html

23. —. TecnoInformacion. [En línea] [Citado el: 25 de febrero de 2009.] Disponible en: http://www.tecnoinformacion.com.ar/siu-toba-soft-libre-uso-Estado.html

24. University Information Services (UIS) Georgetown University. *Data Warehouse: Glossary*. [En línea] [Citado el: 17 de abril de 2009.] Disponible en: http://uis.georgetown.edu/departments/eets/dw/GLOSSARY0816.html

Anexos

Anexo 1. Representación de los diagramas Entidad – Relación pertenecientes a los distintos módulos del área de pre-grado de Akademos v_2.0.

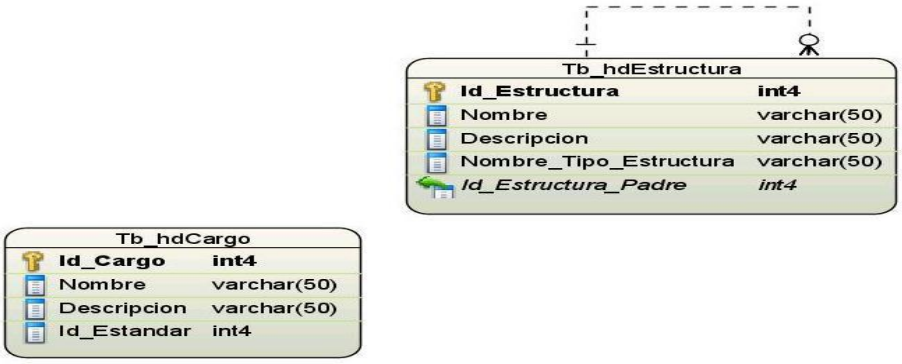

Figura 1 Estructura y Composición

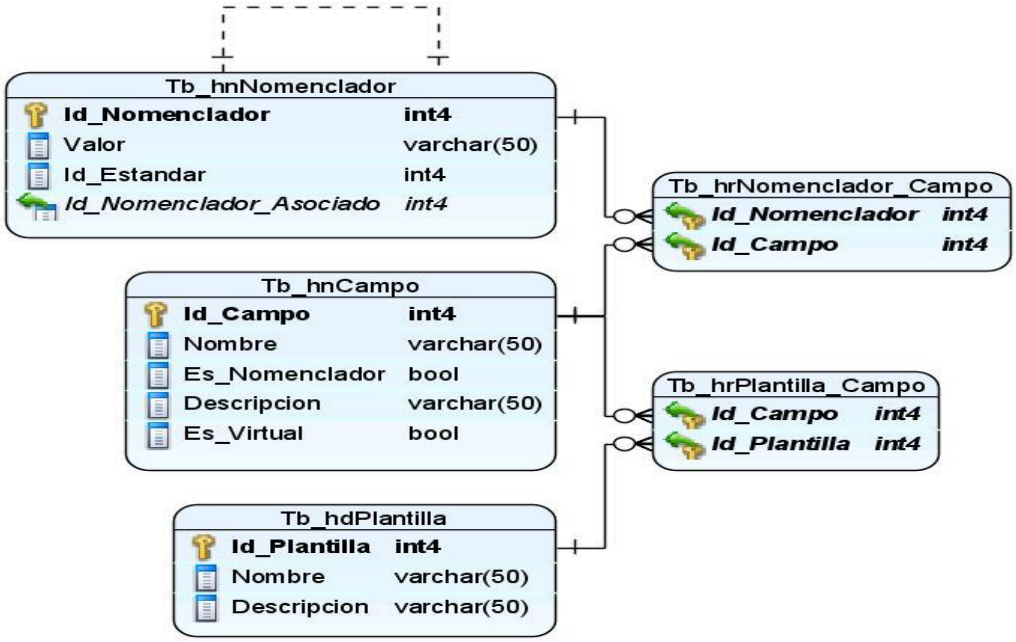

Figura 2 Plantilla

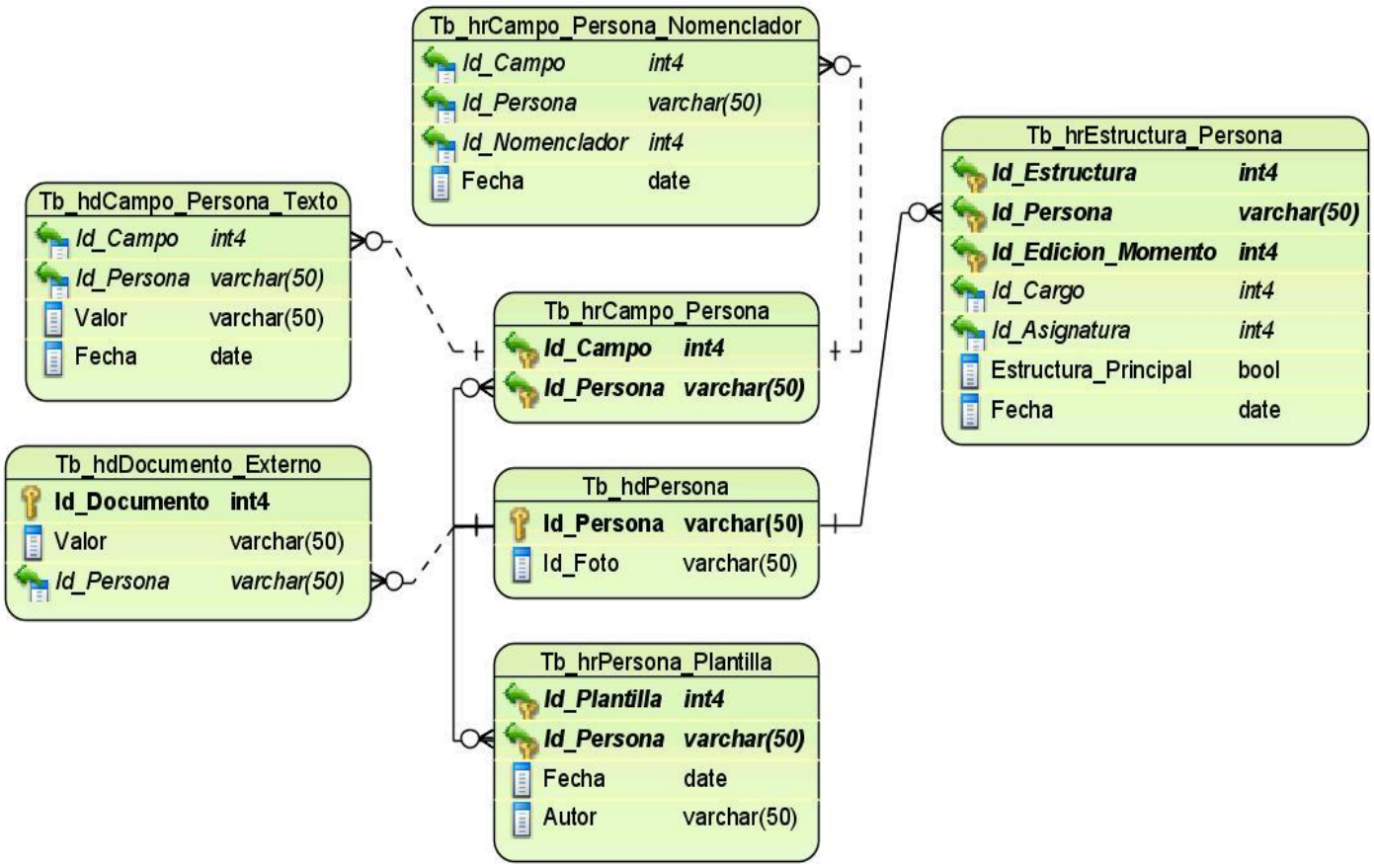

