Universidad de las Ciencias Informáticas

Facultad Regional "Mártires de Artemisa"

Título: Diseño de la base de datos e implementación de la capa de acceso a datos para el Módulo Economía y Finanzas de la Dirección Grupo Empresarial de la Administración Provincial de Artemisa.

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

Autor: Yurisleydis Sancruzado Puñales

Tutor: Ing. Aimé Esther Guzmán Ramírez

Co-Tutor: Msc. Lic. Roberto Moreno Ceruto

Artemisa, Junio 2012

La ciencia más útil es aquella cuyo fruto es el más comunicable.

Leonardo Da Vinci

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad Regional ¨Mártires de Artemisa¨ de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Yurisleydis Sancruzado Puñales

_____

Firma de la autora

Aimé Esther Guzmán Ramírez

_____

Firma de la tutora

Agradecimientos

A mi mamá Ruzmerys Puñales Castanedo, que representa en mi vida y en mi carrera la estrella, luz y guía a seguir, ya que gracias a ella y por ella he llegado hasta aquí.

A mi esposo Luis Enrrigue Socarrás Boye que se ha convertido en mi apoyo y mi confianza para seguir adelante, gracias por todo.

A Leosdanis Rodríguez por ser una de las personas que me apoyó para que tomara este camino.

A mi hermanita Yuneydis por estar ahí a pesar de todos los pesares.

A mis tías, Mara, Leydis, y mis primas Liliana, Yendri, Liliet.

A los familiares que me han apoyado en este largo camino, en especial a mi tío Melwin.

A los amigos que de una forma u otra han aportado su granito de arena, dígase Yandi, Yodaysis, Yutmelis.

A todas las muchachitas del cuarto que de cada una de ellas, he aprendido algo, las gracias a ustedes: Lili, Liana, Yadira, Sonia, Yaicel, Martica, Merly, Adilen, Aylen, Guelmis, Maira, Mimi.

A mis dos hermanas, no de sangre, pero sí de corazón, Liliana y Dayli, que me apoyaron mucho y me han dado cariño como se le da a una hermana, las quiero mucho.

A mi tutora, la vicedecana Juana y a mis compañeros de proyecto y de aula.

A todos los profesores a lo largo de estos cinco años, ya que han sido los responsables de llegar hasta aquí hoy.

A todas las personas que de una forma u otra han convertido este sueño en una realidad.

Dedico esta investigación y todo el esfuerzo hecho durante cinco años a mi mamá, que fue la estrellita que alumbró para que luchara y no me dejara vencer jamás, por ella comencé y por ella aún sigo , para que desde donde quiera que esté disfrute de este triunfo, espero que estés muy orgullosa de mí, mami.

A mi futura bebé, pequeño regalo que dios me dio.

A mi esposo por su confianza, apoyo, cariño, por haberse convertido en mi principal sostén, en la familia que tanto necesité en el momento más difícil de mi vida.

A mi hermanita y sobrinito Leandro, por formar parte de ese tesoro valioso que le llamamos familia.

Resumen

Este trabajo surge como una necesidad de la Administración Provincial de Artemisa de automatizar los procesos que se llevan a cabo en las direcciones que la componen. Esta investigación tiene como objetivo desarrollar la capa de acceso a datos y la base de datos para el módulo de Economía y Finanzas de la Dirección de Grupo Empresarial de la Administración Provincial de Artemisa.

En el presente documento se describe detalladamente el diseño de una base de datos que garantiza la manipulación de la información en esta dirección. Se detalla también la implementación de la capa de acceso a datos para lograr una eficaz conexión a la bases de datos. Con el fin de cumplimentar los objetivos propuestos fue realizada una investigación acerca de las tecnologías existentes relacionadas con el acceso a los datos en base de datos a nivel mundial. Finalmente, la solución fue sometida a un proceso de validación teórica y funcional.

Para el desarrollo de la solución se utilizaron las capacidades de los Frameworks Hibernate y Spring y el Entorno de Desarrollo Integrado NetBeans 7.0.1.

Palabras claves: Base de Datos; Capa de Acceso a Datos; Frameworks.

Índice

Contenido

Introducción

A lo largo de la historia la tecnología ha constituido la invención de técnicas y herramientas con un propósito práctico. La historia moderna está relacionada íntimamente con la historia de la ciencia, pues el descubrimiento de nuevos sucesos ha permitido la adquisición de mayores conocimientos y, recíprocamente, se han podido realizar más hallazgos científicos gracias al desarrollo de nuevas tecnologías, que han extendido las posibilidades de experimentación.

Las Tecnologías de la Información y las Comunicaciones (TIC), han tenido un desarrollo acelerado, estas tecnologías constituyen un apoyo fundamental para la gestión del conocimiento, ayudando a la consolidación de la sociedad de la información,

La sociedad de la información, es el resultado de una revolución tecnológica sin precedentes, estimulada por el progreso de las telecomunicaciones, las tecnologías audiovisuales y la informática, que producen profundas transformaciones que amplían y posibilitan procesos sociales, políticos, económicos y culturales de la vida de las personas en todo el mundo.

El campo de la informática se ha desarrollado de forma creciente, por lo que hoy en día las sociedades, cuentan con sistemas informáticos que permiten almacenar y procesar información. La base de todo sistema informático es el manejo de gran cantidad de información de una forma ordenada y rápida. Existen sistemas informáticos que permiten el almacenamiento, la gestión y el análisis de los datos.

El almacenamiento de información se realiza mediante aplicaciones que permiten mantener eficientemente grandes volúmenes de datos permitiendo administrar y supervisar los recursos accediendo a toda la información de forma confiable, precisa y oportuna, lo que contribuye a solucionar dificultades optimizando los procesos.

En Cuba existen diferentes sistemas informáticos que permiten el almacenamiento de información, estos son realizados en diferentes instituciones y centros de estudios, unos de estos centros es la Universidad de Ciencias Informáticas (UCI) en la que se desarrolla disimiles proyectos para entidades de todo el país, esta universidad cuenta con varias facultades dentro de las cuales se encuentra la Facultad Regional UCI "Mártires de Artemisa" (FRAUCI), que se encuentra ubicada en el occidente en la naciente provincia Artemisa.

Dicha provincia nace con la nueva división política-administrativa. Artemisa fue aprobada como Provincia en agosto del 2010 por la Asamblea Nacional del Poder Popular y cuyo funcionamiento entró en vigor el 1 de Enero del 2011. En el municipio cabecera de la misma, se encuentra ubicada la Administración Provincial, el cual está dividido en 32 direcciones provinciales entre las cuales se encuentra la Dirección Grupo Empresarial que a su vez consta de varios grupos, entre ellos el Grupo de Economía y Finanzas.

Este grupo se encarga de gestionar toda la información referente a la Empresa de Servicios Comunales donde se recoge la información de las flores en cuanto a su total en existencia, la cantidad producida y comercializada; la cantidad de sarcófagos disponibles y de carros fúnebres funcionando o no. Atiende la información de la Empresa de Transporte que tiene que ver con la transportación de pasajeros en medios de transporte propio, arrendado y mediante porteadores privados en pesos cubanos y la cantidad de parqueo de vehículos disponibles por municipio.

Gestiona, además, la información de la Empresa Minorista de Comercio y Gastronomía donde se tienen los partes de venta para el control de los productos agrícolas, de la producción de cemento a granel y en bolsas y de los productos líderes: materiales de construcción, productos de aseo, insumos agrícolas y productos del mercado ideal.

Dicho grupo registra también la información de la Empresa de Construcción y Mantenimiento relacionada con la venta liberada de materiales para la construcción, a la población y a organismos; el estado de la construcción civil y montaje de nuevas obras, edificaciones e instalaciones, de demolición, montaje, remodelación, reconstrucción y/o rehabilitación de edificaciones, instalaciones y otros objetivos existentes, así como de reparación y mantenimiento constructivo a las entidades pertenecientes al sistema del Poder Popular del territorio artemiseño.

En el caso de la Empresa de Producciones Alimentarias y Artículos Varios la tarea del grupo, consiste en tener un control de las panaderías en los diferentes municipios de la provincia. La Empresa Farmacias y Ópticas analizan todo lo referente a los productos químicos dispensariales y de fuentes naturales que son producidos en la provincia por semana y los medicamentos que están en falta así como las causas que provocan dicho faltante.

El proceso de almacenamiento de toda esta información se realiza de forma manual, en formato duro o digital, lo que trae consigo demora en la tramitación de los datos, provocando que en ocasiones no se tenga constancia real del cumplimiento de la planificación de la producción de medicamentos que se producen en la provincia y de la relación verdadera de la causa de la falta de medicamentos en la provincia, así como su pérdida y duplicado. Los reportes estadísticos que se generan son poco fiables por lo difícil que resulta recopilar un gran volumen de información al transcurrir largos períodos de tiempo. Todas estas dificultades ocasionan problemáticas para la generación de reportes inmediatos y de la búsqueda de los datos referente a las producciones de flores y de los sarcófagos funcionando o no, así como su entrega, en el tiempo determinado y la calidad requerida. La entrega incorrecta y tardía de esta información afecta la toma de decisiones para la Provincia Artemisa con respecto a la economía y las finanzas. Todo lo expuesto anteriormente engloba la **situación problemática.**

Por todas las situaciones anteriormente expuestas se arriba al siguiente **problema de investigación:** ¿Cómo contribuir a la persistencia y recuperación de la información en el módulo de Economía y Finanzas de la Dirección Grupo Empresarial de la Administración Provincial de Artemisa?

A partir del problema a resolver se define como **objeto de estudio**: procesos de gestión de la información en base de datos relacionales, enmarcado en el **campo de acción:** persistencia y recuperación de los datos relacionales en el sector de la Economía y las Finanzas.

Para dar solución al problema anteriormente expuesto se traza el siguiente **objetivo general**: Desarrollar la base de datos y la capa de acceso a datos para el módulo de Economía y Finanzas de la Dirección Grupo Empresarial de la Administración Provincial de Artemisa.

Para cumplir con el objetivo propuesto se definen los siguientes **objetivos específicos** de la investigación:

- Elaborar la Fundamentación Teórica de la Investigación.
- Diseñar la capa de acceso a datos y la base de datos para el módulo de Economía y Finanzas de la Dirección Grupo Empresarial.
- Implementar la capa de acceso a datos para módulo de Economía y Finanzas de la Dirección Grupo Empresarial
- Validar la propuesta de solución.

Para guiar la presente investigación se formula la siguiente **idea a defender:** Con el desarrollo de la base de datos y la capa de acceso a datos para el módulo de Economía y Finanzas de la Dirección Grupo Empresarial de la Administración Provincial de Artemisa, se contribuye a la persistencia y recuperación de los datos en el módulo.

Del análisis del objetivo general se derivan las siguientes **tareas de la investigación:**

- Definición del Marco teórico de la investigación.
- Elaboración del estado del arte de la investigación.
- Definición de la metodología a utilizar para el desarrollo de la solución propuesta.
- Especificación de las herramientas y tecnologías a utilizar para el desarrollo de la solución propuesta.
- Diseño de la capa de acceso a datos y la base de datos para módulo de Economía y Finanzas de la Dirección Grupo Empresarial.
- Implementación de la capa de acceso a datos para módulo de Economía y Finanzas de la Dirección Grupo Empresarial.
- Realización de pruebas a la capa de acceso del sistema de gestión.

La investigación se apoya en los métodos teóricos y métodos empíricos:

Métodos teóricos

- **Análisis histórico-lógico:** permitió analizar las características de las bases de datos relacionales, los sistemas gestores, las herramientas necesarias para la implementación de una capa de acceso y realizar un estudio histórico de los mismos, lo que permitió valorar y determinar lo más factible a usar para la presente investigación.
- **Inductivo-deductivo:** a partir del planteamiento de la idea a defender y siguiendo la lógica de deducción se llega a nuevos conocimientos y predicciones que son sometidos a verificaciones. Se deduce que con el desarrollo de la base de datos y la capa de acceso a datos para el Módulo

Economía y Finanzas de la Dirección Grupo Empresarial de la Administración Provincial de Artemisa, se contribuye a la persistencia y recuperación de la información en el área que comprende dicho módulo.

- **Analítico-Sintético:** Se analiza y resume la documentación recopilada, a través de los diferentes medios bibliográficos, para de esa forma desarrollar un mejor diseño e implementación del sistema que se propone.
- **Modelación:** Se modela el problema planteado creando el modelo de datos para la confección de la bases de datos del módulo de Economía y Finanzas de la Dirección Grupo Empresarial.

Métodos empíricos

- **Análisis documental:** Permitió el estudio y análisis de todos los documentos e informes del módulo de Economía y Finanzas, para obtener un mejor entendimiento de los procesos de la misma.

Con el correcto cumplimiento de las tareas se espera obtener como **aportes prácticos:**

- Capa de acceso a datos que contribuya a la persistencia y recuperación de los datos en el módulo de Economía y Finanzas.

El presente trabajo está estructurado en tres capítulos los cuales contienen la siguiente información:

Capítulo 1: Fundamentación teórica: En este capítulo se expone la fundamentación teórica que da sustento al trabajo de diploma. Se brinda información sobre las definiciones de las Bases de Datos (BD) y los Sistemas Gestores de Bases de Datos (SGBD), los tipos de arquitectura y modelos de BD,

así como las herramientas, gestores y tecnologías empleados para la realización de la propuesta.

Capítulo 2: Diseño y arquitectura de la base de datos: En este capítulo se explica la arquitectura a utilizar para la realización de la base de datos, se definen los requisitos funcionales del sistema, se expone la nomenclatura. Se representa el diagrama de clases persistentes y se diseña la BD.

Capítulo 3: Implementación de la capa de acceso y validación de la propuesta de solución: En este capítulo se detalla cómo se realiza la implementación de la capa de acceso de a datos, con esta una descripción del mapeo de las clases, así como la descripción de los métodos de la clase GenericDao. También se realiza la validación de los métodos implementados para la persistencia de objetos.

Capítulo 1. Fundamentación Teórica

En este capítulo se referencia la fundamentación teórica de la presente investigación, donde se realiza un estudio tanto conceptual, así como del estado del arte. Su objetivo principal es indagar y detectar los diferentes sistemas y aplicaciones que tengan similitudes a las del sistema que se quiere lograr.

