UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

FACULTAD 3

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Diseño de la base de datos para el módulo de IPC de la Oficina Nacional de Estadísticas.

Autores

William Bravo Caballero

Rodolfo Ávila Trujillo

Tutora

Lic. Arismayda Dorado Risco

Co-Tutora

MSc. Eugenia Genoveva Muñiz Lodos

Ciudad de La Habana. Junio de 2007

"Año 49 de la Revolución"

DECLARACION DE AUTORIA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad y a la Vicerrectoría de Formación de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año_____.

_____ _____

William Bravo Caballero Rodolfo Ávila Trujillo

_____

Lic. Arismayda Dorado Risco

Acumular información es sólo el primer paso hacia la sabiduría. Pero compartir información es el primer paso hacia la comunidad.

Henry Lewis Gates.

AGRADECIMIENTOS

A la profesora Eugenia M. Lodos. ¿Qué nos hubiéramos hecho sin usted…?

A Arismayda Dorado Risco nuestra tutora, por guiarnos, por tanto apoyo y acertados consejos.

A Leonel Jesús del Castillo por la invaluable ayuda que nos brindaste y su empeño para que esto saliera bien.

A los especialistas del Módulo IPC de la Oficina Nacional de Estadísticas por la confianza depositada y el entusiasmo.

A nuestros compañeros por apoyarnos…

DEDICATORIA

De William:

A mi mamá por su amor, comprensión, por darme fuerzas y apoyo en todo, por ser mi guía y enseñarme el sacrificio, que es el verdadero camino hacia el futuro.

A mi familia por darme tanto apoyo, cariño y confianza...

A mi querida Kenia, por ser tan especial con su presencia en mi vida y por su amor.

A mi papá, por su apoyo y cariño.

A Rosendo, por su amistad, tan firme y fuerte...

A todos mis amigos por estar cuando los necesité.

De Rodolfo:

A todas aquellas personas que contribuyeron en mi desarrollo tanto profesional como personal.

A mis padres y hermanos que siempre me apoyaron y fueron mi inspiración.

A mis amigos por compartir y apoyarme en los momentos buenos y malos.

De ambos:

A nuestro Comandante en Jefe Fidel Castro Ruz, por habernos dado la posibilidad de estudiar en esta universidad de excelencia.

A todos los profesores de la universidad que aportaron su granito de arena en nuestra formación profesional.

A la tutora Lic. Arismayda D. Risco por su paciencia, dedicación y aporte.
A la MSc. Eugenia G. Muñiz Lodos por sus consejos constructivos.

RESUMEN

Con el avance actual de las tecnologías en diversas esferas de la sociedad, las empresas han optado por automatizar sus actividades, para lograr realizarlas en menor tiempo y con una mayor productividad. Cuba no ha sido una excepción en este sentido. La Oficina Nacional de Estadísticas (ONE) ha solicitado a la Universidad de las Ciencias Informáticas el desarrollo de un sistema que reemplace uno existente.

La ONE, con su sede en Ciudad de La Habana, es la entidad encargada de archivar y procesar gran cantidad de información de muchas esferas de nuestra sociedad. Dentro de las disímiles operaciones que se procesan se encuentra el Cálculo de Índices de Precios al Consumidor (IPC), que es de extrema importancia para analizar el comportamiento de varios indicadores económicos y sociales del país.

El presente trabajo de diploma, tiene como principal objetivo implementar el diseño de una base de datos que almacene la información que procesará el sistema informático para calcular el IPC para la ONE, a partir del procesamiento de ficheros, con vistas a garantizar almacenamiento, disponibilidad, integridad, autenticidad y consistencia de la información.

El desarrollo de dicha base de datos estuvo guiado por las especificaciones que se proponen en los objetivos específicos y las herramientas que son utilizadas tales como el sistema gestor de base de datos y la implementación de la capa de acceso a datos, cuya selección fue resultado de un estudio comparativo entre las tendencias y tecnologías actuales, dando una medida a la solución propuesta.

Palabras claves.

Diseño, base de datos (BD), capa de acceso a datos (CAD), modelo lógico y físico, clases, herramienta, Microsoft SQL Server, TierDeveloper, Sistema Gestor de Base de Datos (SGBD), Índice de precios del Consumidor (IPC), Oficina Nacional de Estadísticas (ONE).

INDICE

INDICE DE FIGURAS

Capitulo 2

INDICE DE TABLAS

Capítulo 2

Capítulo 3

INTRODUCCION

En la actualidad y como consecuencia del desarrollo alcanzado por las tecnologías de la información y las comunicaciones (TICs), la automatización de los procesos que se realizan en las empresas es cada vez mayor, permitiendo realizar dichas actividades en menor tiempo de desarrollo y aportando mayor productividad, Cuba no esta ajena a estos cambios por lo que en los últimos años se ha impulsado el desarrollo de la informática.

La Universidad de las Ciencias Informáticas (UCI) surge como parte de ese esfuerzo que realiza el país para lograr la informatización de la sociedad; en dicha institución existe una fuerte infraestructura tecnológica, informática y telemática para la formación de profesionales altamente calificados y la producción de aplicaciones informáticas.

Actualmente la Oficina Nacional de Estadísticas se encuentra en la necesidad de mejorar las prestaciones del sistema de gestión existente, con vistas a responder eficazmente al desarrollo vertiginoso que ha tenido la tecnología en los últimos tiempos y una mejor toma de decisiones.
Además de realizar otras funcionalidades dicha entidad, efectúa el cálculo del Índice de Precios del Consumidor (IPC) que es específicamente, una medida estadística de la evolución de los precios de los productos y servicios que consumen los hogares cubanos. El conjunto de productos y servicios se obtiene básicamente del consumo de las familias, y la importancia de cada uno de ellos en el cálculo del IPC esta determinado por dicho consumo

En la ONE la gestión de la información se realiza, en su mayoría, de forma manual. Esto conlleva a que ocurran errores muchas veces no detectables. El sistema informático utilizado presenta insuficiencia en un gran número de funcionalidades, además el almacenamiento de los datos no responde a las necesidades de dicha entidad, dada la rigidez del mismo, lo cual no permite hacer modificaciones. Esta situación impide que se garanticen todas las necesidades de almacenamiento, disponibilidad, integridad, control, autenticidad y consistencia de la información.

Para dar respuesta a dicha problemática se creó un proyecto productivo en la Facultad 3 con el objetivo de implementar un sistema informático para el módulo IPC de la Oficina Nacional de Estadísticas que

garantice la gestión automatizada de los procesos que ocurren en el mismo.
El problema científico de la presente investigación es que no existe una base de datos que responda a las necesidades de almacenamiento de datos del Módulo IPC en la ONE.

El objeto de estudio de este trabajo es el almacenamiento de la información a través de una base de datos, y el campo de acción se enmarca en almacenamiento de la información en el proceso de desarrollo de software.

El objetivo general de este trabajo diseñar la base de datos del Módulo IPC, y para lograrlo fueron trazados los siguientes objetivos específicos:

1. Realizar un estudio de las herramientas gestoras de bases de datos y generadoras de la capa de acceso a datos.

2. Diseñar la BD para el módulo Índice de Precios del Consumidor.

3. Implementar el Acceso a Datos para el módulo Índice de Precios del Consumidor.

Para darle cumplimiento a los objetivos será necesario realizar una serie de actividades:

- Analizar los requisitos funcionales y no funcionales en un marco contextual para el módulo IPC de ONE.
- Instalar los softwares necesarios.
- Confeccionar los modelos lógico y físico de la BD.
- Realizar la validación teórica del diseño, a través de:
    - Normalización de la Base de datos.
    - Análisis de redundancia de información.

- Realizar la validación funcional del diseño, a través de:
  - Llenado automático y voluminoso de las tablas de la BD.
  - Transacciones de los datos.
  - Pruebas de cargas a la BD.
- Implementar la BD con las tablas, triggers, funciones, vistas, usuarios y demás objetos que se necesite para cumplir con los requisitos.
- Implementar la capa de acceso a los datos (CAD).
- Analizar y valorar los resultados obtenidos según la implementación de la BD.

A partir de los objetivos trazados se considera que si se diseña una BD entonces se podrá garantizar la gestión automática de los datos en el sistema de gestión del Módulo IPC de la ONE.

Métodos de investigación

Para llevar a cabo esta investigación se utilizaron los siguientes métodos:

Métodos teóricos:

- Método Análisis histórico – lógico. Para llevar a cabo esta investigación es necesario desarrollar un estudio analítico detallado de la trayectoria histórica así como la evolución y desarrollo del sistema que existe en ONE y el efecto que ha proporcionado el mismo en la funcionalidad de dicha institución nacional.
- Método Analítico-Sintético – Fue necesario la investigación y estudio de documentos para extraer los elementos necesarios más importantes que se relacionan con la gestión de la información en ONE.
- Método de modelación. Para facilitar aun más el trabajo se crean modelos o estrategia para investigar la realidad de los procesos estadísticos que se llevan a cabo en ONE.

Métodos empíricos:

- Método de la observación. Visualidad de la investigación que ofrece la posibilidad de poder efectuar un análisis anticipado de la información, también se puede verificar y comprobar varias concepciones teóricas agilizando la obtención de las principales características del objeto de estudio definido en la investigación.

- Método de la entrevista. Se hace necesario realizar entrevistas para tener un intercambio con las personas que conforman la unidad de estudio, en este caso el personal que labora en el modulo de IPC de ONE, para de esta forma obtener conocimientos de cómo se desarrollan todos los procesos que se llevan a cabo en esta oficina y poder determinar los requisitos necesarios para el buen desempeño de este trabajo y así por mediación de este método se puede obtener información verídica de la situación así como de las necesidades reales de la misma.

Resultados esperados

Con este trabajo se pretende lograr el diseño e implementación de una base de datos en Microsoft SQL Server 2000 que permita la automatización de la gestión de toda la información que se maneja en la ONE de forma eficaz.

EL trabajo está estructurado en tres capítulos. El Capítulo 1 se aborda la fundamentación teórica de la investigación. En este se incluye el estado del arte del tema base de datos a nivel internacional, nacional y de la Universidad. Las tendencias, técnicas, tecnologías relacionadas con este tema en la actualidad y que son la base para la solución del problema.

En el Capítulo 2 se aborda la descripción y análisis de la solución propuesta, en el mismo se exponen los aspectos del diseño de la BD: los diagramas relacionales y la descripción de las entidades correspondientes. Además, se explican los aspectos relacionados a la implementación de la capa de acceso de datos a través de diagramas y descripciones de las clases necesarias para el acceso.

Finalmente, el Capítulo 3 se describe las actividades de validación teórica y funcional del diseño realizado. También se realiza una valoración de los resultados obtenidos. En las páginas finales de la tesis se encuentran las secciones de las conclusiones, bibliografía, anexos y glosario de términos.

CAPITULO 1 FUNDAMENTACION TEORICA

Introducción

En este capítulo se exponen los principales fundamentos teóricos en los que se basa la presente investigación y se hace un análisis de las tecnologías y tendencias actuales, específicamente de los sistemas gestores de bases de datos.

1.1 Conceptos de Bases de Datos

Una Base de Datos (*BD)* es un conjunto de datos relacionados entre sí. Se entiende por *dato,* los hechos conocidos, que pueden registrarse y que tienen significado implícito. Ejemplo: Nombre del Producto, Código del Producto. Según esta definición el conjunto de palabras que conforman un libro es una BD pues estarían todas relacionadas entre sí. Una definición más rigurosa que la anterior es la siguiente:
Una *BD* es un conjunto de datos que tiene las siguientes propiedades implícitas:

- Representa algún aspecto del mundo real, llamado minimundo o universo de discurso. Las modificaciones del minimundo se reflejan en la BD.
- Es un conjunto de datos lógicamente coherentes, con un cierto significado inherente. Una colección aleatoria de datos no puede considerarse propiamente una BD.
- Una BD se diseña, construye y puebla con datos para propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a distintos usuarios.