Figura 3 Gestión de Personal

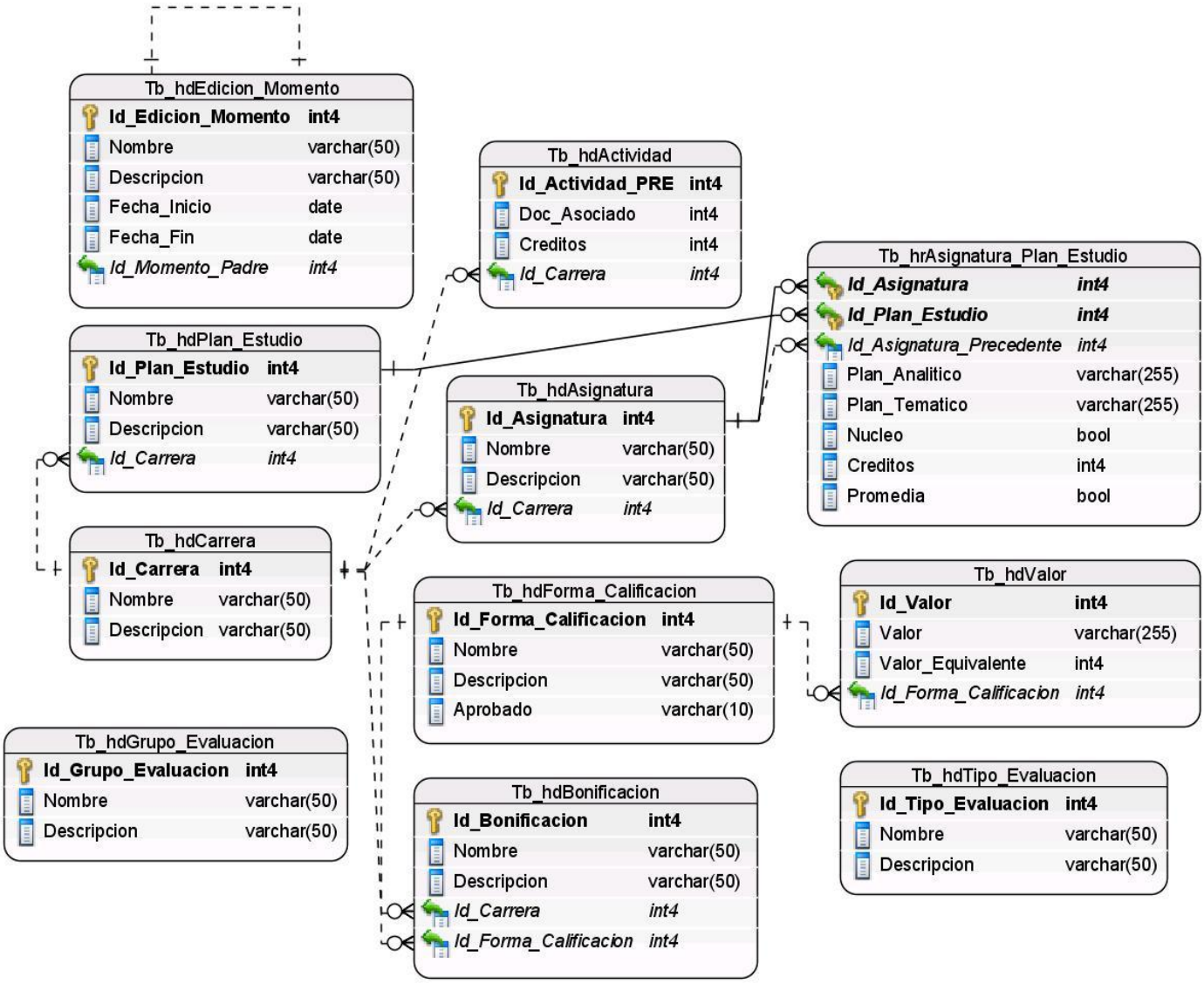

Figura 4 Gestión de Carreras

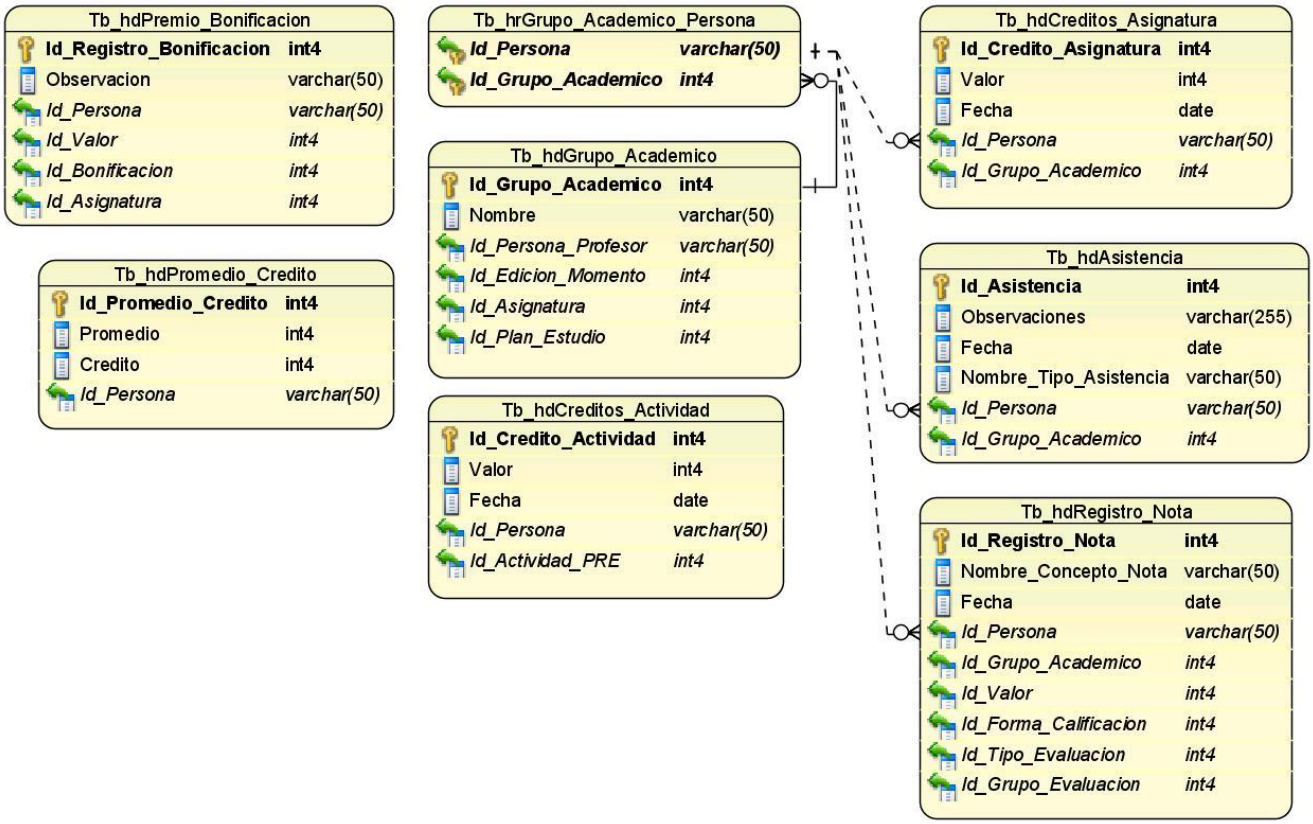

Figura 5 Registro y Control Docente

Anexo 2. Estimación del costo necesario para realizar la investigación.