1.1 Fundamentos teóricos de las Bases de Datos

Las bases de datos tuvieron sus orígenes en la Antigüedad donde se contaba con la existencia de bibliotecas y todo tipo de registros. Seguidamente, las bases de datos se desarrollaron partiendo de la necesidad de recolectar enormes conjuntos de información o datos. Desde la llegada de las primeras computadoras, el concepto de bases de datos ha estado estrechamente ligado a la informática. La terminología acerca de las bases de datos fue escuchado por vez primera en California en el año 1963. Por estos años se le dio comienzo a las primeras generaciones de bases de datos de red y las bases de datos jerárquicas, ya que estas brindaban la posibilidad de almacenar estructuras de datos en listas y árboles. Con la definición del modelo relacional y las reglas para los sistemas de datos relacionados se dio paso a la segunda generación de los Sistemas Gestores de Bases de Datos.

Después de la época de los años ochenta es desarrollado el *Structured Query Language* (SQL) o lenguaje de consultas o lenguaje declarativo de acceso a bases de datos relacionales que admite efectuar consultas con la finalidad de recuperar información de una base de datos y efectuar transformaciones sobre la base de datos de forma simple; también analizan grandes volúmenes de información y permiten precisar numerosos tipos de operaciones frente a la misma información.

En los años 1990 las investigaciones acerca de las bases de datos cambió el rumbo, comenzándose a investigar acerca de bases de datos orientadas a objetos,

a las que se les comenzó a añadir novedosas expresiones regulares, consultas recursivas, *triggers* entre otras características orientadas a objetos, sufriendo así disímiles modificaciones hasta dar las posibilidades con que cuenta hoy en día.

1.1.1 Conceptos Asociados al dominio del problema

A continuación se describe algunos conceptos asociados al dominio del problema, para un mejor entendimiento del mismo. Para la selección de los conceptos asociados se realiza un estudio de estos y se selecciona el que se ajuste a la investigación.

Sistema Gestor de Base de Datos:

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. (Alvarez, 2007)

Un sistema de Gestión de Bases de datos es un tipo de software muy especifico dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan; o lo que es lo mismo, una agrupación de programas que sirven para definir, construir y manipular una base de datos, permitiendo así almacenar y posteriormente acceder a los datos de forma rápida y estructurada. (Midepa, 2011). Se asume este concepto de SGBD, ya que se considera el más completo y adecuado en la presente investigación.

Capa de acceso a datos:

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. (Panitsch, 2008)

Capa que sirve entre como puente entre la capa lógica de negocio y el proveedor de datos. Este capa pretende encapsular las especificidades del proveedor de datos tales como (SQL, Oracle, Sybase, archivos XML, texto, hojas electrónicas), a la siguiente capa. Para que si cambia el proveedor de datos solo necesitemos cambiar en una sola capa, el proveedor de datos. (Herrera, 2007). Se asume este concepto de capa de acceso a datos, ya que se considera el más completo y adecuado en la presente investigación.

Bases de datos relacionales:

Es una colección de archivos interrelacionados, son creados con un *Database Management System* (DBMS). El contenido de una base de datos engloba a la información concerniente (almacenadas en archivos) de una organización, de tal manera que los datos estén disponibles para los usuarios, una finalidad de la base de datos es eliminar la redundancia o al menos minimizarla. (Oropeza, 2007)

Una 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 operan sobre estas tablas. (Miranda, 2007)**.** Se asume este concepto de base de datos relacional, ya que se considera el más completo y adecuado en la presente investigación.

1.1.2 Estado del arte de las bases de datos

Las bases de datos son muy utilizadas a nivel mundial, por la facilidad que brinda de almacenar grandes volúmenes de información que nos puede servir en posteriores tomas de decisiones. En todas las esferas de la sociedad son utilizadas estas ya sea en la educación, en la economía, la salud, los deportes. Para la elaboración del estado del arte, se realizó un estudio de las bases de datos para el sector de la economía y las finanzas, a continuación se exponen algunos ejemplos en Cuba y el mundo.

Ejemplos a nivel mundial

ITCS International Trade by Commodity: Base de datos en línea, producida por la OCDE*(*Organización para la Cooperación y el Desarrollo Económicos*),* que constituye una fuente fiable de datos estadísticos anuales sobre importación y exportación en los países de la OCDE, y países fuera de la OCDE seleccionados. (Buc, 2005)

ESTACOM: Base de datos en línea, producida por ICEX (Instituto Español de Comercio Exterior), de estadísticas comerciales. (Buc, 2005)

Informes: España. **Ministerio de Hacienda**. **Informes y dictámenes** relacionados con el control de la actividad económico - financiera del sector público, emitidos fundamentalmente por la IGAE (Intervención General de la Administración del Estado), desde 1984. (Vitoria, 2009)

Estadísticas Banco de España: Boletín estadístico de la economía española. Contiene: indicadores económicos, créditos, depósitos, detalles por sectores económicos, provincias y Comunidades Autónomas, cuentas financieras, Balanza de Pagos y PII, listas de activos elegibles, etc. (Vitoria, 2009)

Las cuentas nacionales (incluido el PIB): En esta sección se ofrece acceso a una selección de información de antecedentes metodológicos y de otra índole en relación con las cuentas nacionales básicas, tales como los agregados del producto interno bruto (PIB) y sus principales componentes. (eurostat, 2011)

SEC 95 Suministro, uso y tablas Input-Output: Tablas de suministro, uso y entrada-salida proporcionan información detallada para un año determinado en las actividades de producción, la oferta y la demanda de bienes y servicios, los consumos intermedios, los insumos primarios y el comercio exterior. (eurostat, 2011)

Las cuentas del sector europeo del grupo en conjunto los temas económicos con un comportamiento similar en los sectores institucionales, tales como: hogares, las sociedades no financieras, las sociedades financieras y el gobierno.

Agrupación de temas económicos de esta manera en gran medida ayuda a comprender el funcionamiento de la economía. (eurostat, 2011)

Estadísticas Financieras Monetarias y Otros: Estas estadísticas cubren muchos de los elementos necesarios para la comprensión de la evolución monetaria y financiera: los agregados monetarios, activos y pasivos externos (incluyendo las reservas internacionales oficiales), archivo y la información sobre el mercado de bonos, las transacciones bancarias. (eurostat, 2011)

Inflación: Los Índice de Precios de Consumo Armonizado (IPCA) son los indicadores económicos construidos para medir los cambios en el tiempo en los precios de los bienes y servicios de consumo adquiridos por los hogares. Los IPCA dar medidas comparables de la inflación en la zona del euro, la Unión Europea (UE), del Espacio Económico Europeo y de otros países, incluidos los países adherentes y candidatos. Se calculan de acuerdo a un enfoque armonizado y un único conjunto de definiciones. Ellos proporcionan la medida oficial de la inflación de precios al consumidor en la zona del euro a los efectos de la política monetaria en la zona del euro y la evaluación de convergencia de la inflación según lo dispuesto en los criterios de Maastricht. (eurostat, 2011)

Ejemplos en Cuba

Base de Datos del Banco Central de Cuba: Base de Datos contratada con ELTON B. STEPHENS COMPANY (EBSCO)**,** es un líder mundial proporcionando acceso a información y soluciones de dirección a través de la impresión y servicios de subscripción al periódico electrónico, investigación de desarrollo y producción de

bases de datos, acceso online a más de 100 bases de datos y miles de periódicos electrónicos, y procuración de libros de comercio electrónico.

EBSCO ha especializado productos y servicios para las bibliotecas académicas, médicas, gubernamentales, públicas y escolares así como para corporaciones y otras organizaciones. EBSCO mantiene una base de datos comprensiva de más de 282,000 títulos de serie y mantiene relaciones con más de 60,000 publicadores mundiales. (Banco Central de Cuba, 2009)

Base de Datos de pasajeros: una minúscula proporción de los pasajeros arribados a Cuba**.** (CubaGenWeb.org, 1996-2010)

Base de datos de la aduana: encuentran el registro y codificación de las personas naturales y jurídicas vinculadas al despacho de mercancías y medios de transportes internacionales. Realiza el registro e inscripción de los tripulantes cubanos de naves y aeronaves. (Aduana - CUBA - Customs, 2012)

1.2 Bases de datos

A continuación se describen algunas definiciones de bases de datos:

En una primera aproximación, se puede decir que una base de datos es un conjunto de información relacionada que se encuentra agrupada o estructurada. Desde el punto de vista informático, una 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. (Midepa, 2011)

Se define una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular. (Valdés, 2007)

Características de las bases de datos

Entre las principales características de los sistemas de base de datos podemos mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

1.2.1 Clasificación de las bases de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, o la utilidad de la misma. Según la variabilidad de los datos almacenados:

Bases de datos estáticas

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

Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc. (Gómez, 2008)

1.2.2 Modelos de Bases de Datos

El modelo de datos es el lenguaje con el que se describe una base de datos, este permite describir los elementos que intervienen en un problema y relación de estos elementos entre sí. Existen disímiles modelos los cuales se nombran a continuación:

- ✓ Modelo Jerárquico
- ✓ Modelo de Red
- ✓ Modelo Relacional
- ✓ Modelo Orientados a Objetos
- ✓ Modelo Declarativo.
- ✓ Modelo Bases de Datos Documentales.
- ✓ Modelo de Bases de Datos Distribuida.

Entre los modelos más utilizados en la actualidad se encuentran:

Modelo de datos jerárquico

Este modelo utiliza árboles para la representación lógica de los datos. Este árbol está compuesto de unos elementos llamados nodos. El nivel más alto del árbol se denomina raíz. Cada nodo representa un registro con sus correspondientes campos.

La representación gráfica de este modelo se realiza mediante la creación de un árbol invertido, los diferentes niveles quedan unidos mediante relaciones.

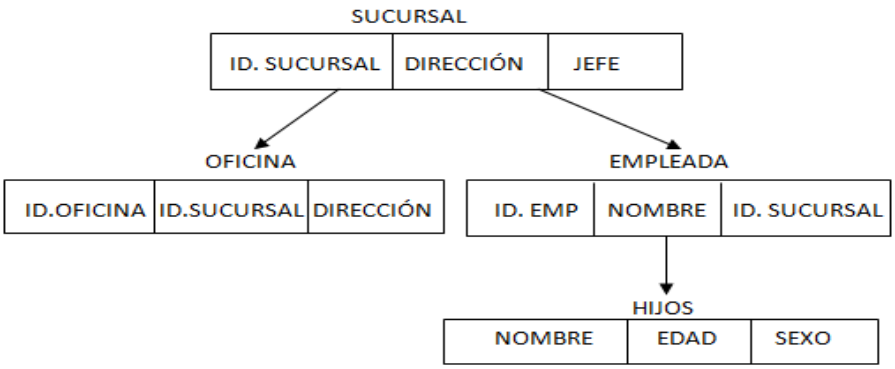


Figura.1 Modelo de Datos Jerárquico

En este modelo solo se pueden representar relaciones 1: M, por lo que presenta varios inconvenientes:

- No se admiten relaciones N:M
- Un segmento hijo no puede tener más de un padre.
- No se permiten más de una relación entre dos segmentos.
- Para acceder a cualquier segmento es necesario comenzar por el segmento raíz
- El árbol se debe de recorrer en el orden designado.

Modelo de datos en red

En este modelo las entidades se representan como nodos y sus relaciones son las líneas que los unen. En esta estructura cualquier componente puede relacionarse con cualquier otro.

A diferencia del modelo jerárquico, en este modelo, un hijo puede tener varios padres.

Los conceptos básicos en el modelo en red son:

- El tipo de registro, que representa un nodo.
- Elemento, que es un campo de datos.
- Agregado de datos, que define un conjunto de datos con nombre.

Este modelo de datos permite representar relaciones N:M

Modelo de datos relacional

Este modelo es el más utilizado actualmente ya que utiliza tablas bidimensionales para la representación lógica de los datos y sus relaciones.

Algunas de sus principales características son:

- Puede ser entendido y usado por cualquier usuario.

- Permite ampliar el esquema conceptual sin modificar las aplicaciones de gestión.
- Los usuarios no necesitan saber donde se encuentran los datos físicamente. (Alvarez, 2007)

1.2.3 Etapas del diseño de bases de datos

El diseño de una base de datos no es un proceso sencillo. La complejidad de la información y la cantidad de requisitos de los sistemas de información hacen que sea complicado. Por lo que resulta conveniente descomponer el proceso del diseño en varias etapas:

1) **Etapa del diseño conceptual:** en esta etapa se obtiene una estructura de la información de la futura BD independiente de la tecnología que hay que emplear. No se tiene en cuenta todavía qué tipo de base de datos se utilizará –relacional, orientada a objetos, jerárquica, etc.–; en consecuencia, tampoco se tiene en cuenta con qué SGBD ni con qué lenguaje concreto se implementará la base de datos. Así pues, la etapa del diseño conceptual nos permite concentrarnos únicamente en la problemática de la estructuración de la información, sin tener que preocuparnos al mismo tiempo de resolver cuestiones tecnológicas.

2) **Etapa del diseño lógico**: en esta etapa se parte del resultado del diseño conceptual, que se transforma de forma que se adapte a la tecnología que se debe emplear. Más concretamente, es preciso que se ajuste al modelo del SGBD con el que se desea implementar la base de datos. Por ejemplo, si se trata de un SGBD relacional, esta etapa obtendrá un conjunto de relaciones con sus atributos, claves primarias y claves foráneas.

3) **Etapa del diseño físico**: en esta etapa se transforma la estructura obtenida en la etapa del diseño lógico, con el objetivo de conseguir una mayor eficiencia;

además, se completa con aspectos de implementación física que dependerán del SGBD.

Por ejemplo, si se trata de una base de datos relacional, la transformación de la estructura puede consistir en lo siguiente: tener almacenada alguna relación que sea la combinación de varias relaciones que se han obtenido en la etapa del diseño lógico, partir una relación en varias, añadir algún atributo calculable a una relación, etc. Los aspectos de implementación física que hay que completar consisten normalmente en la elección de estructuras físicas de implementación de las relaciones, la selección del tamaño de las memorias intermedias (buffers) o de las páginas, etc.