O sea, una BD tiene:

- Una fuente de la cual se derivan los datos.
- Cierto grado de interacción con los hechos del mundo real.
- Un público activamente interesado en el contenido de la BD.
- Su tamaño es variado.
- Debe ser posible buscar, obtener y actualizar los datos siempre que sea necesario.[1]

[1] J., D. C. "Introducción a los sistemas de bases de datos", Capítulos 1, 2, 13, Editorial Félix Varela, La Habana, 2003. [15 de febrero del 2007]. Disponible en: http://teleformacion.uci.cu/mod/resource/view.php?id=13590.

1.2 Tipos de bases de datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido:

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, entre otras.

Según el contenido

- **Bases de datos bibliográficas:**

Solo contienen un representante de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque sino se estaría en presencia de una base de datos a texto completo (o de fuentes primarias) [ver más abajo]. Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

- **Bases de datos de texto completo:**

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

Directorios:

Un ejemplo son las guías telefónicas en formato electrónico.

- **Bases de datos o "bibliotecas" de información Biológica:**

Son bases de datos que almacenan diferentes tipos de información proveniente de las ciencias de la vida o médicas. Se pueden considerar en varios subtipos:

- Aquellas que almacenan secuencias de nucleótidos o proteínas.
- Las bases de datos de rutas metabólicas.
- Bases de datos de estructura, comprende los registros de datos experimentales sobre estructuras 3D de biomoléculas.
- Bases de datos clínicas.
- Bases de datos bibliográficas (biológicas)

1.3 Modelos de bases de datos

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Un modelo de datos es básicamente una "descripción" de algo conocido como *contenedor de datos* (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de *base de datos*; por lo general se refieren a algoritmos, y conceptos matemáticos.

Algunos modelos con frecuencia utilizados en las bases de datos:

- **Bases de datos jerárquicas:**

Estas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un *nodo padre* de información puede tener varios *hijos*. El nodo que no tiene padres es llamado *raíz*, y a los nodos que no tienen hijos se los conoce como *hojas*.

Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento.

Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

- **Base de datos de red:**

Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de *nodo*: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).
Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

- **Base de datos relacional:**
  - ***Modelo relacional***

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por *registros* (las filas de una tabla), que representarían las tuplas, y *campos* (las columnas de una tabla).
En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es *Structured Query Language* o *Lenguaje Estructurado de Consultas* (SQL), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.
Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Durante los años '80 (1980-1989) la aparición de dBASE produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que dBase no utilizaba SQL como lenguaje base para su gestión.

Bases de datos orientadas a objetos:

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los *objetos* completos (estado y comportamiento).
Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

- **Encapsulamiento** - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- **Herencia** - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- **Polimorfismo** - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.
Se está trabajando en SQL3**,** que es el estándar de SQL92 ampliado, que soportará los nuevos conceptos orientados a objetos y mantendría compatibilidad con SQL92.

Bases de datos documentales:

Permiten la indexación a texto completo, y en líneas generales realizar búsquedas más potentes.

Tesaurus es un sistema de índices optimizado para este tipo de bases de datos.

- **Base de datos deductivas:**

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

- **Bases de datos Objeto-Relacionales.**

Los modelos de datos relacionales orientados a objetos extienden el modelo de datos relacional proporcionando un sistema de tipos más rico y que se adapte mejor a las nuevas aplicaciones, que requieren guardar un tipo de datos más complejos. Permiten que los atributos de las tuplas tengan tipos complejos. Además, la base de datos es relacional por lo que conserva su rapidez y eficiencia, y permite hacer uso de nuevos elementos, como las relaciones de herencia, que modelan los objetos que serán guardados en dicha BD, por lo que el analista y el diseñador ven un modelo orientado a objetos.

- **Bases de datos distribuidas:**

La base de datos está almacenada en varias computadoras conectadas en red. Surgen debido a la existencia física de organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a distintas universidades, sucursales de tiendas, etcétera. [2]

1.4 Análisis y descripción de los diversos Sistemas Gestores de Bases de Datos.

Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. En la actualidad existe una gran variedad de SGBD, tienen presente que las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de

[2] La Enciclopedia Libre. Bases de Datos. Wikipedia, [17 de febrero del 2007]. Disponible en: http://es.wikipedia.org/wiki/Base_de_datos.

datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad.[3]

Principales objetivos de un SGBD:

- **Independencia de los datos y los programas de aplicación**

Ya se pudo notar que con archivos tradicionales la lógica de la aplicación contempla la organización de los archivos y el método de acceso. Por ejemplo, si por razones de eficiencia se utiliza un archivo secuencial indexado, el programa de aplicación debe considerar la existencia de los índices y la secuencia del archivo. Entonces es imposible modificar la estructura de almacenamiento o la estrategia de acceso sin afectar el programa de aplicación (naturalmente, lo que se afecta en el programa son las partes de éste que tratan los archivos, lo que es ajeno al problema real que el programa de aplicación necesita resolver). En un SBD sería indeseable la existencia de aplicaciones y datos dependientes entre sí, por dos razones fundamentales:

- Diferentes aplicaciones necesitarán diferentes aspectos de los mismos datos (por ejemplo, puede requerirse la representación decimal o binaria).

- Se debe poder modificar la estructura de almacenamiento o el método de acceso según los cambios en el fenómeno o proceso de la realidad sin necesidad de modificar los programas de aplicación (también para buscar mayor eficiencia).

La independencia de los datos se define como la inmunidad de las aplicaciones a los cambios en la estructura de almacenamiento y en la estrategia de acceso y constituye el objetivo fundamental de los SBD.

[3] KRONOS, J. T. Introducción a la Documática. Los sistemas de bases de datos y los SGBD., 5 de marzo del 2007]. Disponible en: http://tramullas.com/documatica/2-4.html

Minimización de la redundancia

Con los ficheros tradicionales, se produce redundancia de la información. Uno de los objetivos de los SBD es minimizar la redundancia de los datos. Se dice disminuir la redundancia, no eliminarla, pues, aunque se definen las BD como no redundantes, en realidad existe redundancia en un grado no significativo para disminuir el tiempo de acceso a los datos o para simplificar el método de direccionado. Lo que se trata de lograr es la eliminación de la redundancia superflua.

Integración y sincronización de las bases de datos

La integración consiste en garantizar una respuesta a los requerimientos de diferentes aspectos de los mismos datos por diferentes usuarios, de forma que, aunque el sistema almacene la información con cierta estructura y cierto tipo de representación, debe garantizar entregar al programa de aplicación los datos que solicita y en la forma en que lo solicita.
Está vinculada a la sincronización, que consiste en la necesidad de garantizar el acceso múltiple y simultáneo a la BD, de modo que los datos puedan ser compartidos por diferentes usuarios a la vez. Están relacionadas, ya que lo usual es que diferentes usuarios trabajen con diferentes enfoques y requieran los mismos datos, pero desde diferentes puntos de vista.

Integridad de los datos

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

Seguridad y recuperación

Seguridad (también llamada protección): garantizar el acceso autorizado a los datos, de forma de interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intención.

Recuperación: que el sistema de bases de datos disponga de métodos que garanticen la restauración de las bases de datos al producirse alguna falla técnica, interrupción de la energía eléctrica, etc.

Facilidad de manipulación de la información

Los usuarios de una BD pueden acceder a la misma con solicitudes para resolver muchos problemas diferentes. El SBD debe contar con la capacidad de una búsqueda rápida por diferentes criterios, permitir que los usuarios planteen sus demandas de una forma simple, aislándolo de las complejidades del tratamiento de los archivos y del direccionamiento de los datos. Los SBD actuales brindan lenguajes de alto nivel con diferentes grados de facilidad para el usuario no programador que garantizan este objetivo, los llamados sublenguajes de datos.

Control centralizado

Uno de los objetivos más importantes de los SBD es garantizar el control centralizado de la información. Permite controlar de manera sistemática y única los datos que se almacenan en la BD, así como el acceso a ella.

Lo anterior implica que debe existir una persona o conjunto de personas que tenga la responsabilidad de los datos operacionales: el administrador de la BD, que puede considerarse parte integrante del SBD y cuyas funciones se abordarán en la siguiente sesión.

Existen otros objetivos que deben cumplir los SBD que en muchos casos dependen de las condiciones o requerimientos específicos de utilización del sistema.[4]

Los sistemas gestores de bases de datos, pueden definirse como un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos a los datos y actuando de interfaz entre los datos físicos, el usuario y la aplicación que lo utiliza; ejemplos de estos, se explicaran a continuación.

Dentro de las principales funciones que debe cumplir un sistema gestor de bases de datos, se relacionan

[4] MATOS, R. M. Sistemas de Bases de Datos., (22 de febrero del 2007)

con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad.[5]

1.4.1 Oracle.

Es un sistema de gestión de base de datos fabricado por Oracle Corporation.

Se considera a Oracle como uno de los sistemas de bases de datos más completos que existe.

Ventajas:

- Soporte de transacciones.
- Gran estabilidad y seguridad.
- Escalabilidad.
- Es multiplataforma.

Inconveniente:

- Su mayor defecto es su enorme precio, que es de varios miles de euros (según versiones y licencias).
- Otro aspecto a criticar es la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque ha tenido un amplio dominio en el mercado de servidores sufre la competencia del Microsoft SQL Server de Microsoft y con licencia libre como PostgreSQL, MySql o Firebird.[6]

1.4.2 MySQL.

Es uno de los Sistemas Gestores de bases de Datos más populares desarrollados bajo la filosofía de código abierto. MySQL tiene como una de sus principales ventajas la velocidad en la lectura de datos, pero a costa de eliminar un conjunto de facilidades que presentan otros SGBD: integridad referencial, bloqueo de registros, procedimientos almacenados, entre otros. En recientes versiones de MySQL se

[5] Obra Citada [3]

[6] Wikipedia. La Enciclopedia Libre. Oracle. . [12 de marzo del 2007]. Disponible en: http://es.wikipedia.org/wiki/Oracle

incluyen algunas de estas características, pero indudablemente esto va en detrimento de la velocidad, como se expresa anteriormente es un SGBD basado en Open Source (Código abierto) diseñado para los sistemas Unix, aunque existen versiones para Windows.

Ventajas:

- Diseñado en vistas a la velocidad.
- Consume muy pocos recursos de CPU y memoria. Muy buen rendimiento.
- Tamaño del registro sin límite.
- Buena integración con PHP.
- Utilidades de administración (phpMyAdmin).
- Buen control de acceso usuarios-tablas-permisos.
- Trabaja bajo diferentes plataformas: AIX 4x 5x, Amiga, BSDI, Digital Unix 4x, FreeBSD 2x 3x 4x, HP-UX 10.20 11x, Linux 2x, Mac OS, NetBSD, Novell NetWare 6.0, OpenBSD 2.5, OS/2, SCO OpenServer, SCO UnixWare 7.1.x, SGI Irix 6.x, Solaris 2.5, SunOS 4.x, Tru64 Unix y Windows 9x, Me, NT, 2000,XP, 2003.[7]
- Desarrollo de APIs para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.[8]

Inconvenientes:

- No soporta subconsultas.
- No soporta transacciones.
- No soporta vistas.
- Se hace inestable cuando contiene gran cantidad de datos.[9]

[7] PABON., M. A. G. C. W. R. "COMPARACION ENTRE SISTEMAS DE GESTION DE BASES DE DATOS (SGBD) BAJO LICENCIAMIENTO LIBRE Y COMERCIAL", UNIVERSIDAD CATOLICA DE COLOMBIA FACULTA DE INGENIERIA DE SISTEMAS BOGOTA 2005, [1 de marzo del 2007]. Disponible en: http://www.ilustrados.com/documentos/sgbd.pdf

[8] Obra Citada [7]

[9] Wikipedia. La Enciclopedia Libre. MySQL. . [14 de marzo del 2007]. Disponible en: http://es.wikipedia.org/wiki/MySQL

1.4.3 PostgreSQL.

Está considerado el SGBD de código abierto más avanzado del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales de alto calibre tales como Oracle.