Materiales Básicos	**Costo CUC**	**Cantidad**	**Costo total**	**Fuente de Financiamiento**
Computadora	500.00	2	1000.00	Universidad
Mesa de Computadora	35.00	2	70.00	Universidad
Silla de Computadora	15.00	2	30.00	Universidad
Memoria Flash	35.00	1	35.00	Medio propio
Materiales de Oficina	**Costo CUC**	**Cantidad**	**Costo total**	**Fuente de Financiamiento**
Lápiz con goma	0.5	10	5.00	Universidad
Lapicero	0.8	5	4.00	Universidad
CD-R y CD-RW	1.00	2	2.00	Universidad
Paquete de Hojas	6.00	1	6.00	Medio Propio
Otros	**Costo CUC**	**Cantidad**	**Costo total**	**Fuente de Financiamiento**
Electricidad	5.00	----	5.00	Universidad
Total			**1157.00**	

Anexo 3. Operacionalización de las variables:

Variable	Indicadores	U/M
Base de Datos Histórica	Eficiencia	Alta
		Media
		Baja
	Adaptabilidad	Alta
		Media
		Baja
Rendimiento del modelo de datos general de sistema	Rapidez	Lento
		Medio
		Rápido

Glosario de Términos

Base de Datos: Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Base de Datos Relacional: Es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos se realizan sobre estas tablas. Estas bases de datos son percibidas por los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo.

BeOS: Es un sistema operativo orientado principalmente a proveer alto rendimiento en aplicaciones multimedia.

Cardinalidad: Describe la dimensión cuantitativa en la relación entre un par de entidades.

Claves foráneas o *foreign key*: Expresan relaciones entre objetos representados, incluyendo en el esquema de una relación atributos de otra. Utilizado para relacionar tablas.

Clúster: El término clúster se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de componentes de hardware comunes y que se comportan como si fuesen una única computadora.

CMM: *Capability Maturity Model* o Modelo de Capacidad y Madurez (CMM) es un modelo de evaluación de los procesos de una organización.

***Data Mining* o Minería de Datos**: Un proceso no trivial de identificación válida, novedosa, potencialmente útil y entendible de patrones comprensibles que se encuentran ocultos en los datos.

DBAL: Acrónimo de *Database Abstraction Layer* (Capa de Abstracción de la Base de Datos). Una capa de abstracción de bases de datos es una interfaz de programación de aplicaciones que unifica la comunicación entre una aplicación informática y bases de datos tales como MySQL, PostgreSQL, Oracle o SQLite. La capa de abstracción de bases de datos permite reducir la cantidad de trabajo proporcionando una API consistente para el desarrollador y ocultado las especificaciones de la base de datos, detrás de la interfaz que provee, tanto como sea posible.

Dependencia funcional: Es una relación entre atributos de una misma relación (tabla). Si *X* e *Y* son atributos de la relación *R*, se dice que *Y* es funcionalmente dependiente de *Y* (se denota por *X*→*Y*) si cada valor de *X* tiene asociado un solo valor de *Y*(*X* e *Y* pueden constar de uno o varios atributos). A *X* se le denomina determinante, ya que *X* determina el valor de *Y*. Se dice que el atributo *Y* es completamente dependiente de *X* si depende funcionalmente de *X* y no depende de ningún subconjunto de *X*.

Dependencia Funcional Transitiva: La dependencia *X*→*Z* es transitiva si existen las dependencias *X*→*Y*, *Y*→*Z*, siendo *X*, *Y*, atributos o conjuntos de atributos de una misma relación. Para eliminar las dependencias transitivas en una relación, 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).

Entidad: Se refiere a cualquier concepto del mundo real con una existencia independiente.

Framework: Un framework denota un conjunto de objetos que definen un diseño abstracto para solucionar un conjunto de problemas relacionados. Puede incluir programas, bibliotecas de objetos o lenguaje interpretado, por lo que su uso facilita la elaboración de sistemas informáticos.

Gestor de base de datos: Es un tipo de software específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Integridad: En términos de datos se refiere a la corrección y completitud de los datos en una base de datos.