En la etapa del diseño físico –con el objetivo de conseguir un buen rendimiento de la base de datos–, se deben tener en cuenta las características de los procesos que consultan y actualizan la base de datos, como por ejemplo los caminos de acceso que utilizan y las frecuencias de ejecución. También es necesario considerar los volúmenes que se espera tener de los diferentes datos que se quieren almacenar. (Dataprix, 2007)

1.2.4 Metodología para el diseño de bases de datos

La metodología para el diseño de la BD, consta de los siguientes pasos:

- Determinación de entidades y atributos.
- Normalización de entidades.
- Determinación de relaciones.
- Obtención del modelo lógico global de los datos.
- Diseño físico de la BD.

Cuando se realiza el diseño de la BD para un sistema determinado, es necesario establecer los datos a tomar en cuenta y las dependencias funcionales existentes entre ellos.

Rigurosamente, esto se obtiene luego de realizada la etapa de análisis del sistema y partiendo de lo obtenido en esta. (Mato Garcia, 2004)

1.2.5 Gestión de información de una base de datos.

Uno de los aspectos de mayor importancia en una base de datos es la gestión de información. Este aspecto abarca diferentes actividades como el almacenamiento, la recuperación, la actualización y la eliminación de los datos. Es posible realizar estas acciones mediante las facilidades que brindan los Sistemas Gestores de Base de Datos. Estos sistemas poseen lenguajes especiales llamados sublenguajes de datos (DSL, *Data Sublanguage*) que permiten manipular la información.

Los sublenguajes de datos se emplean para tratar los objetos de la BD y sus operaciones. Están compuestos por al menos dos lenguajes subordinados: "un lenguaje de definición de datos (DDL, *Data Definition Language*), el cual garantiza la definición o descripción de los objetos de la BD, y un lenguaje de manipulación de datos (DML, *Data Manipulation Language*), que garantiza la manipulación o tratamiento de esos objetos" (Mato, 1999).

Algunos gestores incluyen un lenguaje de control de los datos (DCL, *Data Control Language*) que permite controlar los permisos de los objetos de la BD.

Los sublenguajes de datos son empleados por las aplicaciones informáticas para manejar la información almacenada, llevando a cabo tareas específicas. Con este fin se implementan un conjunto de clases y funciones que conforman lo que se conoce como capa de acceso a los datos. Esta capa es la encargada de proveer a los programas de aplicación el acceso a los datos almacenados de forma persistente.

El empleo de distintas tecnologías de BD y las posibles sustituciones de una tecnología por otra, hacen necesario que las soluciones informáticas brinden soporte para distintos tipos de BD. Por estas razones se recomienda la

implementación de una capa de abstracción que permita manipular la información con independencia del tipo de BD utilizada. La capa de abstracción de BD es una Interfaz de Programación de Aplicaciones (API, *Applications Programming Interface*) que permite la comunicación entre una aplicación informática y diferentes SGBD, abstrayendo las diferencias existentes entre estos sistemas (A. Otto, 2004)

1.3 Tipos de Arquitectura de los SGBD

1.3.1 Arquitectura Cliente/Servidor

La computación cliente/servidor es la extensión lógica de la programación modular. El supuesto principal de la programación modular es la división de un programa grande en pequeños programas (llamados módulos), siendo más fáciles el desarrollo y la mantenibilidad (divide y vencerás).

Los sistemas cliente/servidor no están limitados a aplicaciones de bases de datos. Cualquier aplicación que tenga una interfaz de usuario (*front-end*, sección frontal o parte cliente) que se ejecute localmente en el cliente y un proceso que se ejecute en el servidor (*back-end*, sección posterior, o sistema subyacente) está en forma de computación cliente/servidor.

Un sistema cliente/servidor es aquel en el que uno o más clientes y uno o más servidores, conjuntamente con un sistema operativo subyacente y un sistema de comunicación entre procesos, forma un sistema compuesto que permite cómputo distribuido, análisis, y presentación de los datos.

Si existen múltiples servidores de procesamiento de base de datos, cada uno de ellos deberá procesar una base de datos distinta, para que el sistema sea considerado un sistema cliente/servidor. Cuando dos servidores procesan la misma base de datos, el sistema ya no se llama un sistema cliente/servidor, sino que se trata de un sistema de base de datos distribuido.

Los clientes, a través de la red, pueden realizar consultas al servidor. El servidor tiene el control sobre los datos; sin embargo los clientes pueden tener datos privados que residen en sus computadoras. Las principales características de la arquitectura cliente/servidor son:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Tipos de arquitecturas cliente/servidor.

Arquitectura de 2 capas: La arquitectura cliente/servidor tradicional es una solución de 2 capas. La arquitectura de 2 capas consta de tres componentes distribuidos en dos capas: cliente (solicitante de servicios) y servidor (proveedor de servicios). Los tres componentes son:

- Interfaz de usuario.
- Gestión del procesamiento.
- Gestión de la base de datos.

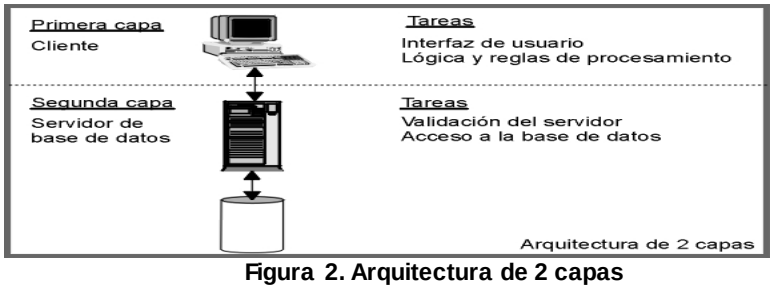


Figura 2. Arquitectura de 2 capas

Arquitectura de 3 capas: La arquitectura de 3 capas surgió para superar las limitaciones de la arquitectura de 2 capas. La tercera capa (servidor intermedio) está entre el interfaz de usuario (cliente) y el gestor de datos (servidor). La capa intermedia proporciona gestión del procesamiento y en ella se ejecutan las reglas y lógica de procesamiento.

Permite cientos de usuarios (en comparación con sólo 100 usuarios de la arquitectura de 2 capas). La arquitectura de 3 capas es usada cuando se necesita un diseño cliente/servidor que proporcione, en comparación con la arquitectura de 2 capas, incrementar el rendimiento, flexibilidad, mantenibilidad, reusabilidad y escalabilidad mientras se esconde la complejidad del procesamiento distribuido al usuario.

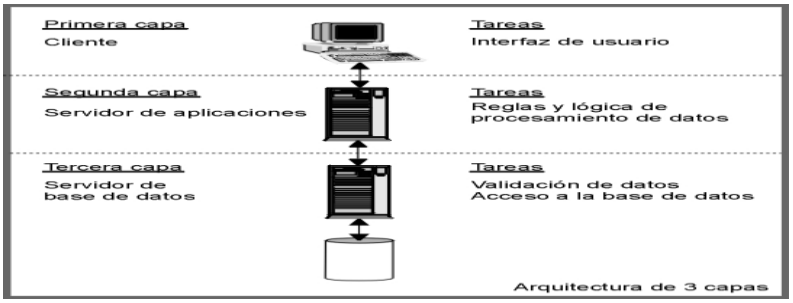


Figura 3. Arquitectura de 3 capas

1.3.2 Arquitectura distribuida

Un sistema de computación distribuida consiste en un conjunto de computadores (que no necesariamente tienen que ser homogéneos), que están interconectados entre sí formando una red, y que cooperan para realizar una determinada tarea. Un sistema de computación distribuida parte un problema grande en pequeñas piezas, y soluciona cada una de ellas eficientemente de una manera coordinada.

En la arquitectura distribuida el SGBD y la BD no están asociados a un determinado ordenador, sino a una red cuyos nodos se reparten las funciones. Una base de datos distribuida es vista por las aplicaciones igual que si fuera

centralizada. Es el SGBDD el que se encarga de preservar la integridad y coherencia de la BD.

Un SGBD que soporte una arquitectura de base de datos distribuida es mucho más complejo que uno para base de datos centralizada y el número de SGBDD disponibles en el mercado es mucho menor. Existen algunos SGBDs que ofrecen la posibilidad de implementar una BD distribuida sólo para sistemas homogéneos. Es una tecnología que no está tan probada como la centralizada.

Los usuarios acceden a la base de datos distribuida a través de aplicaciones. Estas aplicaciones se pueden clasificar en aquellas que no requieren datos de otros computadores (aplicaciones locales) y aquellos que requieren datos de otros computadores (aplicaciones globales). Un SGBDD tiene las siguientes características:

Una colección de datos compartidos y relacionados lógicamente. Los datos están divididos en fragmentos. Los fragmentos se pueden duplicar. Los fragmentos se colocan en varios emplazamientos (computadores). Dichos emplazamientos están conectados por una red. Los datos de cada emplazamiento están bajo el control de un SGBD. El SGBD en cada emplazamiento puede manejar aplicaciones locales autónomamente. Cada SGBD participa en al menos una aplicación global.

Procesamiento distribuido.

Es importante hacer una distinción entre SGBD distribuido y procesamiento distribuido. Procesamiento distribuido: Una base de datos centralizada que puede ser accedida a través de una red. El punto clave de la definición de una base de datos distribuida es que el sistema está formado por datos que están físicamente distribuidos a través de una red. Si los datos están centralizados, incluso aunque los usuarios puedan acceder a los datos a través de la red, no se puede considerar

como un sistema distribuido. La topología de un sistema de procesamiento distribuido se muestra en la siguiente figura:

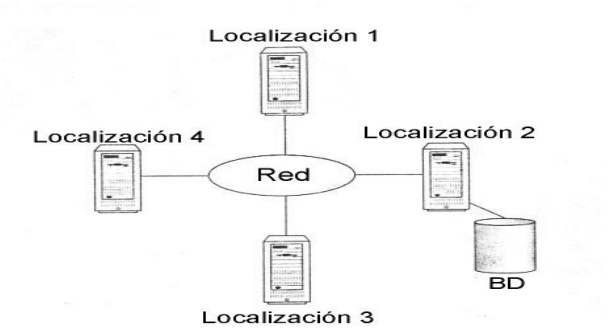


Figura 4. Sistema de procesamiento distribuido

Arquitectura de referencia para un SGBDD.

En la siguiente figura muestra una arquitectura de referencia para un SGBDD conforme a la arquitectura ANSI-SPARC:

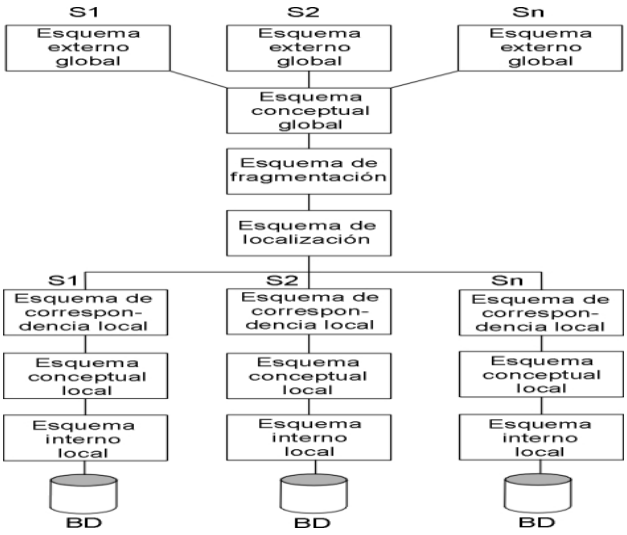


Figura 5 Arquitectura de referencia para un SGBDD

Esquema conceptual global: El esquema conceptual global es la descripción lógica de la base de datos completa, como si no estuviera distribuida. Este nivel

corresponde al nivel conceptual de la arquitectura ANSI-SPARC y contiene definiciones de entidades, relaciones, constantes e información sobre seguridad e integridad. Proporciona independencia de datos físicas desde el entorno distribuido. Los esquemas externos globales proporcionan independencia de datos lógica.

Esquemas de fragmentación y localización: El esquema de fragmentación es una descripción de cómo los datos están particionados lógicamente. El esquema de localización es una descripción de dónde están localizados los datos. El esquema de localización tiene en cuenta cualquier replicación.

Esquemas locales: Cada SGBD local tiene su propio conjunto de esquemas. Los esquemas conceptual e interno locales corresponden a los equivalentes de la arquitectura ANSI-SPARC. (Martín, 1999-2000)

1.4 Sistemas Gestores de Bases de Datos (SGBD)

Entre los SGBD más utilizados se encuentran: Oracle, Mysql, PostgreSQL, SQL Server de Microsoft.

1.4.1 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el *copyright* del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia *General Public License* (GPL). (Pecos, 2002)

Características:

Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación

multihilo. Soporta gran cantidad de tipos de datos para las columnas. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc). Gran portabilidad entre sistemas. Soporta hasta 32 índices por tabla. Gestión de usuarios y *passwords*, manteniendo un muy buen nivel de seguridad en los datos. Condición de *open source* de MySQL hace que la utilización sea gratuita y se puede modificar con total libertad. Se puede descargar su código fuente.

Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones. Es una de las herramientas más utilizadas por los programadores orientados a Internet. Infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación. MYSQL, es el manejador de base de datos considerado como el más rápido de Internet. Gran rapidez y facilidad de uso. Infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación. Fácil instalación y configuración. (Culiacán, 2009)

1.4.2 Oracle

El SGBD Oracle, fabricado por Oracle *Corporation*, utiliza la arquitectura cliente/servidor. Ha incorporado en su sistema el modelo objeto-relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. Así ofrece un servidor de bases de datos híbrido. Es uno de los más conocidos y ha alcanzado un buen nivel de madurez y de profesionalidad. Se destaca por su soporte de transacciones, estabilidad y escalabilidad. Los tipos objeto de Oracle son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos. A partir de la versión 10g del año 2004, se añade a los servidores la capacidad de funcionar según el paradigma de "*Grid*" (o rejilla) y se ofrecen mejoras en la administración e integración de algunos elementos que previamente no funcionaban correctamente juntos.