Es un SGBD objeto-relacional, ya que aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Su avanzada funcionalidad se pone de manifiesto con las consultas SQL declarativas, el control de concurrencia multiversión, soporte multiusuario, transacciones, optimización de consultas, herencia y valores no atómicos (atributos basados en vectores y conjuntos).

Es altamente extensible: soporta operadores y tipos de datos definidos por el usuario. Soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92. Cuenta con una API (del inglés Application Program Interface) flexible lo cual ha permitido dar soporte para el desarrollo con PostgreSQL en diversos lenguajes de programación.

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

Otras características

- Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (*foreign keys*).
- Disparadores (*triggers*).
- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.

Funciones

Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientación a objetos o la programación funcional.

Los disparadores (triggers en inglés) son funciones enlazadas a operaciones sobre los datos.

Algunos de los lenguajes que se pueden usar son los siguientes:

- Un lenguaje propio llamado PL/pgSQL (similar al PL/SQL de oracle).
- C.
- C++.
- Java (via PL/Java).
- PL/Perl.
- Pl/PHP.
- PL/Python.
- PL/Ruby.
- PL/sh.
- PL/Tcl.
- PL/Scheme. PL/Scheme.
- Lenguaje para aplicaciones estadísticas R through PL/R.

PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de

valores que pueden ser tratados igual a una fila retornada por una consulta en SQL (query en inglés)[10]

Ventajas con respecto a otros Sistemas gestores de base de datos.

- COSTO.
- Estabilidad, Confiabilidad.
- Funcionalidad (es un motor de bases de datos *Avanzado y* con herramientas gráficas como PgAdmin, phpPgAdmin la administración de la base de datos es sencilla).[11]

1.4.4 Microsoft SQL Server.

Es un sistema de gestión de bases de datos relacionales desarrollado por Microsoft. Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para la mayoría de las plataformas de desarrollo, incluyendo .NET, además de ser uno de los mejores sistemas de gestión de bases de datos relacionales (SGBD) basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Las necesidades y requerimientos de los clientes han llevado a la creación de innovaciones de producto significativas para facilitar la utilización, escalabilidad, confiabilidad y almacenamiento de datos.

Ventajas:

- Soporta la configuración automática y la auto-optimización.
- Administración multiservidor para un gran número de servidores.
- Gran variedad de opciones de duplicación de cualquier base de datos.
- Acceso universal a los datos (Universal Data Access).
- Fácil de usar.
- Escalabilidad. Permite realizar un escalamiento hasta 32 CPU y 64 gigabytes (GB) de RAM, siendo

[10] Wikipedia. La Enciclopedia Libre. PostgreSQL. [16 de marzo del 2007]. Disponible en: http://es.wikipedia.org/wiki/PostgreSQL

[11] KUROKI, C. PostgreSQL 8. Migración a PostgreSQL desde otras bases de datos, 2005. [15 de marzo del 2007]. Disponible en: http://www.dbrunas.com.ar/postgres/migrapg.pdf.

capaz de manejar al máximo multiprocesamiento simétrico aprovechando al máximo el hardware.[12]

- Potencia: Microsoft SQL Server es la mejor base de datos para Windows NT Server.
- Posee los mejores registros de los benchmarks independientes (TCP) tanto en transacciones totales como en coste por transacción.
- Gestión: Con una completa interfaz gráfica que reduce la complejidad innecesaria de las tareas de administración y gestión de la base de datos.
- Múltiples instancias y Failover.
- Integración con la Web
  Acceso Web a datos
- Soporte para XML (cláusula FOR XML)

Desventajas:

- Licencias con costos altos.
- Plataformas Windows[13]

1.5 Análisis y descripción de herramientas para los diagramas relacionales.

1.5.1 ER/Studio 7.0

Es una herramienta de diseño UML, de modelado de datos fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico se utiliza para realizar el modelado para el diseño y creación de bases de datos lógicas y físicas. Permite crear y mantener aplicaciones de bases de datos. ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes.

[12] Obra Citada [7]

[13] Wikipedia. La Enciclopedia Libre. SQL Server. [19 de marzo del 2007]. Disponible en: http://es.wikipedia.org/wiki/SQLServer

Las capacidades de diseño que contiene, ayudan a crear un diseño lógico que puede transformarse en cualquier número de diseños físicos. Como resultado, se puede mantener un diseño lógico normalizado mientras se no se normalizan los diseños físicos para su desempeño. ER/Studio mantiene ligas entre todos los niveles de su diseño por lo tanto puede mezclar cambios en cualquier dirección entre ellos. Revisa la normalización y la compilación con la sintaxis de la plataforma de la base de datos.

ER/Studio permite tomar por omisión las opciones para todos los diagramas así como realizar cambios al momento de la ejecución. Ayuda a prolongar la inversión que se ha hecho. Soporta el proceso de diseño interactivo inherente en el ciclo de vida de la aplicación[14]

1.5.2 CASE Studio 2 2.18

CASE Studio es una herramienta profesional con la que se pueden diseñar bases de datos, la cual facilita la creación de diagramas de relación, modelado de datos y gestión de estructuras. Tiene soporte para trabajar con una amplia variedad de formatos de base de datos (Oracle, SQL, MySQL, PostgreSQL, Access, etc.) y te permite además generar scripts SQL, aplicar procesos de retroingeniería (*reverse engineering*) a las bases de datos, usar plantillas de diseño personalizables y crear detallados informes en HTML y RTF.

A través de los diagramas de relación se podrá tener una visión más clara del contenido y una estructura de la base de datos, facilitando la gestión y mantenimiento de la misma[15]

1.6 Análisis y descripción de herramientas para la implementación de la CAD

1.6.1 TierDeveloper

TierDeveloper es una herramienta de mapeo a objetos relacionales y de generación de código, que genera negocios y objetos de datos en .NET, aplicaciones en ASP.NET y aplicaciones en Formularios

[14] ALVARADO, P. Modelo de base de datos con ER/Studio, 2007. [21 de marzo del 2007]. Disponible en: http://www.monografias.com/trabajos14/modelodebase/modelodebase.shtml

[15] CASE Studio 2 2.18. 23 de marzo del 2007]. Disponible en: http://software.elpais.com/ie/27636-CASE-Studio-2

Windows. Proporciona al usuario la oportunidad de definir Objetos de Datos que son mapeados a una o más tablas de la base de datos, que ayudan a extraer la información requerida, en el menor tiempo posible. Se pueden definir los Objetos de Datos, utilizando los esquemas existentes de la base de datos, generas código fuente de los Objetos de Datos, construir y desplegar los Objetos de Datos y probar los Objetos de Datos en la aplicación, la cual es generada sin gran esfuerzo.

Para ganar ventaja completa de TierDeveloper se requiere estar familiarizado con las tecnologías que incluyen:

- Componentes de Microsoft .NET.
- Microsoft Visual Studio .NET 2002/2003.
- Microsoft ADO.NET (acceso a la base de datos a través de OLEDB y proveedores gestionados para SQL Server 2000, Oracle y DB2).
- Aplicaciones Web en Microsoft ASP.NET.
- Aplicaciones en Formularios Windows.
- Servicios Web & Formularios Windows basados en clientes remotos.

Adicionalmente, TierDeveloper soporta los siguientes servidores de base de datos:

- Microsoft SQL Server 7.0/2000.
- Oradle GI/GI/Log.
- DBR 7.x/8.1.
- MS Access 2000 o versiones posteriores[16]

1.6.2 NHibernate

NHibernate es un mapeador objeto-relacional u ORM (por las siglas en ingles de object/relational mapping). Trabaja con software orientado a objetos y bases de datos relacionales puede incrementar el tiempo de desarrollo debido a la diferencia entre el modelo relacional y el paradigma orientado a objetos. Un ORM se encarga de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa. NHibernate no solo realiza esta transformación sino que

[16] TierDeveloper. [26 de marzo de 2007]. Disponible en: http://press.arrivenet.com/technology/article.php/493512.html

proporciona potentes capacidades para la obtención y almacenamiento de datos de la base de datos. Se utiliza muy ampliamente, se desarrolla activamente y soporta una extensa comunidad de Open Source, además tiene la ventaja de que es totalmente transparente el uso de la base de datos pudiendo cambiar de base de datos simplemente cambiando los ficheros de configuración, sin necesidad de cambiar una línea de código de la aplicación.

NHibernate surge del Framework Hibernate muy utilizado en java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones[17]

1.7 Análisis y descripción de la herramienta para el llenado automático de datos en las tablas de la base de datos.

1.7.1 EMS SQL Data Generator.

El generador de datos para Servidores SQL (EMS) es una potente herramienta para generar datos de pruebas para muchas tablas de bases de datos en Microsoft SQL a la vez. La aplicación permite definir tablas y campos para generar datos, rangos de grupos de valores, cargar valores de campos BLOB (Binary Large Object) desde los ficheros, obtener listas de valores de consultas en SQL y muchas otras características para generar datos de pruebas de una manera simple y directa. También proporciona una aplicación en consola, las cual permite generar datos a través del uso de plantillas de generación.

Características

- Interfaz de usuario amigable.
- Soporta todo tipo de datos en Servidores SQL, incluyendo UNIQUE, IDENTIFIER, BIT, SQL_VARIANT y TIMESTAMP.

[17] CABRERA, M. Introducción a NHibernate., Junio 2005. [30 de marzo del 2007]. Disponible en: http://mcabrera.datacenter1.com/articles/dotNET/nhibernate/

- Diversos tipos de generación para cada campo, incluyendo la generación de listas, datos aleatorios e incrementales, etc.
- Habilidad para usar los resultados de las consultas en SQL como listas de valores para la generación de datos.
- Control automático sobre la integridad referencial para la generación de datos en las tablas vinculadas.
- Gran variedad de parámetros de generación para cada tipo de campo.
- Habilidad para establecer valores nulos para un determinado por ciento de casos.
- Almacenar todos los grupos de parámetros de generación en la sección actual.
- Utilidad Command-line para generar datos usando la plantilla de los archivos. [18]

1.8 Justificación de la propuesta de solución.

Por todo expuesto anteriormente en este capítulo, para la elaboración de la propuesta de solución que plantea que se desarrollará una base de datos para el modulo de IPC, lo que permitirá su conexión al sistema para que cumpla todas sus funcionalidades.

Brevemente se expusieron algunas de las razones por las cuales se usaran tanto el Microsoft SQL Server 200 como gestor de bases de datos, el Framework de acceso a datos será Tier Developer por su familiaridad con la Arquitectura NET, para programar en C#, con un ambiente accesible al Servidor SQL Server 2000 y el Visual Studio.NET como plataforma, se tomo el C# como lenguaje de programación que ha demostrado ser una herramienta invaluable para desarrolladores en busca de afinar el comportamiento y desempeño de sus aplicaciones. Cabe destacar que el SQL Server ostenta marcas de referencia en cuanto escalabilidad y confiabilidad, que son críticas para el éxito de una base de datos empresarial. Tanto si lo que se mide es la velocidad en el desarrollo de aplicaciones como la velocidad del procesamiento de transacciones y por su gran integración con la plataforma Visual Studio.NET y por las ventajas que brinda

[18] EMS Data Generator for SQL Server. 2007. [23 de abril de 2007]. Disponible en: http://sqlmanager.net/products/mssql/datagenerator

para la construcción de la propuesta.

Se utilizará la herramienta ER/Studio por su probada eficacia en el modelado de los datos permitiendo realizar de forma sencilla el modelo lógico y físico de la base de datos, además de la utilización de MS SQL Data Generator, por su facilidad y rapidez en el llenado automático y voluminoso de los datos en las tablas de la base de datos.