ISO 9000: La familia de normas ISO 9000 es un conjunto de normas de calidad establecidas por la Organización Internacional para la Estandarización (ISO), aplicables en cualquier tipo de organización.

Kylix: Herramienta de programación visual para Linux, basada en el lenguaje Pascal (análoga a Delphi), desarrollada por Borland.

Licencia BSD: Es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

Licencia GPL: La Licencia Pública General de GNU o más conocida por su nombre en inglés *GNU General Public License* o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Fundación de Software Libre a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Llave primaria o *primary key*: Conjunto de atributos de su esquema que son elegidos para servir de identificador unívoco de sus tuplas.

Modelo entidad-relación: Es una técnica para el modelado de datos utilizando diagramas de entidad-relación.

Normalización de bases de datos: Consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional. Se usa para evitar la redundancia de los datos, problemas de actualización de los datos en las tablas, proteger la integridad de los datos.

ORM: Acrónimo de *Object-Relational Mapping* (Mapeo Objeto-Relacional). Framework que utiliza técnicas de programación para convertir datos a objetos y viceversa, permitiendo el trabajo con datos persistentes como si formaran parte de una Base de Datos orientada a objetos.

Procedimiento almacenado o *Stored procedure*: Un Procedimiento Almacenado es un programa autocontrolado escrito en lenguaje del SGBD, son almacenados como parte de la base de batos y sus metadatos.

***Querys*:** Consiste en una cadena de consulta, normalmente se utilizan para: insertar, actualizar o editar valores de la base de datos.

Servidor: En informática, un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término servidor ahora también se utiliza para referirse al computador en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos.

SIU-Toba: El SIU-Toba es un entorno de desarrollo Web creado por el SIU con la finalidad de disponer de una herramienta adecuada para la construcción de sistemas transaccionales de mediana y alta complejidad. Entre sus características cabe destacar que dispone de un entorno de edición propio, permite

crear interfaces complejas en forma declarativa, posibilita el manejo de transacciones ocultando problemáticas inherentes a la tecnología Web y facilita el trabajo grupal tanto a nivel local como a distancia. El mismo está basado en un conjunto de herramientas libres ampliamente difundidas: Apache, PHP y PostgreSQL.

Software libre: Software que puede ser distribuido, modificado, redistribuido, copiado y usado libremente. Que un software sea libre no quiere decir que sea gratuito, error que viene de la traducción *Free Software*, pues *free* en inglés significa lo mismo libre que gratuito.

Tabla: En las bases de datos, una tabla significa el tipo de modelado de los datos, donde se guardan los datos recolectados por su programa. Su estructura general se asemeja a la vista general de un programa de hoja de cálculo.

***Tablespace*:** Un *Tablespace* es un lugar de almacenamiento donde los datos reales y esenciales de los objetos de una base de datos pueden ser almacenados. Es la porción física de una base de datos usada para asignarle almacenamiento a todos los segmentos administrados por el SGBD. Un segmento de una base de datos es un objeto de una base de datos que ocupa espacio físico tales como datos de tabla o índices de tabla.

Triggers: Un *trigger* o disparador en una base de datos, es un procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación de inserción (INSERT), actualización (UPDATE) o borrado (DELETE).

Tupla: Es la representación de una fila en una de las tablas que se está almacenando datos. Y las cuales serán llamadas por los administradores de base de datos en el tiempo de ejecución de un sistema.

UNIX: Sistema operativo multiproceso, multiprograma y multiusuario. Software diseñado por AT&T para ingeniería de telecomunicación. Ha sido el primer sistema operativo concebido con independencia de los fabricantes. Posee una gran facilidad para adaptarse a ordenadores con diferentes arquitecturas, siendo ampliamente autónomo respecto del hardware. Está escrito en lenguaje de alto nivel C.

Vista: Es el resultado de una consulta SQL de una o varias tablas, también se le puede considerar una tabla virtual.

Mayo 2009

Universidad de las Ciencias Informáticas