Características

Oracle es un Sistema Gestor de Bases de Datos con características objeto-relacionales, que pertenece al modelo evolutivo de SGBD. Sus características principales son las siguientes:

Entorno cliente/servidor. Gestión de grandes bases de datos. Usuarios concurrentes. Alto rendimiento en transacciones. Sistemas de alta disponibilidad. Disponibilidad controlada de los datos de las aplicaciones. Adaptación a estándares de la industria, como SQL-92. Gestión de la seguridad. Autogestión de la integridad de los datos. Opción distribuida. Portabilidad. Compatibilidad. Conectabilidad. Replicación de entornos. (Velasco, 2004)

1.4.3 Microsoft SQL Server.

Es un sistema para la gestión de bases de datos creado por Microsoft, el mismo se basa en el modelo relacional. Microsoft SQL Server utiliza como lenguajes de consulta T-SQL y Instituto Nacional de Normalización Estadounidense (*ANSI* por su sigla en inglés) SQL. Microsoft SQL Server revoluciona el concepto de Base de datos para la Empresa. Reúne en un sólo producto la potencia necesaria para cualquier aplicación empresarial, crítica junto con unas herramientas de gestión que reducen al mínimo el coste de propiedad. Con Microsoft SQL Server, la empresa tiene todo de serie.

Características fundamentales

Microsoft SQL Server revoluciona el concepto de Base de datos para la Empresa. Reúne en un sólo producto la potencia necesaria para cualquier aplicación empresarial, crítica junto con unas herramientas de gestión que reducen al mínimo el coste de propiedad. Con Microsoft SQL Server, la empresa tiene todo de serie.

Dentro de sus características fundamentales se encuentran:

Soporte de transacciones. Escalabilidad, estabilidad y seguridad. Soporta procedimientos almacenados. Incluye también un potente entorno gráfico de

administración, que permite el uso de comandos de lenguaje de definición de datos (*Data Definition Language*, *DDL)* y Lenguaje de Manipulación de Datos (*Data Manipulation Language*, *DML*) gráficamente. Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información. Además permite administrar información de otros servidores de datos.
(Ecured, 2011)

1.4.4 PostgreSQL

PostgreSQL es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. (GuiaUbuntu, 2011)

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

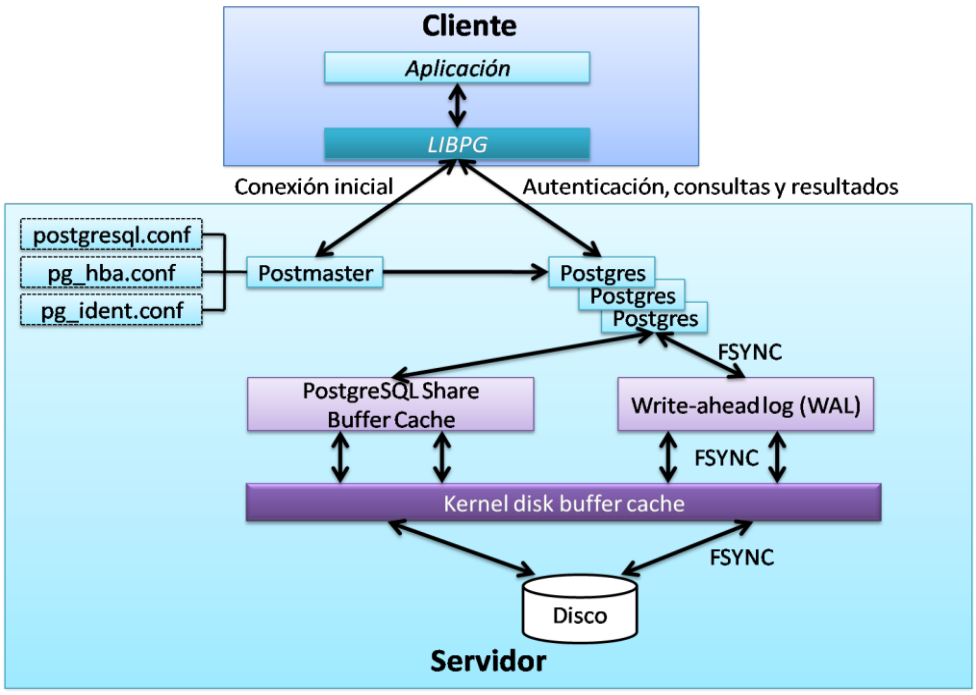


Figura 6. Componentes más importantes en un sistema PostgreSQL.

- ✓ **Aplicación cliente**: Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir vía TCP/IP ó sockets locales.
- ✓ **Demonio postmaster**: Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargaran de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes
- ✓ **Ficheros de configuración:** Los 3 ficheros principales de configuración utilizados por PostgreSQL, *postgresql.conf, pg_hba.conf y pg_ident.conf*
- ✓ **Procesos hijos postgres**: Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes
- ✓ **PostgreSQL share buffer cache**: Memoria compartida usada por POstgreSQL para almacenar datos en caché.
- ✓ ***Write-Ahead Log (WAL)****:* Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO)
- ✓ ***Kernel disk buffer cache***: Caché de disco del sistema operativo
- ✓ **Disco:** Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione.

Características de PostGreSQL

La última serie de producción es la 9.1. Sus características técnicas la hacen una de las bases de datos más potentes y robustos del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las

características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

A continuación alguna de las características más importantes y soportadas por PostgreSQL:

Generales

- Es una base de datos 100% ACID (*atomicity, consistency, isolation, durability*)
- Integridad referencial
- *Tablespaces*
- *Nested transactions (savepoints)*
- Replicación asincrónica/sincrónica / *Streaming replication - Hot Standby*
- *Two-phase commit*
- *PITR - point in time recovery*
- Copias de seguridad en caliente (*Online/hot backups*).
- *Unicode.*
- Juegos de caracteres internacionales.
- Regionalización por columna.
- *Multi-Version Concurrency Control (MVCC).*
- Multiples métodos de autentificación.
- Acceso encriptado vía SSL.
- Actualización in-situ integrada (pg_upgrade).
- SE-postgres.
- Completa documentación.
- Licencia *Berkeley Software Distribution* (BSD).
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit. (Martinez, 2010)

Ventajas

Posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos *benchmarks* se dice que ha llegado a soportar el triple de carga de lo que soporta MySQL).

Implementa el uso de *rollback's*, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que MySQL no podría. Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser Oracle.

Desventajas

Consume gran cantidad de recursos. Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.

Es de 2 a 3 veces más lento que MySQL. (Ecured, 2011)

1.5.5 Selección del Sistema Gestor de Base de Datos

La selección del sistema gestor utilizado, se realizó luego de un estudio comparativo de los sistemas gestores expuestos anteriormente.

Criterios	Sistemas Gestores de base de Datos			
	SQL Server	MySql	Oracle	PostgreSql
Plataforma	Windows	Windows/GNU/Linux	Windows/GNU/ Linux	Windows/GNU/Linux
Coste	-	+	-	+
Potencia	+	+	+	+
Velocidad	+	+	+	-
Escalabilidad	+	-	+	+

Requerimientos de hardware	+	+	-	+
Aspecto positivo: +			Aspecto negativo-	

Tabla 1. Comparación entre los SGBD

Luego del estudio comparativo llegamos a la conclusión de que el sistema más indicado para el desarrollo de la BD del módulo de economía y finanzas es el PostgreSQL en su versión 9.1. Esta elección se basa en que SQL Server no es multiplataforma, Oracle para ser usado es necesario pagar importes por su licencia, lo que no cumple con las políticas de migración al software libre de la universidad.

Por otra parte PostgreSQL y MySQL se acogen a las políticas de la Universidad, sin embargo, se selecciona el primero ya que en comparación con MySQL posee una gran escalabilidad. Además PostgreSQL es capaz de ajustarse al número de computadoras y a la cantidad de memoria que posee el sistema de forma óptima, por lo que está apto para soportar mayor cantidad de peticiones simultáneas de manera correcta y tolera el triple de carga de lo que podría MySQL.

Asimismo MySQL no es viable para su uso con grandes bases de datos, ya que como se decía antes no implementa una buena escalabilidad.
El uso de PostgreSQL es cada vez mayor en las empresas que buscan un servidor de base de datos altamente sofisticado, con alto rendimiento, estable y capacitado para lidiar con grandes volúmenes de datos. El hecho de ser un producto de código abierto, sin costos de licencia, convierte al PostgreSQL en una alternativa extremadamente atractiva para las empresas que buscan un ahorro significativo de costos en activos.

1.5 Herramientas de Modelado

Las herramientas de modelado utilizadas para el desarrollo y la administración de la base de datos de la presente investigación se encuentran:

1.5.1 SQL Power Architect 0.9.13

El SQL Power Architect herramienta de modelado de datos que fue creada por los diseñadores de almacenamiento de datos y tiene muchas características únicas dirigidas específicamente para el arquitecto de almacenamiento de datos. Permite a los usuarios de la herramienta ingeniería inversa de bases de datos existentes, realizar perfiles de datos en bases de datos de origen y generar automáticamente los metadatos de ETL.

Algunas Características de SQL Power Architect.

- Permite acceder a las bases de datos a través de Java Database Connectivity (JDBC).
- Puedes conectarte a múltiples bases de datos al mismo tiempo. Compara modelos de datos y estructuras de bases de datos e identifica las discrepancias.
- *Drag-and-drop* de las tablas origen y las columnas en el área de trabajo.
- Ingeniería directa/inversa para PostgreSQL, Oracle, MS SQL Server y muchas más.
- Todos los proyectos se guardan en formato XML.
- *On-Line Analytical Processing* (OLAP) modelos de esquema: cubos, medidas, dimensiones, jerarquías y niveles.

(TuInformaticaFacil, 2010)

1.5.2 *Visual Parading 6.4*

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Propósito

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista.

Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos. (Ecured, 2011)

1.6 Tecnologías usadas

Dentro de las tecnologías utilizadas para el desarrollo y administración de la base de datos de la presente investigación se encuentran las siguientes:

1.6.1 Hibernate 3.6+

Es una herramienta para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Hibernate es una herramienta *Object-relational mapping* (ORM) completa que ha conseguido en un tiempo record una excelente reputación en la comunidad de desarrollo posicionándose claramente como el producto *OpenSource* líder en este campo gracias a sus prestaciones, buena documentación y estabilidad. Es valorado por muchos incluso como solución superior a productos comerciales dentro de su enfoque, siendo una muestra clara de su reputación y soporte la reciente integración dentro del grupo JBoss que seguramente generará iniciativas muy interesantes para el uso de Hibernate dentro de este servidor de aplicaciones.

Hibernate – Características

1. No intrusivo (estilo POJO)
2. Muy buena documentación (forums para ayuda, libro)

3. Comunidad activa con muchos usuarios
4. Transacciones, caché, asociaciones, polimorfismo, herencia, *lazy loading*, persistencia transitiva, estrategias de *fetching*.
5. Potente lenguaje de consulta (HQL): *subqueries, outer joins, ordering*, proyeccion (*report query*), paginación.
6. Fácil testeo.
7. No es estándar. (Unifé, 2007)

1.6.2 NetBeans 7.0.1

NetBeans es un entorno de desarrollo – una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE (*Integrated Development Environment*). NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans tiene todas las herramientas necesarias para crear aplicaciones profesionales de Cliente-Servidor, Empresariales, Webs y Aplicaciones móviles con la plataforma Java, así como con C / C + +, PHP, *JavaScript* y *Groovy*. Este IDE introduce soporte de idiomas para el desarrollo de la propuesta de especificación de Java SE 7, el apoyo a *GlassFish* 3.1, Oracle *WebLogic*, Oracle DATABSE, Maven 3, HTML 5 y más.

El IDE es mucho más que un editor de texto: El editor de NetBeans tiene un amplio abanico de ventajas que puede facilitarte el escribir código, resalta el código fuente, asistente sintáctico y semántico. El editor soporta varios lenguajes como Java, C / C + +, XML, HTML, PHP*, Groovy, Javadoc, JavaScript* y JSP (*JavaServer Pages*)**.** Puede ser extendida para soportar otros idiomas. (Ibotme, 2011)

1.6.3 Framework Spring 3.0.2

Spring Framework provee amplio soporte para Hibernate. En particular, brinda implementaciones de DAO que ofrecen diversas utilidades para acceder a la *session* de Hibernate. Sus características principales son inyección de dependencias y programación orientada a aspectos.

Inyección de dependencias:

El objetivo es lograr un bajo acoplamiento entre los objetos de nuestra aplicación. Con este patrón de diseño, los objetos no crean o buscan sus dependencias (objetos con los cuales colabora) sino que éstas son dadas al objeto. El contenedor (la entidad que coordina cada objeto en el sistema) es el encargado de realizar este trabajo al momento de instanciar el objeto.

Programación orientada a aspectos

Se trata de un paradigma de programación que intenta separar las funcionalidades secundarias de la lógica de negocios. En inglés denominan a estas funcionalidades "*cross-cutting concerns*" algo que se traduciría como "preocupaciones transversales". La **AOP** (*Aspect-Oriented Programming*) busca modularizar estos servicios y aplicarlos de manera declarativa a los componentes que deban afectar.

Módulos de Spring

AOP – provee la implementación de AOP, permitiéndonos desarrollar interceptores de método y puntos de corte para desacoplar el código de las funcionalidades transversales.

DAO - Provee una capa de abstracción sobre JDBC, abstrae el código de acceso a datos de una manera simple y limpia. Tiene una capa de expeciones sobre los mensajes de error provistos por cada servidor específico de base de datos. Además cuenta con manejo de transacciones a través de AOP.

ORM – Provee la integración para las distintas APIs de mapeo objeto-relacional incluyendo JPA, JDO, Hibernate e iBatis.

JEE – Provee integración con aplicaciones Java Enterprise Edition así como servicios JMX, JMS, EJB, etc.

Web – Módulo que aporta clases especiales orientadas al desarrollo web e integración con tecnologías como Struts y JSF. Cuenta con el paquete **Spring MVC**, una implementación del conocido patrón de diseño aplicando los principios de Spring. (picandocodigo, 2010)

1.7 Conclusiones del capítulo.

En el presente capítulo se analizó lo referente a la fundamentación teórica de las bases de datos, Con el estudio de las características de los diferentes SGBD se define el PostgreSQL como el indicado para la administración de la base de datos. También se presentan las características de las tecnologías y herramientas usadas.

Capítulo 2. Diseño y arquitectura de la base de datos

En este capítulo se brinda información acerca de las arquitecturas de los sistemas de base de datos, así como de las bases de datos, las técnicas de diseño de bases de datos. Se exponen aspectos relativos a los requisitos del sistema. Se analiza el modelo de la base de datos para el almacenamiento de la información generada en el grupo de economía y finanzas.