1.9 Conclusiones.

En este capítulo se hizo un estudio de las tecnologías a utilizar en el desarrollo de la propuesta de solución así como algunos conceptos y tendencias que se deben tener en cuenta. Se fundamentó la selección de las herramientas escogidas para la solución.

Con las características expuestas sobre las tecnologías a usar se puede concluir que son parte de la revolución informática de la nueva generación de aplicaciones que trabajan en colaboración y el objetivo principal que se logra con el uso de las mismas es la interoperabilidad.

CAPITULO 2 DESCRIPCION Y ANALISIS DE LA SOLUCION PROPUESTA

Introducción

En el presente capítulo se diseña la propuesta de solución. En él se incluyen los requisitos no funcionales de la base de datos, los diseños lógico y físico, así como la descripción de las tablas y otros objetos de la BD.

2.1 Selección y argumentación de los requisitos funcionales y no funcionales del sistema propuesto.

En éste epígrafe se relaciona la lista de requerimientos funcionales y no funcionales del sistema a desarrollar.

2.1.1 Requisitos Funcionales:

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Para este sistema se define:

R1. Autenticar Usuario

1.1 Mostrar la opción de acceder al sistema, según los datos correspondientes (usuario y contraseña).

1.2 Introducir nombre de usuario y contraseña del sistema.

1.3 Mostrar al usuario las opciones a las que tiene acceso según el rol o permisos que posee.

R2. Fusionar Ficheros.

2.1 **Solicitud de ficheros provenientes de provincias.**

2.2 Fusiona todos los ficheros recibidos de provincias para cada tipo de mercado.

2.3 Seleccionado el año y mes que desea procesar.

2.4 Efectuar la fusión.

2.4.1 Mostrar si se ha fusionado los datos seleccionados (año, mes).

2.4.2 Se procesa y almacenan los datos en la base de datos.

R3. Emitir Fusión

3.1. **Solicitud de mostrar en pantalla los datos de la fusión realizada en una fecha y tipo de mercado dado.**

3.2. Seleccionar tipo de mercado y fecha.

3.3. Efectuar muestra.

3.3.1. De no existir esta información, mostrar en pantalla un mensaje: “No se ha procesado con la fecha seleccionada”.

3.3.2. Mostrar en forma de tablas los datos correspondiente a dicha información.

R4. Emitir Tablas de precios.

4.1 **Solicitud de la visualización de los indicadores de precios.**

4.2 Especificación del tipo de mercado, región, año y mes a procesar.

4.3 Efectuar visualización.

4.3.1 De no emitir los datos específicos, mostrar mensaje: “No se pueden mostrar dichos datos”.

4.3.2 Mostrar las tablas de los indicadores de precios.

4.4 Impresión de las tablas de los indicadores de precios.

R5. Emitir Tablas de índices.

5.1 **Solicita la visualización de las tablas de índices de precios.**

5.2 Especificación del mercado, región, año y mes

5.3 Efectuar visualización.

5.3.1 De no emitir los datos específicos, mostrar mensaje: “No se puede emitir con dichos datos”

5.3.2 Mostrar dichas tablas.

5.4 Impresión de las tablas de índices de precios.

R6. Gestionar Clasificador.

6.1 **Solicitud de Gestión Clasificador.**

6.2 **Solicitud de mostrar la lista de los productos que están registrados en clasificador.**

6.3 **Solicitud de “Insertar nuevo producto”**

6.3.1 Mostrar todos los datos de insertar en cuestión.

6.3.2 Introducir los datos del producto.

6.3.3 Efectuar la inserción de dicho producto.

6.3.3.1 Validar las entradas, principalmente (código y/o nombre) según la cantidad de caracteres y que no pueden estar estos campos vacíos, mostrar mensaje: “Verifique los datos entrados”.

6.3.3.2 Chequear que de dicho producto no este registrado en la base de datos.

6.3.3.2.1 De existir dicho producto según sus principales componentes (código y/o nombre), mostrar mensaje: “Este producto ya existe, verifique los datos de entrada”.

6.3.4 Mostrar mensaje de confirmación de la inserción.

6.4 **Solicitud de “Modificar producto”**

6.4.1 Mostrar los productos almacenados.

6.4.2 Seleccionar el producto a modificar.

6.4.3 Efectuar la modificación.

6.4.3.1 Mostrar todos los datos de modificar en cuestión.

6.4.3.2 Guardar los cambios pertinentes.

6.4.4 Mostrar mensaje de confirmación de la modificación.

6.5 **Solicitud de “Eliminar producto”**

6.5.1 Mostrar los productos almacenados.

6.5.2 Seleccionar el producto a eliminar.

6.5.3 Efectuar la eliminación.

6.5.3.1 Mostrar mensaje de advertencia: “¿Está seguro de eliminar este producto?”

6.5.4.1 De no eliminarlo (hacer referencia a 6.5.1)

6.5.4.2 Mostrar mensaje de confirmación de la eliminación.

R7. Gestionar Usuario.

7.1 **Solicitud de Gestión Usuario**

7.2 Solicitud de mostrar la lista de los usuarios registrados en el sistema

7.3 **Solicitud de “Agregar un usuario”.**

7.3.1 Mostrar todos los datos de agregar en cuestión.

7.3.2 Introducir los datos del usuario que se desea registrar.

7.3.3 Seleccionar un rol especifico al que pertenecerá (Operador o Administrador).

7.3.4 Efectuar la agregación.

7.3.4.1 De existir, mostrar mensaje:"El usuario ya existe".

7.3.4.2 Registrar un nuevo usuario al sistema.

7.4 **Solicitud de "Eliminar un usuario".**

7.4.1 Mostrar los usuarios del sistema

7.4.2 Seleccionar el usuario a eliminar.

7.4.3 Efectuar la eliminación.

7.4.3.1 Mostrar mensaje de advertencia: "¿Está seguro de eliminar este usuario?".

7.4.3.2 De no eliminarlo (hacer referencia a 7.4.1)

7.4.3.3 Eliminar dicho usuario.

7.5 **Solicitud de "Modificar información del usuario".**

7.5.1 Mostrar los usuarios del sistema.

7.5.2 Seleccionar un usuario.

7.5.3 Efectuar la modificación.

7.5.3.1 Modificar datos del usuario (nombre del usuario, datos personales, rol que juega).

7.5.3.2 Registrar los cambios realizados.

2.1.2 Requisitos no funcionales:

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Para definirlos se tienen en cuenta propiedades de factibilidad, usabilidad, confiabilidad, entre otros.

Requisitos de Interfaz o Apariencia

- Interfaz con un diseño sencillo que contenga pocos gráficos, con vista a acelerar la velocidad de respuesta hacia el usuario debido a la complejidad de los procesos llevados a cabo en el módulo de ONE.
- La interfaz debe limitarse a presentar las funcionalidades de IPC logrando la concentración del usuario en las tareas que esté realizando.

Requisitos de Usabilidad

- Documentar bien la aplicación y proporcionar materiales de ayuda para hacer mejor uso de todos los servicios que este ofrece.
- Se deberá reducir el tiempo de respuesta a las peticiones por parte de los usuarios a la BD.
- La BD tendrá gran disponibilidad y confiabilidad en cuánto a la información que se almacene.

Requisitos de Rendimiento

- El tiempo de respuesta de cada uno de las interfaces debe ser menor que un minuto, excepto en aquellas que por las actividades que realizan, requieran más tiempo.
- Estará implementado sobre la plataforma de desarrollo C# y se utilizará Microsoft SQL Server 2000 como Sistema Gestor de Bases de Datos.

Requisitos de Soporte

- Garantía de instalación y prueba del sistema, además de un breve entrenamiento a los futuros usuarios.
- Lograr la solidez de los datos realizando mantenimientos automatizados en la base de datos, orientados a la actualización y corrección de la información.
- El servidor de la BD estará montado en una Pentium(R) 4 con procesador de 256 MB de RAM como mínimo.
- Se trazará una estrategia para el mantenimiento de la BD como norma de soporte.

Requisitos de Seguridad

- Debe contar con varios niveles de acceso para permitir el trabajo organizado con el sistema.
- El acceso se realizará como un Subsistema de Seguridad, el mismo permitirá la gestión de usuarios y roles así como el acceso.
- Toda la información guardada será confidencial, por lo que solo podrá ser consultada por personal autorizado y estará prohibida su divulgación, ahí que se garantice de un tratamiento adecuado de las excepciones y validación de las entradas del usuario.
- La información almacenada o manejada desde la BD debe estar protegida de acceso no autorizado y divulgación.

- La BD será independiente de la aplicación.

2.2 Diseño de la Base de Datos

El diseño de la base de datos está dividido en dos módulos: Datos IPC y Seguridad. A continuación se realiza la descripción de dichos módulos.

2.2.1 Módulo de Datos IPC:

En este módulo se almacenan toda la información que proviene de las provincias, referente a un mercado determinado, con su información correspondiente.

- De cada producto se conoce su código (que la identifica), su nombre, una cota de precios según sus valores en monedas nacionales o convertibles, y una unidad de medida (que puede repetirse en diferentes productos), que dicho producto puede presentarse en los cuatros mercados, con un precio diferente, para algún caso, aunque en un mercado se puede presentar varias veces un mismo producto dependiendo de la provincia que se trate, teniendo en cuenta además una determinada fecha (año y mes).
- De cada provincia se conoce su código (que la identifica) y su nombre, además de que se posibilite añadir o eliminar una provincia.
- De cada mercado se conoce que presentan un identificador, que especifica un tipo de mercado, y una descripción de este; también resaltar la posibilidad de que permita añadir o eliminar un mercado determinado.

De acuerdo a lo expuesto anteriormente, se propone un diagrama de entidad relacional que es usado para modelar la estructura lógica de la base de datos, ver figura 2.1:

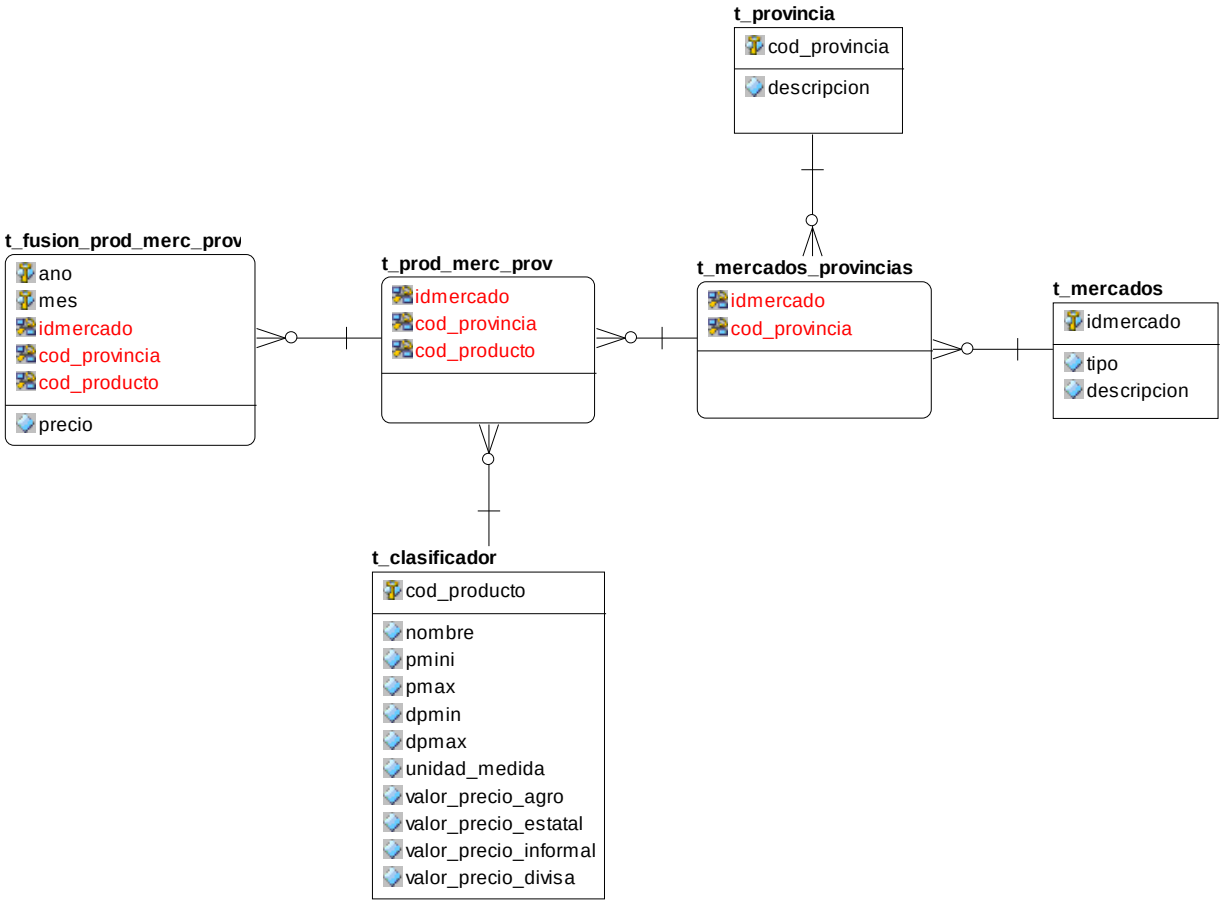


Fig.2.1. Modelo Lógico de Datos para el Módulo de Datos IPC.