2.1 Arquitectura para los Sistemas de Bases de Datos

El estándar ANSI/SPARC: El objetivo principal de la arquitectura ANSI/SPARC es definir un SGBD con el máximo grado de independencia, separando las aplicaciones de usuario y la base de datos física. Para ello se utilizan tres niveles de abstracción conocidos como interno, conceptual y externo.

Arquitectura de sistemas de Base de Datos

Arquitectura centralizada: Los sistemas de bases de datos centralizados son aquellos que se ejecutan en un único sistema informático sin interaccionar con ninguna otra computadora. Tales sistemas comprenden el rango desde los sistemas de bases de datos monousuario ejecutándose en computadoras personales hasta los sistemas de bases de datos de alto rendimiento ejecutándose en grandes sistemas. (Martín, 1999-2000)

2.1.1 Arquitectura usada

La arquitectura usada por el sistema de Base de Datos es la centralizada la que representa la más clásica de todas, pues ella permite al SGBD estar implantado en una sola plataforma desde donde se gestiona directamente de modo centralizado la totalidad de los recursos. La arquitectura del sistema gestor de base de datos usada es la cliente/servidor.

2.2 Técnicas para diseñar bases de datos relacionales

Existen dos escuelas que implican dos grupos de algoritmos a la hora de aplicar la Teoría de Normalización: los métodos de Análisis o descomposición y los Procedimientos de síntesis, que a continuación se detallan.

Diseño descendente, Análisis o descomposición

Este método es el que se emplea mediante un proceso denominado "normalización". El mismo parte de una relación llamada Universal que contiene todos los atributos del universo del discurso y las dependencias funcionales existentes entre ellas. Por medio de descomposiciones sucesivas y cumpliendo determinados principios de conservación de la información y de las dependencias, resultan esquemas de menor grado en formas normales cada vez más avanzadas, por lo que se puede garantizar que se reducen las anomalías.

Se pueden aplicar estos métodos también a cada una de las relaciones obtenidas a partir de un esquema conceptual en un modelo de datos de alto nivel (Modelo Entidad-Relación) y luego transformar el esquema conceptual a un conjunto de relaciones, empleando un procedimiento de transformación.

También se puede aplicar el método a cada una de las relaciones que se obtienen de la transformación del Diagrama Entidad-Relación al modelo relacional. El objetivo inicial que se pretende con estos métodos es separar la información referente a un objeto o entidad diferente.

Síntesis relacional o Procedimientos de síntesis

Este es un método de diseño alternativo a la descomposición, resulta ser un enfoque más purista. Implica contemplar el diseño de esquemas de BD relacionales estrictamente en términos de dependencias funcionales, y otros tipos especificados para los atributos de la BD. Aquí los esquemas de relación en tercera forma normal o forma normal de Boyce-Codd se sintetizan gradualmente agrupando los atributos apropiados.

A partir de conjuntos de atributos y dependencias funcionales se obtienen relaciones. Recorre un camino inverso a la descomposición, es decir, busca agrupar atributos, a fin de tener en una relación toda la información correspondiente a un objeto o entidad. (Carrasco, 2011)

2.2.1 Normalización

La teoría de la normalización se ha desarrollado para obtener estructuras de datos eficientes que eviten las anomalías de actualización. El concepto de normalización fue introducido por E.D. Codd y fue pensado para aplicarse a sistemas relacionales. Sin embargo, tiene aplicaciones más amplias.

La normalización es la expresión formal del modo de realizar un buen diseño. Provee los medios necesarios para describir la estructura lógica de los datos en un sistema de información.

Ventajas:

• Evita anomalías en la actualización.

• Mejora la independencia de los datos, permitiendo realizar extensiones de la BD, afectando muy poco, o nada, a los programas de aplicación existentes que acceden la base de datos.

La normalización involucra varias fases que se realizan en orden. La realización de la 2da fase supone que se ha concluido la 1ra y así sucesivamente. Tras completar cada fase se dice que la relación está en:

Primera Forma Normal (1FN)

Formalmente: Una relación está en (1FN) si cumple la propiedad de que sus dominios no tienen elementos que, a su vez, sean conjuntos. Toda relación normalizada, o sea, con valores atómicos de los atributos. La relación no incluye ningún grupo repetitivo.

Segunda Forma Normal (2FN)

Una relación R se dice que está en 2FN si está en 1FN y si, y sólo si, los atributos no llaves (ni primarias, ni candidatas) de R, si los hubiese, son funcional y completamente dependientes de la llave primaria de R. Entonces, este segundo paso se aplica sólo con relación a llaves compuestas.

Tercera Forma Normal (3FN)

Una relación R está en 3FN si está en 2FN y si, y sólo si, los atributos no llaves son independientes de cualquier otro atributo no llave primaria. Esto es lo mismo que decir que se deben eliminar las dependencias transitivas de atributos no llaves respecto a la llave primaria, estando ya la relación en 2FN.

Forma Normal de Boyce-Codd (FNBC)

Una relación R está en FNBC si, y sólo si, cada determinante es una llave (candidata o primaria). Obsérvese que se habla en términos de llaves candidatas y no sólo de la llave primaria, ya que una llave es un caso especial de superllave y la llave puede ser candidata o primaria. Además, la definición de FNBC es conceptualmente más simple, aunque es una FN más "fuerte". Una relación que esté en FNBC está también en 1FN, 2FN y 3FN. (Mato Garcia, 2004)

2.2 Requisitos del sistema

Los requisitos del sistema son las capacidades y propiedades que el sistema debe cumplir estos se clasifican en funcionales y no funcionales

2.2.1 Requerimientos funcionales

Son capacidades o condiciones que el sistema debe cumplir. En la realización de los casos de uso del negocio, se obtienen las actividades que serán objeto de automatización. Estas actividades no son exactamente los requisitos funcionales, pero sí son el punto de partida para identificar qué debe hacer el sistema.

Los requisitos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen. (Conferencia de Ingenieria de Software, 2011)

Se identificaron 67 requisitos funcionales para la solución del problema, de los cuales se presentan algunos a continuación:

R1 Insertar los datos sobre la Empresa de Servicios Comunales

R2 Modificar los datos sobre la Empresa de Servicios Comunales

R3 Buscar los datos sobre la Empresa de Servicios Comunales

R4 Eliminar los datos sobre la Empresa de Servicios Comunales

R5 Generar reporte de la Empresa de Servicios Comunales

R6 Generar Resumen Valorativo de la Empresa de Servicios Comunales

2.3 Estándares de nomenclatura

Los estándares de nomenclatura se establecen con el objetivo de que las bases de datos sean uniformes.

Objeto	Prefijo	Observación	Ejemplo
Tablas de datos	d_<texto>		d_envase
Tablas de nomencladores	n_<texto>		n_articulos
Campo de llave primaria	id_<texto>	El texto debe estar refiriendo el concepto o clave de negocio representado en esta tabla.	id_mantenimiento

Tabla 2. Estándares de nomenclatura

2.3.1 Tipos de datos

Tipo de datos "*serial*"

El tipo de datos "serial" es un tipo numérico que se incrementa automáticamente admitiendo el dinamismo para la generación de las claves primarias de las entidades, proporcionando números enteros consecutivos a cada tupla que se registra. En el caso de las claves primarias de las personas, su uso obstruye el

curso normal de réplica de datos. Cuando se refiere a la replicación de información surge la tendencia a la duplicación de datos ya existentes, o a la no inclusión de los mismos por existir su llave en el nodo destino.

A causa de ello se propone el uso de un mecanismo interno para la generación de las claves, que está basado en una "variante" del md5. Es decir, dado que el md5 es un algoritmo usado para la generación de un texto encriptado, se usa para generar las claves a partir de la información recogida en uno de los campos de la tabla en cuestión, o puede ser a partir de la combinación lineal de varios campos.

Tipo de datos "*integer*"

Se utiliza para los campos con valores enteros positivos.

Tipo de datos "*date*"

Los tipos de dato date almacena la fecha sin hora de la forma 'aaaa-mm-dd'

Tipo de datos "*varchar*(n)"

Los tipos de datos *varchar* almacenan datos compuestos de caracteres en mayúsculas y minúsculas, como por ejemplo: a, b y C; números como 1, 2 y 3; y caracteres especiales como el signo de arroba (@), "y" comercial (&) y el signo de exclamación de cierre (!). Además este tipo de datos es de longitud variable, n puede ser un valor entre 1 y 8000. En el trabajo con la BD se tiene en cuenta lo siguiente:

varchar (11) representa los campos del carné de identidad.

varchar (35) representa campos cortos.

varchar (50) representa campos medianos.

varchar (100) representa campos largos.

Tipos de datos "*double*"

Los tipo de dato *double* almacenan un valor en punto flotante de doble precisión con un rango de - 1.79769313486232*10308 a -4.94065645841247*10-324 para

valores negativos, 4.94065645841247*10-324 a 1.79769313486232*10308 para valores positivos, y 0.

Tipos de dato "*timestamp*"

Los tipos de datos *timestamp* almacenan datos compuestos por fecha y hora con zonificación. (Carrasco, 2011)

2.4 Diseño de la base de datos

Un Diagrama o Modelo Entidad Relación (a veces denominado por su siglas, E-R "*Entity Relationship*", o, "DER" Diagrama Entidad Relación), es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades. Propuesto por Peter Chen en 1976, mediante el mismo se pretenden 'visualizar' los elementos que pertenecen a una Base de Datos, que reciben el nombre de entidades, las cuales se corresponden con el concepto de clase de la Programación Orientada a Objeto y donde cada tupla de una futura relación representaría un objeto de la Programación Orientada a Objetos. (Conferencia de base de Datos, 2011)

2.4.1 Diagrama semántico de los datos

La modelización semántica de los datos consiste en estudiar los datos que se pretenden almacenar en la base de datos. Esta modelización supone los objetos de interés de la organización (entidades), las propiedades relevantes de dichos objetos (atributos), y las relaciones de estos entre sí (relaciones). Se presenta un pequeño fragmento del diagrama de datos, el diagrama completo se encuentra en el Anexo1:

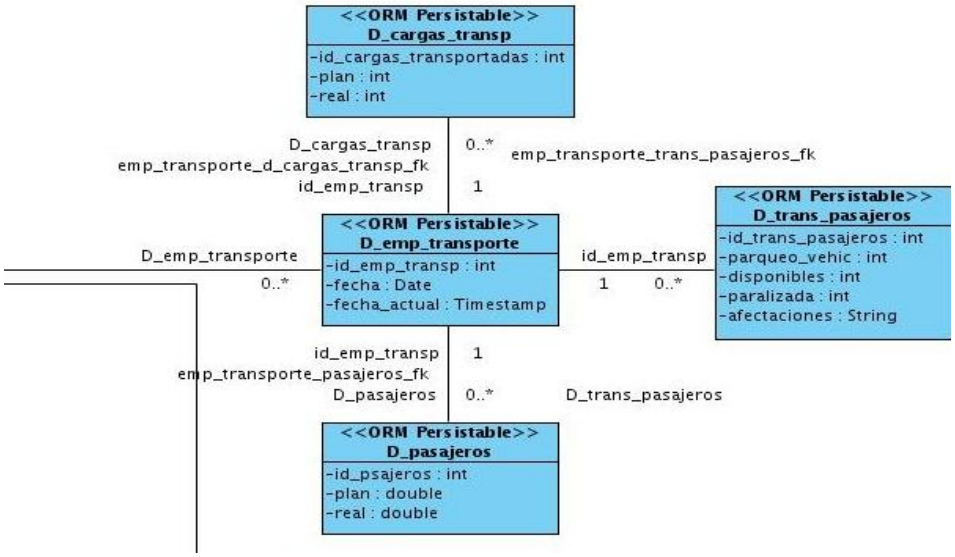


Figura 7. Fragmento del diagrama de Datos

2.4.2 Diagrama de Entidad y Relación de la base de datos del módulo de Economía y Finanzas de la Dirección de Grupo Empresarial.

El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad, mediante un conjunto de representaciones gráficas y lingüísticas.

La base de datos del módulo de Economía y Finanzas de la Dirección de Grupo Empresarial, cuenta con 41 entidades, de ellas 7 nomencladores y 34 tablas, en este módulo se hace uso también del nomenclador nmunicipio que se encuentra dentro de los nomencladores comunes. El DER de esta BD muestra las relaciones que existen entre las tablas, se presenta un pequeño fragmento de este, el diagrama completo se muestra en el Anexo 2:

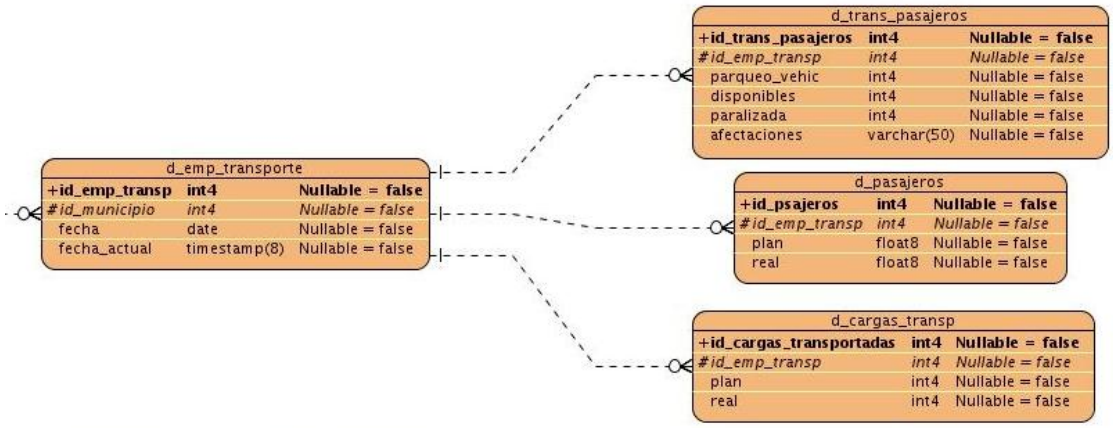


Figura 8. Fragmento del diagrama de Entidad y Relación

2.4.3 Modelo Físico

El paso de convertir el modelo lógico a tablas hace que las entidades pasen a ser tablas y los atributos se convierten en las columnas de dichas tablas. A continuación se muestra algunas de la tabla del modelo físico:

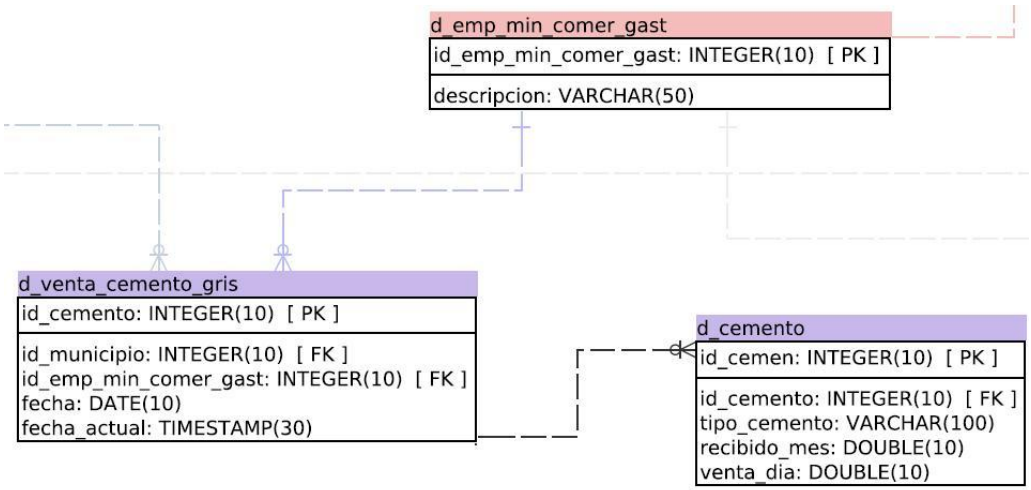


Figura 9. Modelo físico

2.4.4 Descripción de las clases persistentes

Las clases persistentes y sus atributos hacen referencia directamente a las entidades lógicas y a sus atributos. Se presente un ejemplo de la descripción de una clase persistente, para consultar todas las descripciones de las clases dirigirse al Modelo Canónico de Datos:

Clase	emp_min_comer_gast			
Descripción:	Se almacena la información acerca de la empresa minorista de comercio y gastronomía.			
Tipo	*superclase*			
Roles o Propiedades	**Descripción**	**Valor requerido**	**Llave**	**Relación**
id_emp_min_comer_gast	Identificador propio de la tabla.	Entero, identificador	*X*	
descripción	Descripción del identificador	Cadena de caracteres		

Tabla 3. Ejemplo de una clase

2.5 Diagrama de Paquetes

Refleja la organización de los paquetes y sus elementos. Muestra las agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Suministran una descomposición de la jerarquía lógica de un sistema. Cada paquete puede concederse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo necesitado. Se presenta el diagrama de paquetes del sistema que se propone:

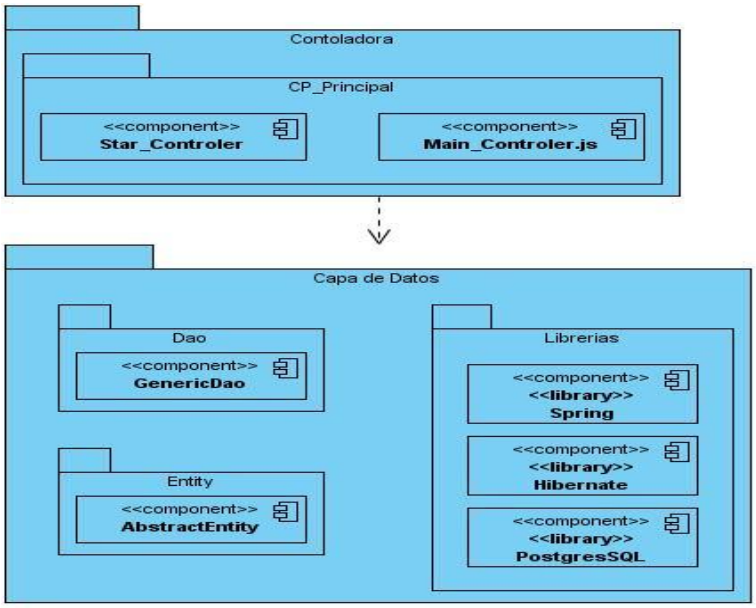


Figura 10. Diagrama de paquetes

La capa controladora es la encargada de integrar todas las clases y funcionalidades relacionadas con el negocio, esta se relaciona con la capa de datos, en la capa de datos se encuentran las librerías utilizadas y las clases con que cuenta el sistema. Permitiendo de esta forma acceder a los datos almacenados en la base de datos.

2.6 Conclusiones del Capitulo

En el presente capítulo se expone la arquitectura del sistema de base de datos, el proceso de normalización, como técnica para diseñar bases de datos relacionales. Se describe los estándares de nomenclatura y tipos de datos. Además, se plantean los principales requisitos funcionales. Se presenta el diagrama del modelo entidad-relación, el diagrama de datos, diagrama de componentes y algunas de las descripciones de las clases persistentes.

Capítulo 3. Implementación de la capa de acceso y validación de la propuesta de solución

En este capítulo se detalla cómo se realiza la implementación de la capa de acceso a datos, con esta una descripción del mapeo de las clases, así como la descripción de los métodos de la clase GenericDao. También se realiza la validación de los métodos implementados para la persistencia de objetos.

3.1 Implementación de la capa de acceso a datos

Para la implementación de la capa de acceso a datos se utilizó el IDE Netbeans 7.0.1 con la característica de soporte a persistencia, el cual a través de asistentes permite configurar la base de datos para la generación de archivos POJOS.

3.1.1 Implementación de las clases

Para mapear las clases desde la base de datos es necesario llevar a cabo una secuencia de pasos, lo primero es crear la conexión desde Netbeans con la base de datos, para ello se especifica, el host en el cual se encuentra la base de datos, el nombre, el esquema entre las cosas más importantes, a continuación se procede a la creación del fichero <hibernate.cgf.xml> que en él se encuentra la conexión anteriormente realizada a la base de datos.

```
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>
    <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
    <property name="hibernate.connection.url">jdbc:postgresql://10.208.1.250:5432/sigob</
    <property name="hibernate.connection.username">user</property>
    <property name="hibernate.connection.password">user</property>
    <mapping class="model.entity.DEmpTransporte"/>
    <mapping class="model.entity.DCemento"/>
  </session-factory>
</hibernate-configuration>
```

Figura 11. Archivo de configuración de Hibernate

Seguidamente se procede a la creación del archivo de ingeniería inversa <hibernate.reveng.xml>, donde se especifica el archivo de configuración, con este se obtiene las clases que serán mapeadas.

```
<hibernate-reverse-engineering>
  <schema-selection match-catalog="sigob" match-schema="mod_infyserv_grupo_empresarial"/>
  <table-filter match-name="d_construccion_matenimiento"/>
  <table-filter match-name="d_envase_saco"/>
  <table-filter match-name="n_prod_naturales"/>
 </hibernate-reverse-engineering>
```

Figura 12. Archivo de Ingeniería Inversa

Una vez creados los archivos de configuración se procede a la generación de los archivos POJOS. Estos archivos POJOS se crean a través de anotaciones EJB3 (*Enterprise JavaBeans*) esta genera POJOs anotados según el estándar de EJB3. Esto es una alternativa a los ficheros xml de mapeo, de forma que, mediantes estas anotaciones, en el mismo POJO es donde se indica cómo se debe mapear con la base de datos. Estas anotaciones además de evitarnos mantener esos XML, tienen la ventaja de que son compatibles con las anotaciones de la nueva especificación 3 de EJBs (podríamos convertir nuestros POJOs en EJBs de forma casi directa, o usar nuestros POJOs con la capa de persistencia de EJB3 en vez de con Hibernate). Esta también permite un mejor manejo de la aplicación ya que no cuenta con tantos ficheros. Las clases mapeadas quedarían de la siguiente manera:

```
@Entity
@Table(name = "d_construccion_matenimiento", schema = "mod_infyserv_grupo_empresarial")
public class DConstruccionMatenimiento extends AbstractEntity {

    private int idConstru;

    public DConstruccionMatenimiento() {
    }

    public DConstruccionMatenimiento(int idConstru, DEconomiaFinanzas DEconomiaFinanzas,
        this.idConstru = idConstru;

    }

    @Id
    @Column(name = "id_constru", unique = true, nullable = false)
    public int getIdConstru() {
        return this.idConstru;
    }

    public void setIdConstru(int idConstru) {
        this.idConstru = idConstru;
    }
    @ManyToOne(fetch = FetchType.EAGER)
```

Figura 13. Ejemplo de un POJO

Las anotaciones en Hibernate

@Entity: A partir de esta *annotation* indicamos a Hibernate que nuestra clase es persistente. Se recomienda el uso de la *annotation* Entity de JPA, la de Hibernate suele traer dificultades. Una vez declarada persistente nuestra clase, todos sus atributos lo serán también por default.

@Table (*name* = "ITEM"): A través de esta *annotation* especificamos un nombre de tabla en caso de diferir al de nuestra clase. De no indicarlo, Hibernate tomará como nombre de la tabla asociada, el mismo que el de la clase. Esta es una convención, y constituye un ejemplo sencillo del concepto **"***convention over configuration***"**.

@Id: A través de esta *annotation* indicamos que la *property* mapeada será la PK de nuestra tabla. En este caso se trata de una clave simple.

@GeneratedValue: Especifica la estrategia de generación de Ids. Hay varias opciones para generadores. En nuestro caso, al no acompañarla de parámetros

adicionales, su valor será AUTO, que decide una estrategia de generación de Ids conveniente dependiendo del motor subyacente. (ej. **Identity**).

@*Column*(name = "ITEM_ID"): En este caso la *annotation* es opcional, si pretendemos que la columna de la tabla correspondiente se llame igual que la property. Aquí valen las mismas consideraciones que en el caso de @Table.

@*ManyToOne*: Nuestra tabla ITEM tendrá una FK a la tabla FACTURA para establecer la relación, de acuerdo al dominio. Coloquialmente podemos traducir este extremo de la relación como: "un item puede pertenecer a una única factura". Como en el otro lado "una factura puede tener más de un item", marcamos este extremo de la relación (el del item) como "Many" y el opuesto (el de la factura), como "One". Esta *annotation* puede estar acompañada de ciertos parámetros, como el tipo de *fetching* y el *cascading*, a los que nos referiremos más tarde. (Tacs, 2012)

3.2 Integración de Spring con Hibernate

Spring provee de una clase llamada ***HibernateTemplate*** que provee de muchos métodos expuestos en una interfaz usada por Hibernate la ***Session*** adicionado con algunos otros métodos. *HibernateTemplate* asegura que las instancias de la *Session* sean correctamente abiertas y cerradas y automáticamente y hacer que las llamadas participen en transacciones. Las instancias de *Session* en *HibernateTemplate* son *thread-safe* y son reusables. Spring también provee de una clase muy útil llamada **HibernateDaoSupport** que da la propiedad de tener el *SessionFactory* y un método *getHibernateTemplate* para usarlo en subclases, esto dará la posibilidad de tener el *HibernateTemplate* en los DAOs para poder hacer implementaciones simples. El método ***execute*** del *HibernateTemplate* es el principal, el cual da acceso directo a la *Session*.

La clase HibernateDaoSupport

Spring provee la clase *HibernateDaoSupport* para brindarle a los DAO soporte para Hibernate.

***HibernateTemplate**:* En particular, uno de los métodos más útiles que provee es *getHibernateTemplate()*. Este método devuelve un *template* con varios métodos útiles, que simplifican el uso de Hibernate. Estos métodos suelen encapsular varias excepciones propias de acceso a datos de Hibernate (y SQL) dentro de una *DataAccessException* (que hereda de *RuntimeException*). (Julian, 2009)

SessionFactory: Este objeto de Hibernate maneja las sesiones que ejecutan las distintas transacciones a la base de datos.

3.3.1 Implementación de la clase GenericDao

Para facilitar el trabajo y encapsular la fuente de datos ocultando la forma de como se accede a los datos se implementó el patrón Dao, esta clase hereda de la clase *HibernateDaoSupport* la cual cómo se explica anteriormente tiene la propiedad de tener el *SessionFactory* y un método *getHibernateTemplate* para usarlo en subclases.

En esta clase genérica fueron creadas las funciones para el trabajo con las entidades, las funciones implementadas en esta clase tienen las siguientes funcionalidades:

- saveOrUpdate: Este funcionalidad recibe como parámetro un objeto, verifica la existencia del objeto en la base de datos, si es encontrado lo actualiza en caso contrario es salvado.

```
public void saveOrUpdate(Object entity) {
    getHibernateTemplate().saveOrUpdate(entity);

}
```

Figura 14. Función saveOrUpdate de la clase Dao Genérico

- save: Almacena el objeto pasado por parámetro.

```
public void save(Object entity){
    getHibernateTemplate().save(entity);
```

Figura 15. Función save de la clase Dao Genérico

- update: Actualiza un objeto determinado pasado por parámetro.

```
public  void update(Object entity){
    getHibernateTemplate().update(entity);
}
```

Figura 16 .Función update de la clase Dao Genérico

- find: Retorna una lista de todas las tuplas de la base de datos mapeadas como objetos, se le pasa como parámetro la clase que se desea buscar.

```
public <T> List<T> find(Class<T> entityClass) {
    return getHibernateTemplate().loadAll(entityClass);
}
```

Figura 17. Función find de la clase Dao Genérico

- findByPK: Busca y retorna un objeto a través de la llave primaria y el nombre de la clase entidad.

```
public Object findByPK(Class entityName, String id){
 try {
       return getHibernateTemplate().get(entityName, id);
    } catch (Exception e) {
       return getHibernateTemplate().get(entityName, Integer.parseInt(id));
    }
}
```

Figura 18. Función findByPK de la clase Dao Genérico

- delete: Elimina un objeto pasado por parámetro.

```
public void delete(Object entity) {
    getHibernateTemplate().delete(entity);
}
```

Figura 19. Función delete de la clase Dao Genérico

- deleteAll: Elimina todas las tuplas de una tabla recibiendo como parámetro el tipo de esta.

```
public <T> void deleteAll(Collection<T> collection){
    getHibernateTemplate().deleteAll(collection);
}
```

Figura 20. Función deleteAll de la clase Dao Genérico

- findByParamAndValue: Devuelve la lista de objetos que cumplan con una condición a partir del nombre de un campo y su valor.

```
public <T> List<T> findByParamAndValue(Class<T> entityClass, String param, Object value){

   DetachedCriteria criteria = DetachedCriteria.forClass(entityClass);
   criteria.add(Restrictions.eq(param, value));
   return getHibernateTemplate().findByCriteria(criteria);
}
```

Figura 21. Función findByParamAndValue de la clase Dao Genérico

- findByParamsAndValues: Retorna todos los objetos que cumplan con una serie de atributos y valores pasados por parámetro.

```
public <T> List<T> findByParamsAndValues(Class<T> entityClass, String[] params, Object[] values)

    List<T> list = new ArrayList<T>();
    if (params.length > 0 && values.length > 0) {
        DetachedCriteria criteria = DetachedCriteria.forClass(entityClass);
        criteria.add(Restrictions.eq(params[0], values[0]));

        if (params.length > 1 && values.length > 1) {
            for (int i = 1; i < params.length; i++) {
                criteria.add(Restrictions.eq(params[i], values[i]));
            }
        }

       list = getHibernateTemplate().findByCriteria(criteria);
    }

    return list;
}
```

Figura 22. Función findByParamsAndValues de la clase Dao Genérico

- findByExample: Retorna la entidad que contenga los atributos pasados por parametro.

```
public AbstractEntity findByExample(Object exampleEntity){
    return (AbstractEntity)getHibernateTemplate().findByExample(exampleEntity);
}
```

Figura 23. Función findByExample de la clase Dao Genérico

3.3 Validación de los datos

Para garantizar la protección de una BD es necesario que exista integridad en sus datos, esta se refiere a la corrección, consistencia, completitud y validez de los datos almacenados, con esto se garantiza que no ocurran operaciones que puedan provocar inconsistencia en los datos.