Interrelaciones:

Para las interrelaciones del modelo presentado anteriormente cumplen primeramente:

- Cada producto se encontrará implicado en la encuesta.
- Dada una provincia determinada puede involucrarse en uno o más mercados
- Cada mercado tendrá una lista de productos.
- Para una fecha determinada, se emite el listado de todas las provincias con sus respectivos mercados y estos a su vez con sus respectivos productos.
- Dada una modificación o eliminación de un mercado o provincia o producto, posibilita que dichos datos sean modificados o eliminados en forma secuencial en todas las entidades que la compongan.

El diagrama de modelo de datos se corresponde con la representación física de la base de datos, como se muestra en la figura 2.2.

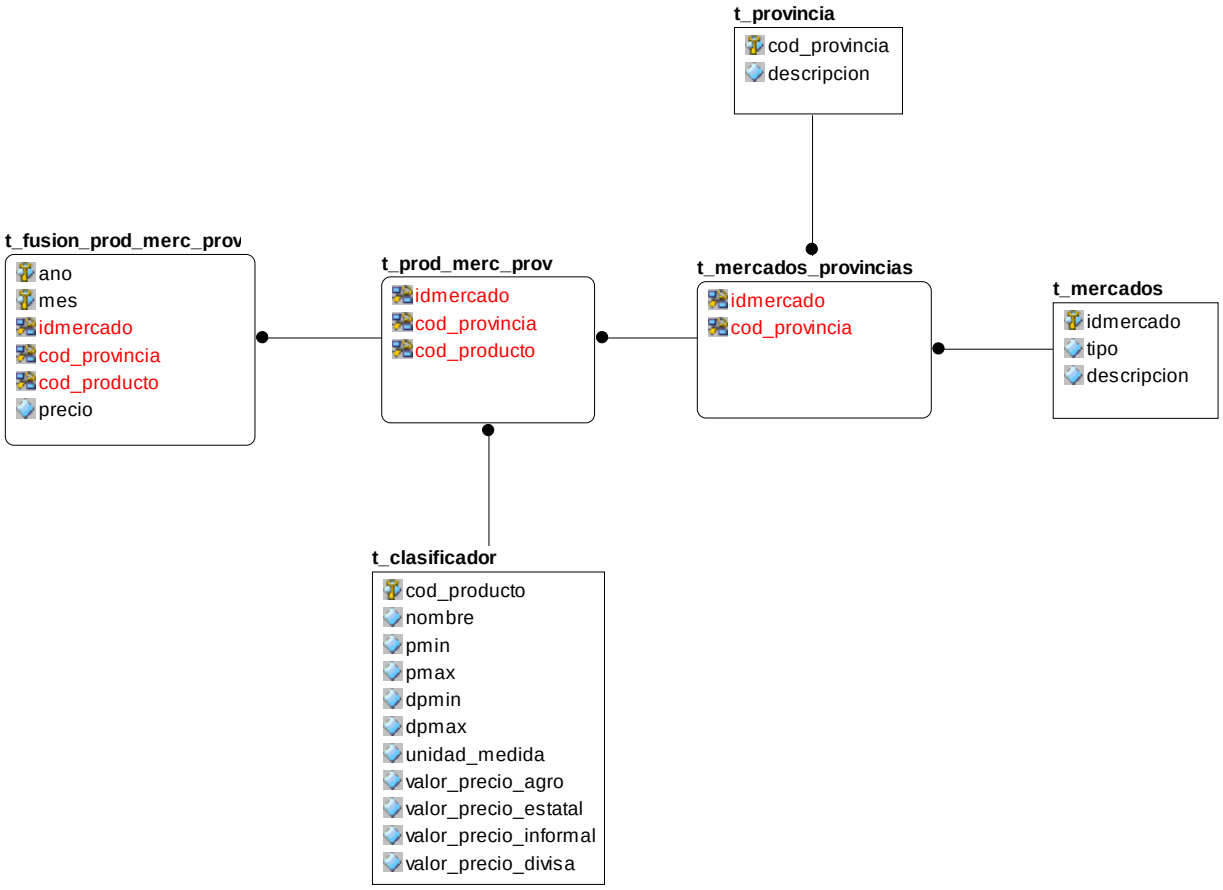


Fig.2.2. Modelo Lógico de Datos para el Módulo de Datos IPC.

2.2.1.1 Descripción de las tablas generadas:

A continuación se hará alusión a la descripción detallada de las tablas de la base de datos representadas para el Módulo de Datos de IPC, haciendo referencia primeramente al objetivo de dicha tabla, además de representar los atributos llaves o claves y no llave con el tipo de dato correspondiente y la declaración de dicho atributo que sea llave primaria o llave primaria compuesta o un valor no nulo.

Nombre: t_provincia				
Descripción: Almacena los datos de las provincias.				
Atributo	**Llave**	**Tipo**	**Requerido**	**Descripción**
cod_provincia	PK	Integer [2]	PK	Código del provincia
descripción		Char[50]	NOT NULL	Nombre de la provincia

Tabla 2.1. Descripción t_provincia.

Nombre: t_mercados				
Descripción: Almacena los datos de los mercados.				
Atributo	**Llave**	**Tipo**	**Requerido**	**Descripción**
idmercado	PK	Integer [4]	PK	Identificador de los tipos de mercados.
Tipo		Char[2]	NOT NULL	Descripción de los tipos de mercados.
descripción		Varchar[50]	NOT NULL	Descripción del mercado.

Tabla 2.2. Descripción de t_mercados.

Nombre: t_clasificador				
Descripción: Almacena los datos de los productos a los cuales se le hace el sondeo.				
Atributo	**Llave**	**Tipo**	**Requerido**	**Descripción**
cod_producto	PK	Char[8]	PK	Código del producto
nombre		Varchar[50]	NOT NULL	Nombre del producto
pmin		Decimal	NOT NULL	Precio Mínimo del producto (Moneda Nacional)
pmax		Decimal	NOT NULL	Precio Máximo del producto (Moneda Nacional)
dpmin		Decimal	NOT NULL	Precio Mínimo del producto (Moneda Libremente Convertible)
dpmax		Decimal	NOT NULL	Precio Máximo del producto (Moneda Libremente Convertible)
unidad medida		Char[15]	NOT NULL	Unidad de media del Producto(LIB ,UNO ,PAQ ,LATA ,LITRO ,KG RACION ,PAR ,SACO ,M3 ,CAJA ,GALON ,LIT ,FRASCO ,BULBO ,ROLLO ,BOT ,SOBRE ,JARRA)
valor_precio_agro		Decimal	NOT NULL	Valor inflado del precio para el mercado agropecuario
valor_precio_informal		Decimal	NOT NULL	Valor inflado del precio para el mercado informal
valor_precio_formal		Decimal	NOT NULL	Valor inflado del precio para el mercado formal
valor_precio_divisa		Decimal	NOT NULL	Valor inflado del precio para el mercado divisa.

Tabla 2.3. Descripción de t_clasificador.

Nombre: t_mercados_provincias				
Descripción: Recoge la información de los mercados por provincias.				
Atributo	**Llave**	**Tipo**	**Requerido**	**Descripción**
idmercado	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_mercados.
cod_provincia	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_provincia.

Tabla 2.4. Descripción de t_mercados_provincias.

Nombre: t_prod_merc_prov				
Descripción: Recoge la información de los mercados por provincias con sus productos.				
Atributo	**Llave**	**Tipo**	**Requerido**	**Descripción**
idmercado	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_mercados.
cod_provincia	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_provincia.
cod_producto	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_clasificador.

Tabla 2.5. Descripción de t_ prod_merc_prov.

Nombre: t_fusion_prod_merc_prov				
Descripción: Recoge la información de una fusión.				
Atributo	**Llave**	**Tipo**	**Requerido**	**Descripción**
ano	PK	Integer	PK	Llave primaria
mes	PK	Integer	PK	Llave primaria
idmercado	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_mercados.
cod_provincia	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_provincia.
cod_producto	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_clasificador
precio		Decimal	NOT NULL	Precio del producto

Tabla 2.6. Descripción de t_fusion_prod_merc_prov.

2.2.1.2 Descripción de procedimientos almacenados del Módulo de Datos IPC.

A continuación se hará referencia a la descripción de los principales procedimientos almacenados para el Módulo de Datos IPC, haciendo una descripción primeramente del objetivo de dicho procedimiento, además de representar los parámetros de entrada con el tipo de datos correspondiente (en algún caso específico) y la obtención de los resultados esperados.

Procedimiento	**GetProductos**	
Descripción	Devolver una lista de los productos según el código del producto y el nombre, que se encuentran involucrados en los mercados por provincias con sus productos.	
Parámetros	**Tipo**	**Descripción**
cod_provincia	int	El código de provincia.
idmercado	int	El identificador del tipo de mercado.
Resultado	record	Muestra la lista de los productos involucrados en una provincia con un mercado determinado. Record formado por los siguientes datos: o Código del producto. o Nombre.

Tabla 2.7 Descripción de GetProductos

Procedimiento	**GetALLClasif**	
Descripción	Devolver todos los elementos implicados en el clasificador.	
	Tipo	**Descripción**
Resultado	record	Muestra todos los elementos del clasificador. Record formado por los siguientes datos: o Código del producto. o Nombre. o Precios Mínimo (pmin) o Precio Máximos (pmax) o Precios Mínimo en Divisa(dpmin) o Precio Máximos en Divisa(dpmax) o Unidad de medida.

Tabla 2.8. Descripción de GetAllClasif.

Procedimiento	**DeleteAllClasif**	
Descripción	Elimina todos los elementos del clasificador.	
	Tipo	**Descripción**
Resultado	void	Muestra vacío el clasificador.

Tabla 2.9. Descripción de DeleteAllClasfi.

Procedimiento	**GetIDProvincia**	
Descripción	Devolver en una lista todas las provincias, según su código.	
Resultado	record	Muestra todas las provincias por su código. Record formado por los siguientes datos: o Código del producto.

Tabla 2.10. Descripción de GetIDProvincia.

Procedimiento	**GetProdMerc**	
Descripción	Devolver una lista de los productos según el código del producto y el precio, que se encuentran involucrados en una fusión.	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusión.
mes	int	Mes de la fusión.
idmercado	int	El identificador del tipo de mercado.
cod_provincia	int	El código de provincia.
Resultado	record	Muestra la lista de los productos involucrados en una fusión. Record formado por los siguientes datos: o Código del producto. o Precio.

Tabla 2.11. Descripción de GetProdMerc.

Procedimiento	**GetIDMercado**	
Descripción	Devolver una lista de los diferentes tipos de mercados presentes en una provincia.	
Parámetros	**Tipo**	**Descripción**
cod_provincia	int	El código de provincia.
Resultado	record	Muestra la lista de los tipos de mercados involucrados en una provincia determinada. Record formado por los siguientes datos: o Idmercado.

Tabla 2.12. Descripción de GetIDMercado.

2.2.2 Módulo de Seguridad del Sistema.

En este módulo se almacenan toda la información que los usuarios que interactúan con el sistema, los cuales contienen un rol, posibilitando acceder a ciertas funcionalidades en el sistema. Existen usuarios con diferentes privilegios según el rol que ocupan en el sistema.

De acuerdo a lo planteado anteriormente se represento la estructura lógica de la seguridad en la base de datos, ver figura 2.3:

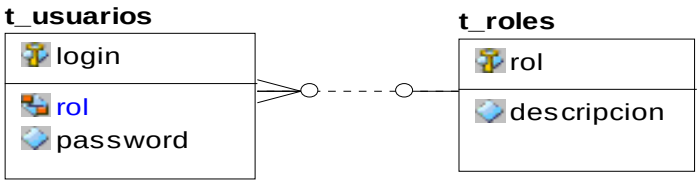


Fig.2.3. Modelo Lógico de Datos para Módulo de Seguridad del Sistema.

Interrelaciones:

Para las interrelaciones del modelo presentado anteriormente se describen los siguientes aspectos:

- Cada usuario del sistema contiene un rol en específico.
- Un rol lo poseen varios usuarios.

El diagrama del modelo de datos para el módulo de seguridad se corresponde con la representación física en la base de datos, como se muestra en la figura 2.4:

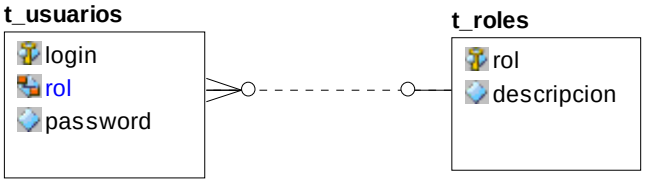


Fig.2.4. Modelo Físico de Datos para Módulo de Seguridad del Sistema.

2.2.2.1 Descripción de las tablas generadas:

A continuación se hará alusión a la descripción de las tablas de la base de datos representadas para el Módulo de Seguridad del Sistema, haciendo referencia primeramente al objetivo de dicha tabla, además de representar los atributos llaves o claves y no llave con el tipo de dato correspondiente y la declaración de dicho atributo que sea llave primaria o llave primaria compuesta o un valor no nulo.

.

Nombre: t_usuarios				
Descripción: Almacena los datos de los usuarios que interactúan en el sistema				
Atributo	**Llave**	**Tipo**	**Requerido**	**Descripción**
login	PK	Varchar[20]	PK	Usuario privilegiado o usuario administrado para acceder a la información del sistema.
password		Varchar[40]	NOT NULL	Contraseña del usuario para acceder al sistema.
rol	PFK	Integer	PFK	Llave primaria y foránea heredada de la tabla t_usuarios.

Tabla 2.13. Descripción de t_usuarios.

Nombre: t_roles				
Descripción: Almacena los datos de los usuarios que interactúan en el sistema				
Atributo	**Llave**	**Tipo**	**Requerido**	**Descripción**
rol	PK	Integer	PK	Identificador de los tipos de roles del sistema.
descripción		Varchar[20]	NOT NULL	Roles del sistema.

Tabla 2.14. Descripción de t_roles.

2.2.2.2 Descripción de los procedimientos almacenados para el Módulo de Seguridad del Sistema.

A continuación se hará referencia a la descripción de los principales procedimientos almacenados para el Módulo de Seguridad del Sistema, haciendo una descripción primeramente del objetivo de dicho procedimiento, además de representar los parámetros de entrada con el tipo de datos correspondiente (en algún caso específico) y la obtención de los resultados esperados.

Procedimiento	**InsertarUsuario**	
Descripción	Inserta un usuario en la base de datos.	
Parámetros	**Tipo**	**Descripción**
login	string	El login del usuario.
password	string	La contraseña del usuario.
rol	int	El identificador del rol del usuario.
Resultado	void	Se inserta un usuario.

Tabla 2.15. Descripción de InsertarUsuario.

Procedimiento	**EliminarUsuario**	
Descripción	Elimina un usuario de la base de datos.	
Parámetros	**Tipo**	**Descripción**
Resultado	void	Se elimina un usuario.

Tabla 2.16. Descripción de EliminarUsuario.

Procedimiento	**ActualizarUsuario**	
Descripción	Actualizar un usuario con los valores de los parámetros de entrada.	
Parámetros	**Tipo**	**Descripción**
login	string	El login del usuario.
password	string	La contraseña del usuario.
rol	int	El identificador del rol del usuario.
Resultado	void	Se actualiza un usuario.

Tabla 2.17. Descripción de ActualizarUsuario.

Procedimiento	**GetRolUser**	
Descripción	Inserta un usuario en la base de datos.	
Parámetros	**Tipo**	**Descripción**
login	string	El login del usuario.
Resultado	Set of record	Muestra el rol que ocupa dicho usuario en la base de datos.

Tabla 2.18. Descripción de GetRolUser.

Procedimiento	**GetALLUser**	
Descripción	Devolver todos los usuarios de la base de datos.	
Resultado	record	Muestra todos los usuarios Record formado por los siguientes datos: o Login o Rol

Tabla 2.19. Descripción de GetAllUser.

2.3 Descripción de la arquitectura Microsoft SQL Server

La arquitectura de Microsoft SQL Server es de 3 niveles lo cual está en correspondencia con la propuesta por el grupo ANSI/SPARC. Esta arquitectura se corresponde suficientemente bien con un gran número de sistemas, aunque no se puede asegurar que cualquier SGBD se corresponda exactamente con ella, ver figura 2.5.

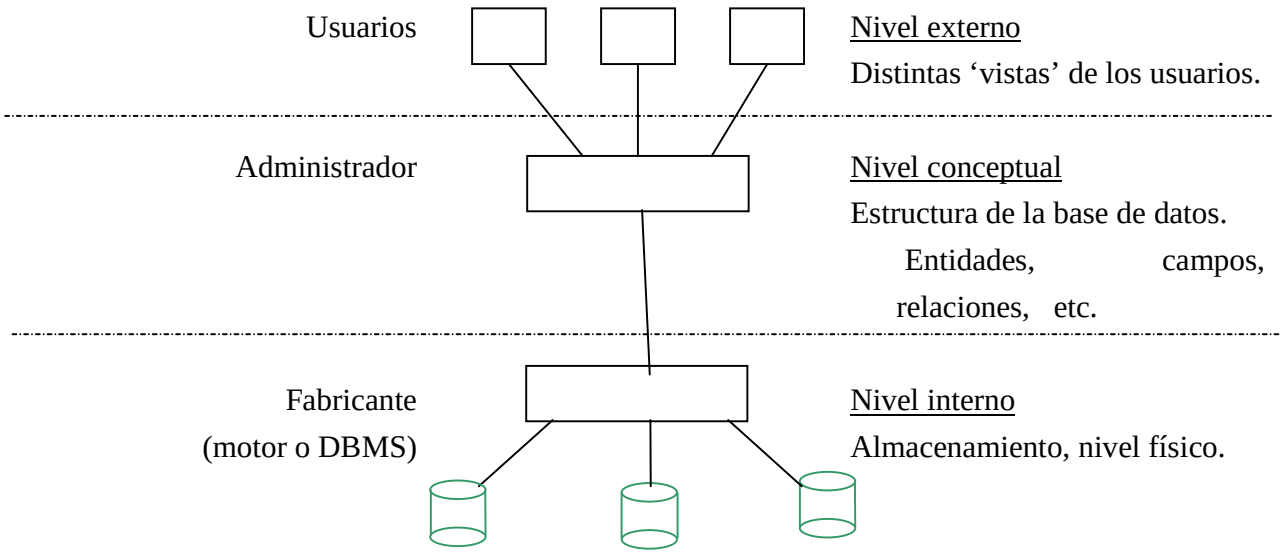


Fig.2.5. Arquitectura del Sistema Gestor de Bases de Datos Microsoft SQL Server 2000.

- **Nivel interno o físico.** Se refiere al almacenamiento físico en el se describe cómo se almacenan realmente los datos en memorias secundarias, en qué archivos, su nombre y dirección. También estarán los registros, longitud, campos, índices y las rutas de acceso a esos archivos, de forma más específica:
  - Describe estructura física de la base de datos
  - Emplea modelo físico
  - Describe detalles de almacenamiento y caminos de acceso

- **Nivel Conceptual.** En el se describen cuáles son los datos reales almacenados en la BD y que relaciones existen entre ellas. Este nivel lo definen los administradores de la BD que son los que deciden que información se guarda en la BD. Este nivel corresponde a la estructura organizacional de los datos obtenida al reunir los requerimientos de todos los usuarios, sin preocuparse de su organización física ni de las vías de acceso.
  - Describe estructura de toda la BD para la comunidad de usuarios
  - Oculta los detalles de estructuras de almacenamiento
  - Describe entidades, tipos, vínculos, operaciones, restricciones

Podría contener de forma específica:

- Entidades del mundo real (clientes, artículos, pedidos, ...)
- Atributos de las entidades (nombre_cliente, NIF, ...)
- Asociaciones entre entidades (compra de artículos)
- Restricciones de integridad (son las normas que deben cumplir los datos).

- **Nivel externo o vistas.** Es el nivel más cercano al usuario y representa la percepción individual de cada usuario. Si los niveles interno y conceptual describen toda la BD, este nivel describe únicamente la parte de datos para un usuario o grupo de usuarios. Habrá usuarios que podrán acceder a más de un esquema externo y uno de éstos puede ser compartido por varios usuarios, se protege así el acceso a los datos por parte de personas no autorizadas. A la hora de construir un esquema externo:
  - Se pueden omitir una o más entidades del sistema.
  - Se pueden omitir uno o más atributos de una entidad.
  - Se pueden omitir una o más relaciones entre los datos.
  - Se pueden cambiar el orden de los atributos.

Esto se puede ver de forma más explicita:

- Describe una parte de la BD que interesa a un grupo de usuarios determinado[19]

[19] Obra Citada [1]

La arquitectura de 3 niveles permite obtener independencia física y lógica de los datos:

La arquitectura de tres niveles es útil para explicar el concepto de independencia de datos que podemos definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Se pueden definir dos tipos de independencia de datos:

- Independencia Lógica
  - Modificación de esquema conceptual no requiere alterar esquemas externos ni programas de aplicación, coexiste en ser la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla. Si, por ejemplo, se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberán verse afectados.

- Independencia Física
  - Modificación de esquema interno no requiere modificar esquema conceptual (ni externos), se entiende como la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos). Por ejemplo, puede ser necesario reorganizar ciertos ficheros físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización de datos. Dado que la independencia física se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de conseguir que la independencia lógica. [20]

Para plasmar los tres niveles en el enfoque o modelo de datos seleccionado, es necesaria una aplicación que actúe de interfaz entre el usuario, los modelos y el sistema físico.
Esta es la función que desempeñan los SGBD, ya reseñados, y que pueden definirse como un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos

[20] ANDRÉS, M. M. M. Arquitectura de los sistemas de bases de datos, [11 de abril de 2007]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/node33.html

a los datos y actuando de interfaz entre los datos físicos y el usuario.

Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad.
El SGBD incorpora como herramienta fundamental dos lenguajes, para la definición y la manipulación de los datos.

El lenguaje de definición de datos (DDL, Data Definition Language) provee de los medios necesarios para definir los datos con precisión, especificando las distintas estructuras.
Acorde con el modelo de arquitectura de tres niveles, habrá un lenguaje de definición de la estructura lógica global, otro para la definición de la estructura interna, y un tercero para la definición de las estructuras externas.