La integridad de la información en una base de datos puede ser violada con cada modificación de su contenido, resultando en la necesidad de comprobar todas y cada una de las restricciones de integridad.

3.3.1 Restricciones de integridad

Integridad de entidad:

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

Integridad de dominio:

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

Integridad referencial:

La integridad referencial protege las relaciones definidas entre las tablas cuando se crean o se eliminan filas. En SQL Server 2005 la integridad referencial se basa en las relaciones entre claves externas y claves principales o entre claves externas y claves exclusivas, mediante restricciones FOREIGN KEY y CHECK. La integridad referencial garantiza que los valores de clave sean coherentes en las distintas tablas. Para conseguir esa coherencia, es preciso que no haya referencias a valores inexistentes y que, si cambia el valor de una clave, todas las referencias a ella se cambien en consecuencia en toda la base de datos. (Morillo, 2010)

Resultados de las pruebas de integridad

Integridad de entidad: Este tipo de integridad se cumple en la BD propuesta en el módulo ya que cada tabla tiene definida su clave primaria. Las llaves primarias de las tablas son campos secuenciales o enteros que representan un código único, de esta forma este campo nunca será nulo ni se repetirá. Es de vital importancia especificar la clave principal para cada registro, ya que esta representa la herramienta fundamental que utiliza el servidor de BD para seleccionar la información que se necesite, evitando así la duplicidad de los registros.

Integridad de dominio: Para velar por esta integridad en la BD, no se definieron atributos nulos y se precisaron correctamente cada uno de los tipos de datos para cada uno de los atributos.

Integridad referencial: Para velar por el cumplimiento de esta integridad en la BD, no se puede introducir un valor en la tabla relacionada si antes no ha sido introducida en la tabla principal. Las acciones de eliminar y modificar se realizan en cascada de esta manera cuando se realiza un cambio en la tabla principal también se cambia en las relacionadas.

3.3.2 Validación funcional

Las pruebas funcionales se realizan con el objetivo de comprobar y observar cómo se comporta la base de datos bajo diferentes circunstancias, estas pruebas

aseguran que la misma cumpla con los requisitos definidos sin que se vea violada la integridad de los datos.

Pruebas a la capa de acceso a datos

Para la realización de las pruebas a la capa de acceso se realizaron nueve casos de pruebas, por cada método del GenericDao. A continuación se muestra los aspectos que se tuvieron en cuenta para la realización de las mismas, a los métodos para la persistencia de objetos:

Método:	**save**	
Condiciones:	Debe existir el objeto de la entidad que se desee salvar.	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Insertar en la BD el objeto que fue pasado por parámetro**.**	Se espera que el objeto pasado por parámetro se almacene en la tabla correspondiente a la BD.	El objeto pasado por parámetro se almacenó satisfactoriamente.
Método:	saveOrUpdate	
Condiciones:	Debe existir el objeto de la entidad que se desea salvar o actualizar.	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Insertar o actualizar en la BD el objeto que fue pasado por parámetro**.**	Se espera que no existir los objetos en la base de datos sean guardados y si existen entonces actualizarlos en las tablas y columnas correspondientes en la base de datos.	Los objetos fueron actualizados correctamente en la base de datos.
Método:	find	

Condiciones:	Se le debe especificar la entidad que se desea buscar.	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Obtener el listado de la entidad especificada.	Se espera obtener una lista con todos los registros existentes en la entidad especificada.	Fue obtenido satisfactoriamente la lista con los registros de la entidad especificada.
Método:	findByPK	
Condiciones:	Se le debe especificar la entidad y el valor de la llave primaria serializable.	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Obtener el objeto de la entidad especificada al que le corresponde la llave primaria indicada.	Se espera obtener el objeto que le corresponde la llave primaria indicada.	Fue obtenido el objeto especificado.
Método:	findByParamAndValue	
Condiciones:	Se le debe especificar la entidad, el parámetro y el valor por el que se desea buscar.	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Obtener el objeto de la entidad especificada que cumpla que el valor del parámetro se igual al	Se espera obtener el objeto cuyo valor del parámetro que se va a buscar sea igual al especificado.	Se obtuvo el objeto que presenta dicho parámetro con el valor indicado.

especificado.		
Método:	findByParamsAndValues	
Condiciones:	En la entidad debe existir dichos parametros	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Obtener el objeto de la entidad especificada que cumpla que el valor de los parámetros se igual a los especificados.	Se espera obtener el objeto cuyos valores de los parámetros que se va a buscar sean iguales a los especificados.	Se obtuvo el objeto que presenta los parámetros con los valores indicados.
Método:	update	
Condiciones:	El objeto de la entidad que se desea actualizar debe existir.	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Actualizar los atributos del objeto de la entidad especificada**.**	Se espera que los atributos del objeto sean actualizados.	Los atributos del objeto fueron actualizados.
Método:	delete	
Condiciones:	El objeto de la entidad especificada debe existir.	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Eliminar el objeto de la entidad especificada.	Se debe eliminar el objeto de la entidad.	El objeto de la entidad fue eliminado.

Método:	deleteAll	
Condiciones:	En la entidad deben existir al menos uno o varios objetos.	
Objetivo de la Prueba	**Resultado Esperado**	**Resultado de la Prueba**
Eliminar todos los objetos de la entidad especificada.	Se espera eliminar todos los objetos de la entidad.	Fueron eliminados los objetos de la entidad especificada.

Tabla 4. Aspectos para realizar las pruebas a los métodos del GenericDao

3.4 Conclusiones del capítulo

En este capítulo se detalla la realización del mapeo de las clases de la BD del módulo de economía y finanzas de la dirección de grupo empresarial; en esta se describe la configuración de Hibernate y todos los pasos para la realización de esta actividad. También se explica las funcionalidades de la clase Dao Genérico, así como la validación de la capa de acceso a datos.

Conclusiones Generales

Con el desarrollo del presente trabajo de investigación se puede afirmar que se logró el objetivo de desarrollar la base de datos e implementar la capa de acceso a datos para el módulo de Economía y Finanzas de la Dirección Grupo Empresarial de la Administración Provincial de Artemisa.

- Con el estudio de los procesos de gestión de información, se logró que el autor se adueñara de los conocimientos necesarios que le permitieron elaborar la fundamentación teórica de la presente investigación.
- Con el estudio de las metodologías para el diseño y su aplicación en el diseño de la base de datos para el módulo de Economía y Finanzas, se obtuvo el modelo de datos del sistema, con lo cual se logró un diseño eficiente y completo, que abarca toda la información que se gestionará en el mismo.
- Con el estudio de las herramientas y tecnologías a usar para el desarrollo de la solución propuesta, se logró seleccionar el SGBD, los Frameworks y las herramientas de modelado, que se utilizaron para desarrollar la propuesta.
- Con el estudio de los métodos para la persistencia de objetos y su desarrollo se logró implementar la capa de acceso a datos, que permite almacenar y recuperar los datos desde la base de datos.
- Con la comprobación de la integridad de los datos y la validación de los métodos para persistir objetos, se logró la optimización de la base de datos y la capa de acceso que permitió avalar la eficiencia del diseño.

Recomendaciones

- Desplegar el sistema en la administración provincial para obtener la opinión el usuario después de un tiempo de uso.
- Continuar análisis de la existencia de entidades en el módulo de Economía y Finanzas, que pueden ser agregadas a la BD, evitando de esta forma la desorganización y duplicación del código.
- Incluir nuevas funcionalidades para la persistencia de objetos.
- Continuar con la investigación para incluir nuevas mejoras futuras a la base de datos y capa de acceso, para nuevas versiones futuras.

Referencias Bibliográficas

A. Otto, D. Hinderink. 2004. *TYPO3 Database Abstraction.* 2004.

Aduana - CUBA - Customs. 2012. Aduana - CUBA - Customs. *Aduana - CUBA - Customs.* [En línea] 2012. http://www.aduana.co.cu/index.php?option=com_content&view=article&id=30&Itemid=82&lang=es.

Alvarez, Sara. 2007. Desarrollo Web. *Desarrollo Web.* [En línea] 31 de Julio de 2007. [Citado el: 1 de Junio de 2012.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

—. 2007. DesarrolloWeb. *DesarrolloWeb.* [En línea] 14 de Agosto de 2007. [Citado el: 13 de Enero de 2012.] http://www.desarrolloweb.com/articulos/modelos-base-datos.html.

Banco Central de Cuba. 2009. Banco Central de Cuba. *Banco Central de Cuba.* [En línea] 2009. http://www.bc.gov.cu/Espanol/default.asp.

Buc, La. 2005. Universidad de Cantabria. *Universidad de Cantabria.* [En línea] 10 de Marzo de 2005. http://www.buc.unican.es/Servicios/formacion/DEC/bases_datos.htm.

Carrasco, Yailin Fundora. 2011. *Diseño e implementación de la base de datos del Sistema de Gestión Académica de Postgrado.* La Habana : s.n., 2011.

Conferencia de base de Datos. 2011. Entorno Virtual de Aprendizaje. *Entorno Virtual de Aprendizaje.* [En línea] 2011. http://eva.hab.uci.cu.

Conferencia de Ingenieria de Software. 2011. Entorno Virtual de Aprendizaje. *Entorno Virtual de prendizaje.* [En línea] 2011. [Citado el: 20 de Febrero de 2012.] http://eva.hab.uci.cu.

CubaGenWeb.org. 1996-2010. Centro de la Genealogía Cubana. *Centro de la Genealogía Cubana*. [En línea] 1996-2010. http://www.cubagenweb.org/ships/e-index.htm.

Culiacán, Sinaloa. 2009. anadicSinaloa. *anadicSinaloa*. [En línea] 18 de Agosto de 2009. [Citado el: 1 de Febrero de 2012.] http://www.anadicsinaloa.com/index.php?option=com_content&view=article&id=207:caracteristicas-mysql&catid=16:anadic-sinaloa&Itemid=33.

Dataprix. 2007. Dataprix. *Dataprix*. [En línea] 2 de Enero de 2007. [Citado el: 19 de Enero de 2012.] http://www.dataprix.com/11-etapas-diseno-bases-datos.

Ecured. 2011. Ecured. *Ecured*. [En línea] 2011. [Citado el: 3 de Febrero de 2012.] http://www.ecured.cu/index.php.

eurostat. 2011. European Commission Eurostat . *European Commission Eurostat* . [En línea] 31 de Octubre de 2011. [Citado el: 1 de Junio de 2012.] http://epp.eurostat.ec.europa.eu/portal/page/portal/hicp/introduction.

Gómez, Marichelo. 2008. Slideshare. *Slideshare*. [En línea] 30 de Noviembre de 2008. [Citado el: 19 de Enero de 2012.] http://www.slideshare.net/marichelogomez/qu-son-las-bases-de-datos-presentation .

GuiaUbuntu. 2011. Guia Ubuntu. *Guia Ubuntu*. [En línea] 13 de Julio de 2011. [Citado el: 9 de Noviembre de 2011.] http://www.guia-ubuntu.org/index.php?title=PostgreSQL.

Herrera, Manolo. 2007. Desarrollo + . *Desarrollo* + . [En línea] sábado 6 de Enero de 2007. [Citado el: 17 de Diciembre de 2011.] http://jmhogua.blogspot.com/2007/01/programacin-en-capas-primera-parte-capa.html.

Ibotme. 2011. Ibotme. *Ibotme*. [En línea] 2011. [Citado el: 9 de Febrero de 2012.] http://www.ibotme.com/2011/05/entorno-de-desarrollo-para-ubuntu-linux-netbeans/.

Julian. 2009. Java en Castellano, Hibernate Framework, Struts Framework, Spring Framework y mucho más! *Java en Castellano, Hibernate Framework, Struts*

Framework, Spring Framework y mucho más! [En línea] 2009. http://javaencastellano-hibernate.blogspot.com/2009/09/la-clase-hibernatedaosupport.html.

Martín, Óscar González. 1999-2000. Escuela Superior de Informática. *Escuela Superior de Informática.* [En línea] 1999-2000. http://www.inf-cr.uclm.es/www/fruiz/bda/doc/trab/T9900_OGonzalez.pdf.

Martinez, Rafael. 2010. PostgreSql. *PostgreSql.* [En línea] 2 de Octubre de 2010. [Citado el: 1 de Diciembre de 2011.] http://www.postgresql.org.es/sobre_postgresql.

Mato Garcia, Rosa Maria. 2004. *Sistemas de Base de Datos.* Ciudad Habana : Felix Varela, 2004.

Mato, Rosa María García. 1999. *Diseño de Bases de Datos.* 1999.

Midepa. 2011. Historia de la Informática. *Historia de la Informática.* [En línea] 4 de Enero de 2011. [Citado el: 10 de Noviembre de 2011.] http://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/ .