El lenguaje de manipulación de datos (DML, Data Manipulation/ Management Language), que es el encargado de facilitar a los usuarios el acceso y manipulación de los datos. Pueden diferenciarse en procedimentales (aquellos que requieren qué datos se necesitan y cómo obtenerlos) y no procedimentales (que datos se necesitan, sin especificar como obtenerlos), y se encargan de la recuperación de los datos almacenados, de la inserción y supresión de datos en la base de datos, y de la modificación de los existentes. [21]

(Ver Anexo1 Arquitectura de los Sistemas Gestores de Bases de Datos).

[21] TRAMULLAS, J. Los sistemas de gestión de bases de datos., [12 de abril de 2007]. Disponible en: http://tramullas.com/documatica/2-4.html.

2.4 Descripción de la arquitectura y fundamentación.

Siguiendo la filosofía del modelo actual de desarrollo del software, para el desarrollo de este sistema, de una forma estructurada, se tomó como base la arquitectura de 3 capas:

- La capa de presentación o interfaz de usuario.

  En este caso, está formada por los formularios y los controles que se encuentran en los formularios. Capa con la que interactúa el usuario (Windows Forms para desarrollar interfaces de usuario, usando como plataforma Microsoft Visual C#).

- La capa de negocio

Esta capa está formada por las entidades empresariales, que representan objetos que van a ser manejados o consumidos por toda la aplicación. En este caso, están representados por las clases y los DataTables tipados que se crean.

- La capa de acceso a datos.

  Contiene clases que interactúan con la base de datos, estas clases altamente especializadas se encuentran en la arquitectura ADO.NET y permiten, utilizando los procedimientos almacenados generados, realizar todas las operaciones con la base de datos de forma transparente para la capa de negocio (ADO.NET para conectar las aplicaciones a bases de datos). [22]

(Ver Anexo2. Arquitectura de 3 Capas).

En éste epígrafe se presentará una visión general de la arquitectura de acceso a datos disponible a través de ADO .NET, ver figura 2.6.

ADO .NET, una evolución de la arquitectura de acceso a datos proveída por el modelo de programación *Microsoft ActiveX Data Objects* (ADO), provee a los programadores de .NET un acceso a las fuentes de datos relacionales, XML y aplicaciones de datos.

ADO .NET ayuda a una variedad de necesidades de desarrollo, incluyendo clientes de bases de datos y objetos del negocio intermedios usados por aplicaciones, herramientas, lenguajes y navegadores de Internet.

[22] FANJUL, A. Arquitectura de 3 Capas, [17 de abril de 2007]. Disponible en: http://www.generatorfd.com/Arquitectura.aspx

Es importante señalar que ADO .NET está diseñado para fundamentarse en la fuerza del modelo de programación de ADO, mientras que provee una evolución en la tecnología de acceso a datos para responder a las necesidades cambiantes del desarrollador manteniendo un control más fino sobre los componentes, recursos y comportamiento de la aplicación cuando accede y trabaja con datos.

Arquitectura de ADO.NET

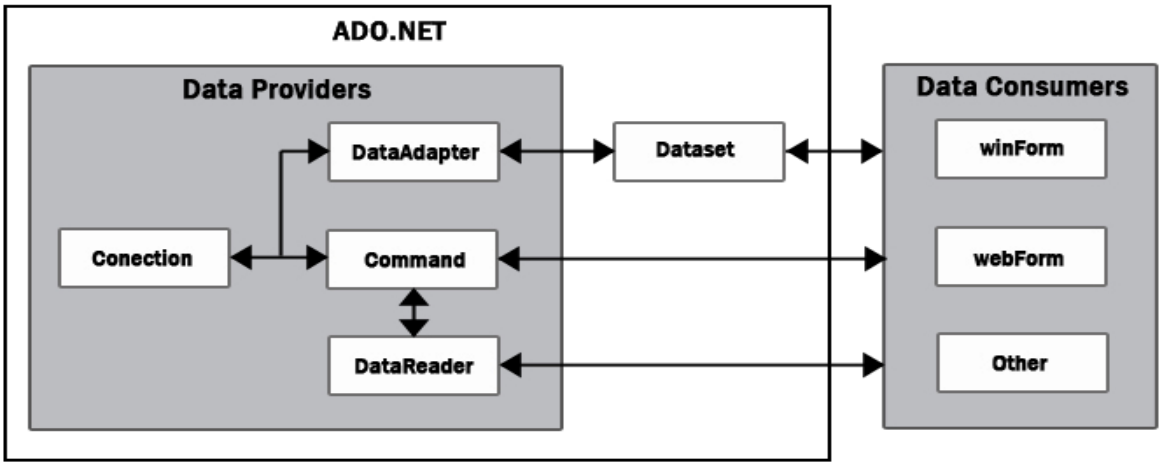


Fig.2.6. Arquitectura de ADO.

En el diseño de ADO .NET, se tuvieron en consideración tareas a las cuales se enfrenta un desarrollador comúnmente cuando accede o trabaja con datos. Al contrario de usar un objeto simple para llevar a cabo un determinado número de tareas, ADO .NET agencia funcionalidades específicas en objetos explícitos que están optimizados para permitir al desarrollador llevar a cabo de cada una de las tareas.
La funcionalidad que un *Recorset* de ADO provee ha sido dividida en los siguientes objetos explícitos en ADO .NET:

- el ***DataReader***, que provee un acceso rápido, hacia delante (***forward-only***) y de sólo lectura en los resultados de una consulta;
- el ***DataSet***, que provee una representación relacional de los datos en memoria;
- el ***DataAdapter***, que provee un puente entre el ***DataSet*** y la fuente de datos.
- El objeto ***Command*** de ADO .NET incluye además funcionalidad explícita tales como el método

ExecuteNonQuery para comandos que no retornan filas, y el método ***ExecuteScalar*** para consultas que devuelven un único valor en cambio de un conjunto de filas.

Para una mejor comprensión de cómo el diseño de ADO .NET está constituido por objetos optimizados para llevar a cabo un comportamiento explícito, considera la siguiente tarea común cuando se trabaja con datos:

- **Recuperando y actualizando datos desde la fuente de datos**:

El ***DataAdapter*** de ADO .NET provee un puente entre el ***DataSet*** y la fuente de datos. El ***DataAdapter*** se usa para llenar un ***DataSet*** con resultados obtenidos desde la base de datos, y para leer cambios en el ***DataSet*** y actualizarlos a la base de datos.

Usando un objeto separado, el ***DataAdapter***, para comunicarse con la base de datos permite completamente al ***DataSet*** mantenerse genérico con respecto a los datos que contiene, y permite que tengas un mayor control sobre cuándo y cómo se ejecutan los comandos y se envían los cambios a la base de datos. ADO desempeña mucho de este comportamiento implícitamente, sin embargo, el diseño explícito de ADO .NET permite una mejor interacción con la fuente de datos para un mejor rendimiento y escalabilidad. [23]

[23] CEBALLOS, G. M. ADO .NET para Programadores de ADO, 2007. [18 de abril de 2007]. Disponible en: http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_2289.asp

2.5 Modelo de Implementación de la Capa de Acceso a Datos

El modelo de implementación de la capa de acceso a datos describe cómo se implementan los elementos del modelo de diseño. Al analizar la arquitectura de la CAD se puede observar una vista general y detallada de cada uno de los paquetes, en aras de lograr una mayor claridad y comprensión del modelo.
La arquitectura de la capa de acceso a datos está formada por 4 paquetes, ver la figura 2.7:

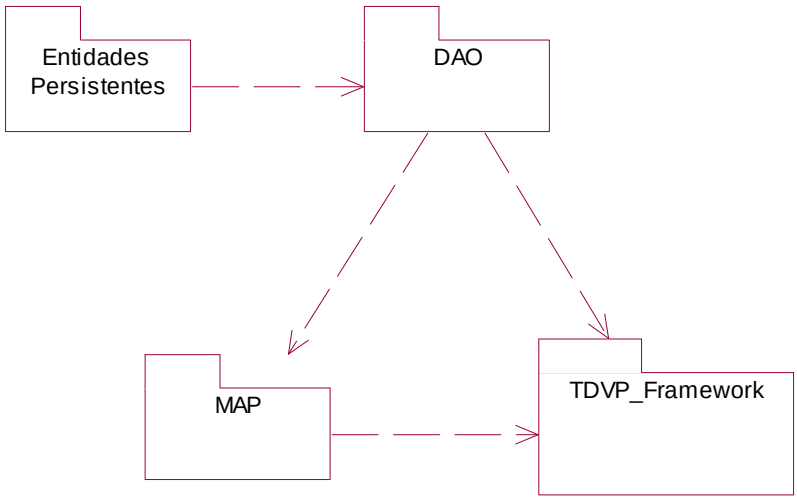


Fig. 2.7. Arquitectura de la Capa de Acceso a Datos.

El paquete Entidades Persistentes representa las clases persistentes que constituyen las entidades del negocio con los atributos que se guardan en la base de datos; a continuación está el paquete DAO, el cual permite coger las entidades persistentes del negocio y guardarlas en la base de datos, así como recuperarlas de la misma, además de proveer algunas funciones específicas para obtener la información necesaria de la base de datos.

El paquete MAP contiene las clases generadas del mapeo de la base de datos con la herramienta TierDeveloper, por cada tabla se genera una clase entidad que representa una tupla de la tabla, una clase Factory que tiene todos los métodos para las operaciones con las tablas, una clase Collection que contiene una lista de tuplas donde se hace referencia al paquete TDVP_Framework.

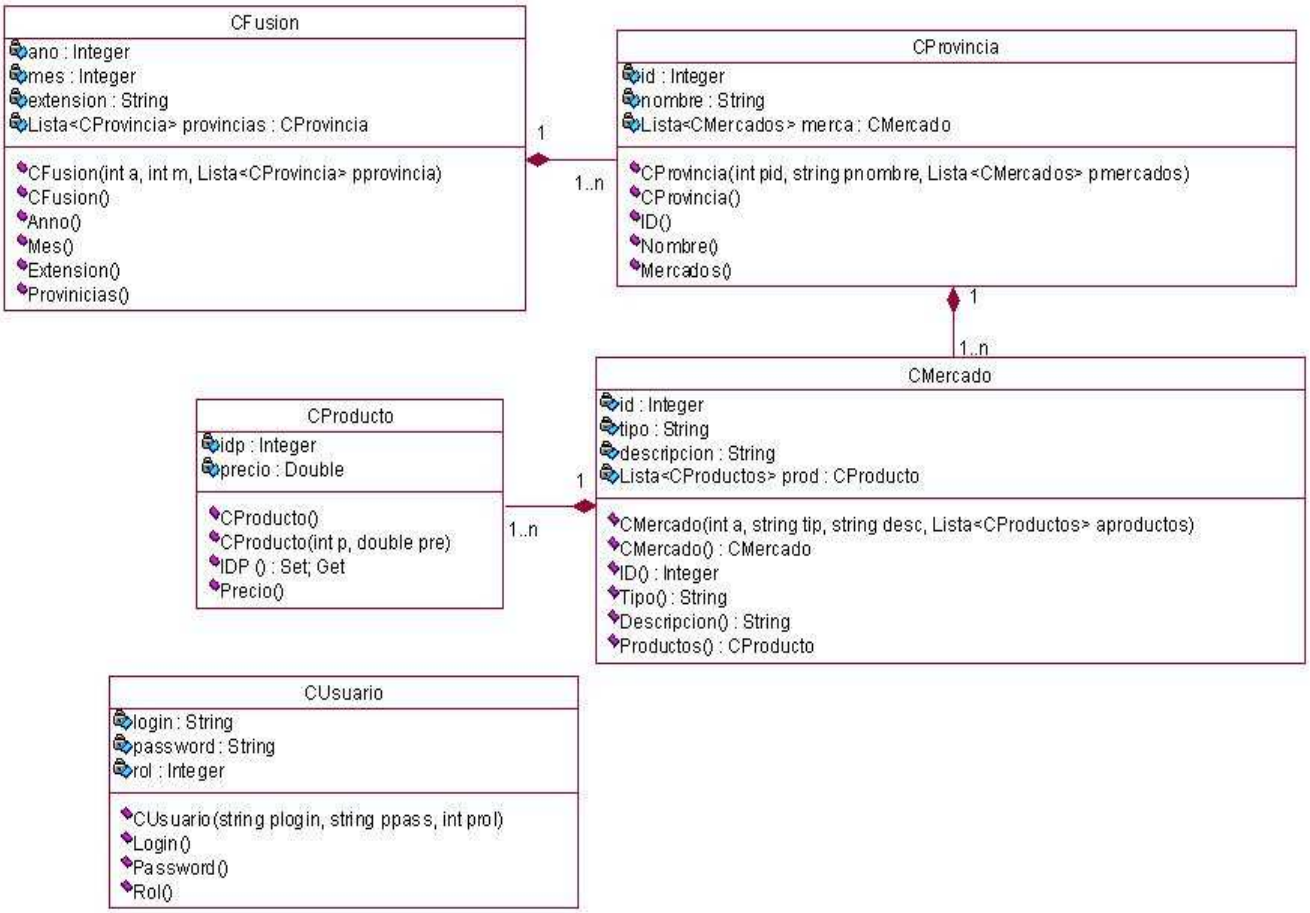
TierDeveloper introduce la biblioteca “TDevFramework” como parte del código de los componentes generado y el propósito básico de esta biblioteca es manejar el código comúnmente reusable a través de un solo punto de referencia. Así se elimina cualquier duplicación del código por los componentes y se proporcionan clases básicas, que son heredadas por las clases de implementación de los componentes. Todas las clases básicas son parte de la dll de TDevFramework, que debe desplegarse junto con los componentes de la aplicación generados en la plataforma específica designada. [24]