Miranda, Alfredo Terán. 2007. Base de Datos. *Base de Datos.* [En línea] 11 de Octubre de 2007. [Citado el: 29 de Mayo de 2012.] http://basededatosrelacionales.blogspot.com/2007/10/concepto-de-base-de-datos-relacionales.html.

Morillo, Darwin Durand. 2010. slideshare. *slideshare.* [En línea] 5 de mayo de 2010. [Citado el: 3 de abril de 2012.] http://www.slideshare.net/sistemasddm/integridad-sql-server.

Oropeza, Lisbeth. 2007. lacoctelera. *lacoctelera.* [En línea] 25 de Noviembre de 2007. [Citado el: 1 de Junio de 2012.] http://basededatos.lacoctelera.net/post/2007/11/25/definicion-base-dato-relacional.

Panitsch, Andrés. 2008. desdesarrollodesoftware. *desdesarrollodesoftware.* [En línea] 26 de Agosto de 2008. [Citado el: 1 de Junio de 2012.] http://desdesarrollodesoftware.blogspot.com/2008/08/la-capa-de-acceso-datos.html.

Pecos, Daniel. 2002. Danielpecos.com. *Danielpecos.com*. [En línea] 7 de Junio de 2002. [Citado el: 1 de Febrero de 2012.] http://danielpecos.com/docs/mysql_postgres/x57.html.

picandocodigo. 2010. picandocodigo. *picandocodigo*. [En línea] 29 de noviembre de 2010. [Citado el: 11 de Abril de 2012.] http://picandocodigo.net/2010/introduccion-a-spring-framework-java/.

Tacs. 2012. TACS - Tecnologías Avanzadas en la Construcción de Software. *TACS - Tecnologías Avanzadas en la Construcción de Software*. [En línea] 2012. [Citado el: 1 de junio de 2012.] https://sites.google.com/a/tacs-utn.com.ar/tacsalumnos/Home/apuntes/hibernate---mapping-y-relaciones.

TuInformaticaFacil. 2010. TuInformaticaFacil. *TuInformaticaFacil*. [En línea] 19 de Octubre de 2010. [Citado el: 9 de Febrero de 2012.] http://www.tuinformaticafacil.com/herramientas-desarrollo/sql-power-architect-herramienta-de-modelado-de-datos.

Unifé. 2007. Unifé. *Unifé*. [En línea] 2007. [Citado el: 9 de Febrero de 2012.] www.unife.edu.pe/ing/desarrollo.doc.

Valdés, Damián Pérez. 2007. Maestros del Web. *Maestros del Web*. [En línea] 2007. [Citado el: Jueves 1 de Noviembre de 2011.] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

Velasco, Roberto Hernando. 2004. rhernando.net. *rhernando.net*. [En línea] 24 de Abril de 2004. [Citado el: 3 de Febrero de 2012.] http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.

Vitoria, Biblioteca Francisco de. 2009. Biblioteca Francisco de Vitoria. *Biblioteca Francisco de Vitoria*. [En línea] 2009. [Citado el: 1 de junio de 2012.]

Bibliografía

Campoy, Lourdes Arlín. 2009. Tutorial Base de Datos I. [En línea] 2009. http://sistemas.itlp.edu.mx/tutoriales/basedat1/tema1_1.htm..

Codd, Frank Edgar. 2009. Base de datos: Modelo de Datos - C# – ASP.NET – WEB – PHP. [En línea] 6 de 2 de 2009. http://www.imgeek.net/?p=542.

Cruz, Tavarez Carvalho. 2009. El modelo Entidad-Relación. [En línea] 2009. http://civil.fe.up.pt/acruz/access/modeloER.htm..

Hibernate.org. 2012. Hibernate. [En línea] 2012. http://www.hibernate.org/.

Infante, Daniel. 2009. Herramientas de Modelado. [En línea] 2009. http://www.geocities.ws/daniel.infante/fase2/t1.html.

King, G. 2006. Hibernate Reference Documentation 3.2.2. [En línea] 2006. http://www.hibernate.org/hib_docs/v3/reference/en/pdf/hibernate_reference.pdf.

Mato, Rosa María García. 1999.*Diseño de Bases de Datos.* 1999.

Microsoft. 2011. Microsoft Open Database Connectivity (ODBC). [En línea] 2011. http://msdn.microsoft.com/en.../ms710252(v=vs.85).aspx..

Netbeans.org. 2012. Netbeans. [En línea] 2012. http://netbeans.org/index_es.html.

Nguèn, M. H. 2008.*Data Abstraction Layer: Independet for Database.* 2008.

Pao, Angie. 2010. [En línea] 2010. http://www.gratisblog.com/all_the_drama/i147406-base_de_datos_iv._modelos_de_datos.htm.

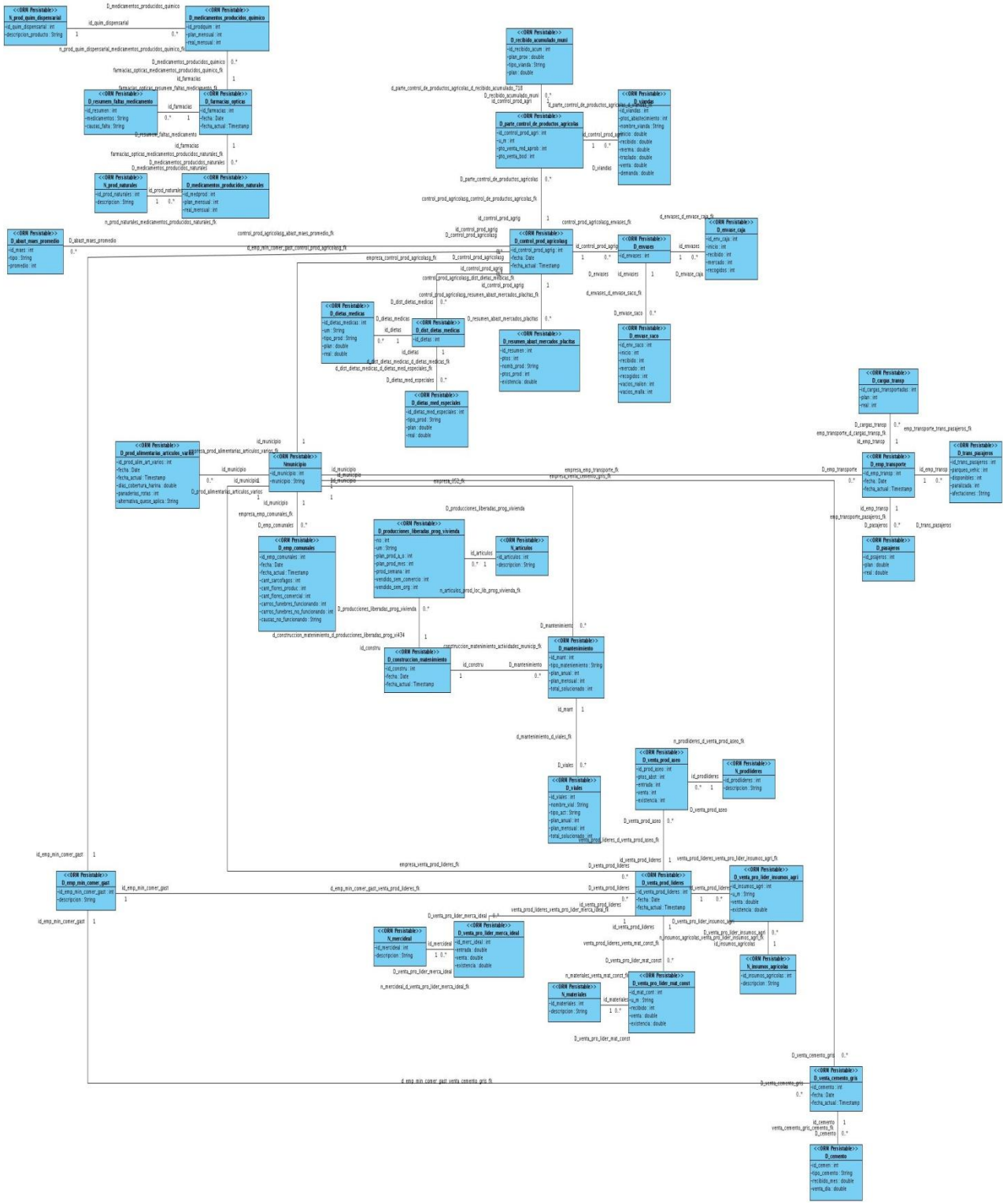
Rodríguez, Rubén. 2004. Bases De Datos (La Historia). [En línea] 2 de 1 de 2004. http://www.fudim.org/comunicacion/notas/nota.php?id=22&a=Adim..

Rozic, Sergio Ezequiel. 2004. Base de Datos. [En línea] 2004. http://www.recol.es/index.php?option=com_content&task=view&id=110&Itemid=417.

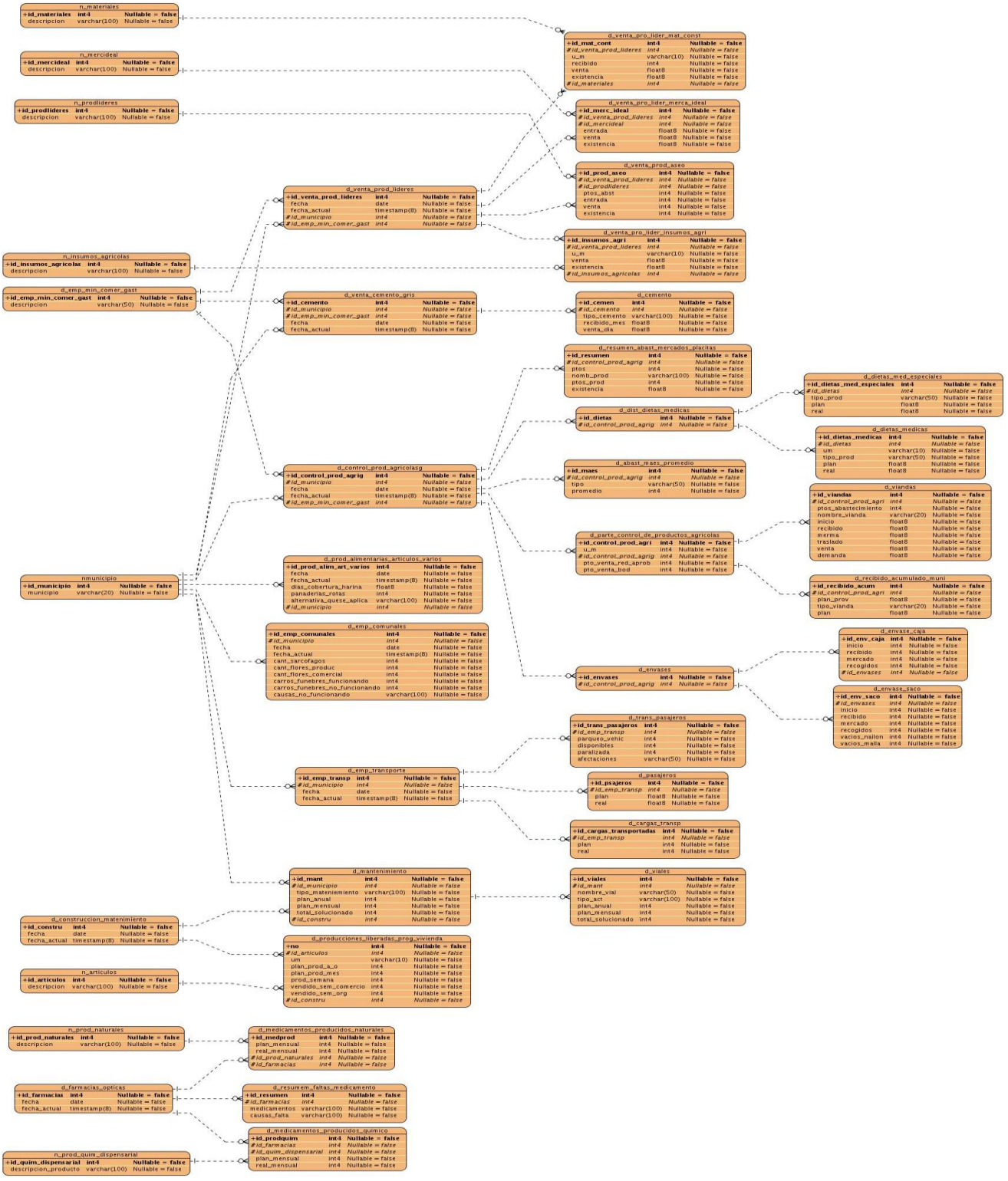
Schmieg, Sebastian. 2008. Fundamentación Informáticos. [En línea] 2008. http://fundamentosinformaticosjl.wordpress.com/category/base-de-datos/.

Thelin, J. 2007.*Foundations of QT Development.* 2007.

Anexo 1. Diagrama de clases persistentes



Anexo 2. Diagrama de Entidad-Relación



Glosario de Términos

ACID: es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español. Es un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción.

ETL: Extract, Transform and Load (*Extraer, transformar y cargar* en inglés, frecuentemente abreviado a ETL) es el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos y limpiarlos, y cargarlos en otra base de datos, data mart, o data warehouse para analizar, o en otro sistema operacional para apoyar un proceso de negocio.

IDE (en inglés integrated development environment, entorno de desarrollo integrado) es un programa informático compuesto por un conjunto de herramientas de programación

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

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

Persistencia en informática de modo genérico, se refiere a la propiedad de los datos para que estos sobrevivan de alguna manera.

Recuperación: Adquisición de una cosa que antes se tenía o lo que se había perdido.

Rollback es una operación que devuelve a la base de datos a algún estado previo. Los Rollbacks son importantes para la integridad de la base de datos, a causa de

que significan que la base de datos puede ser restaurada a una copia limpia incluso después de que se han realizado operaciones erróneas.

Savepoint (en español, punto de recuperación) es una forma de implementar subtransacciones dentro de un ORDBMS indicando un punto dentro de una transacción de BD que puede ser "rolled back" (devuelta) sin afectar a cualquier trabajo realizado en la transacción antes de que el punto de recuperación fuera creado

SQL (Structured Query Language) Es un lenguaje de acceso a las Base de Datos, permite especificar todas las operaciones sobre la base de datos como por ejemplo: Inserción, Borrado, Actualización. Utiliza características de álgebra y cálculo relacional permitiendo de esta forma realizar consultas a la base de datos de forma sencilla.