(Ver Anexo 3. Diagrama de Clases TDevFramework)

[24] TierDeveloper. Programmer’s Guide for .Net. . 2006. [25 de abril de 2007]. Disponible en: http://www.alachisoft.com/tdev/

2.5.1 Clases Persistentes

El diagrama de clases persistentes es usado para representar las clases que maneja la aplicación y contienen los datos que se recuperan y guardan de la BD, como se observa en la figura 2.8.



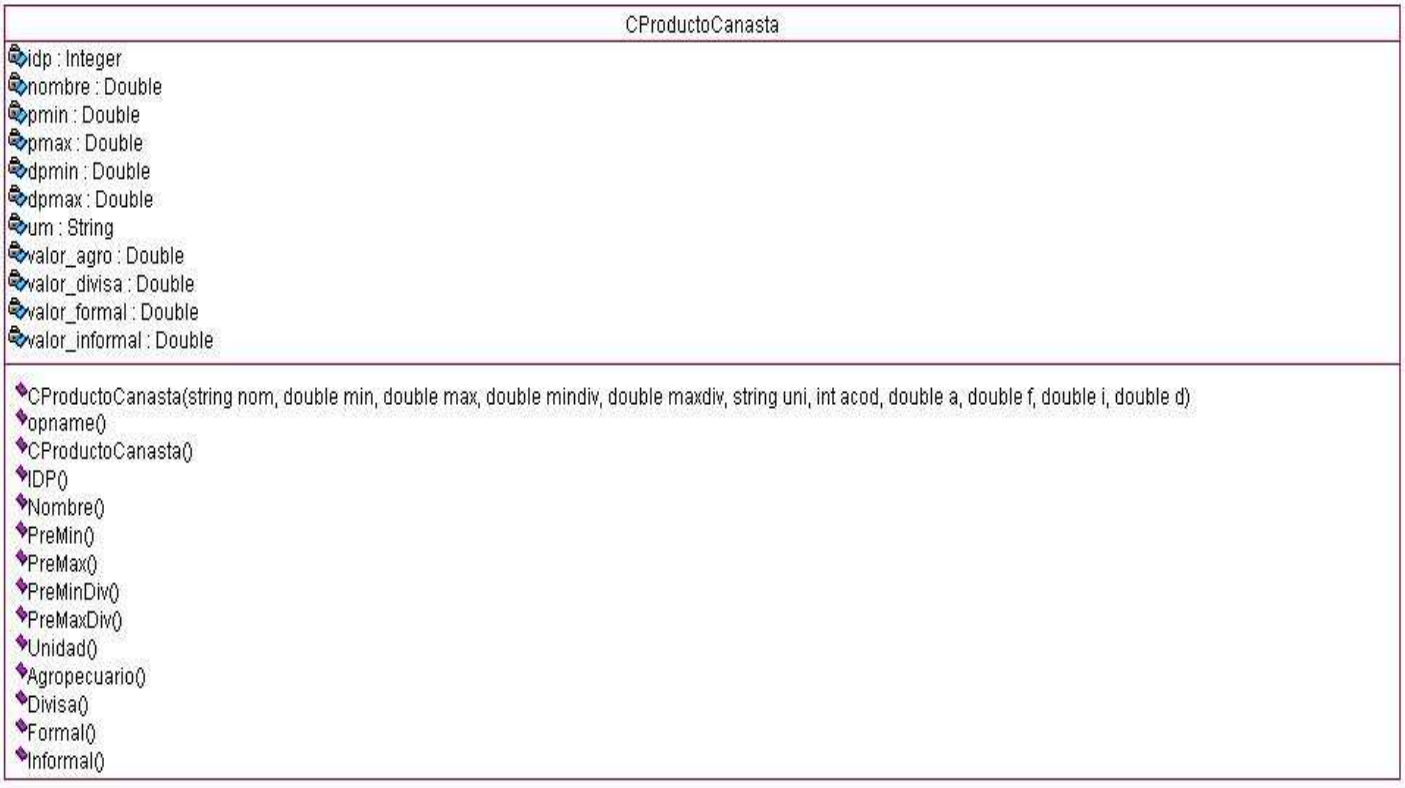


Fig.2.8. Diagrama de clases persistentes.

2.5.1.1 Descripción de las clases persistentes.

A continuación se representará la descripción de las clases persistentes, haciendo referencia para cada clase los atributos privados con el tipo de dato correspondiente, y sus respectivos métodos.

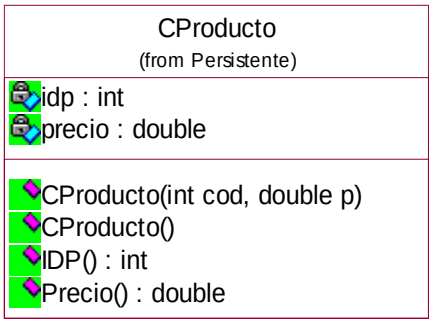


Fig. 2.9. Entidad CProducto.

Nombre: CProducto	
Tipo de clase (Entidad**)**	
Atributo	**Tipo**
idp precio	int double
Para cada responsabilidad:	
Nombre:	**Descripción:**
CProducto()	Constructor vacío, para la creación de objetos del tipo CProducto
CProducto(int p, double pre)	Constructor por parámetros, para la creación de objetos con parámetros del tipo CProducto
IDP	Propiedades para la asignación y obtención del atributo idp.
Precio	Propiedades para la asignación y obtención del atributo precio.

Tabla 2.20. Descripción de la clase persistente CProducto.

Fig.2.10. Entidad CMercado.

Nombre: CMercado	
Tipo de clase (Entidad**)**	
Atributo	**Tipo**
id	string
tipo	string
descripción	string
prod	List<CProducto>
Para cada responsabilidad:	
Nombre:	**Descripción:**
CMercado()	Constructor vacío, para la creación de objetos del tipo CMercado
CMercado(string a, string tip, string descripcion, List<CProducto> aproductos)	Constructor por parámetros, para la creación de objetos con parámetros del tipo CMercado
ID	Propiedades para la asignación y obtención del atributo id.
Tipo	Propiedades para la asignación y obtención del atributo tipo.
Descripción	Propiedades para la asignación y obtención del atributo descripción.

List<CProducto> Productos()	Operación para obtener la lista de productos de un mercado determinado.

Tabla 2.21. Descripción de la clase persistente CMercado.

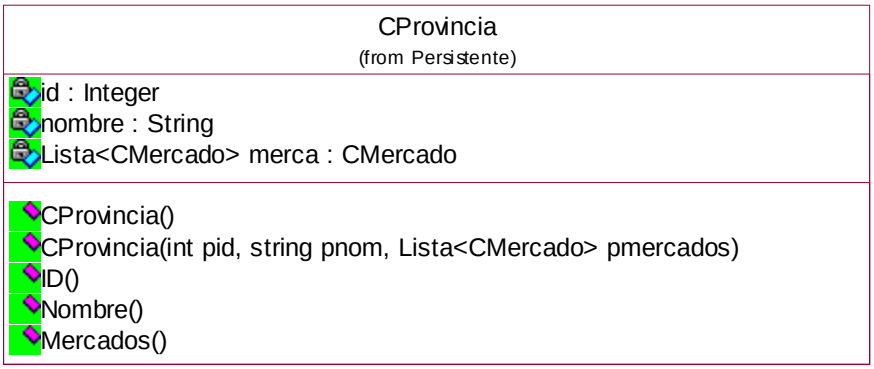


Fig. 2.11. Entidad CProvincia.

Nombre: CProvincia	
Tipo de clase (Entidad**)**	
Atributo	**Tipo**
id nombre merca	string string List<CMercado>
Para cada responsabilidad:	
Nombre:	**Descripción:**
CProvincia()	Constructor vacío, para la creación de objetos del tipo CProvincia.
CProvincia(string pid, string pnombre, List<CMercado> pmercados)	Constructor por parámetros, para la creación de objetos con parámetros del tipo CProvincia.
ID	Propiedades para la asignación y obtención del atributo id.

Nombre	Propiedades para la asignación y obtención del atributo nombre.
List<CMercado> Mercados()	Operación para obtener la lista de mercados de una provincia determinada.

Tabla 2.22. Descripción de la clase persistente CProvincia.

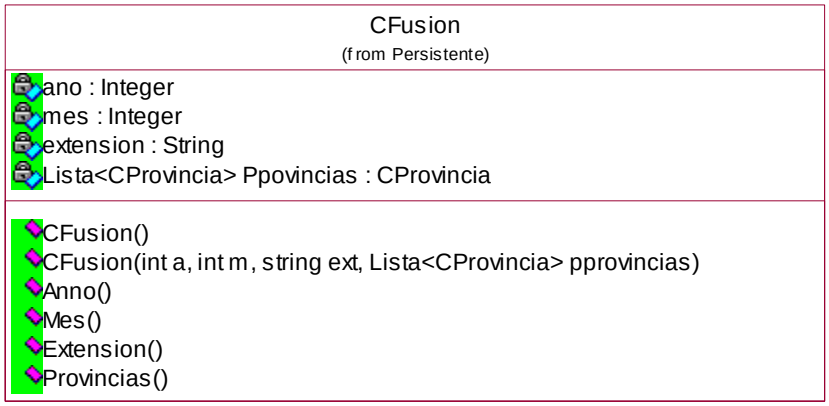


Fig.2.12. Entidad CFusion.

Nombre: CFusion	
Tipo de clase (Entidad**)**	
Atributo	**Tipo**
año	int
mes	string
ext	string
provincias	List<CProvincia>
Para cada responsabilidad:	
Nombre:	**Descripción:**
CFusion()	Constructor vacío, para la creación de objetos del tipo CFusion.
CFusion(int a, string m, List<CProvincia> pprovincias)	Constructor por parámetros, para la creación de objetos con parámetros del tipo CFusion.

Anno	Propiedades para la asignación y obtención del atributo año.
Mes	Propiedades para la asignación y obtención del atributo mes.
Extension	Propiedades para la asignación y obtención del atributo ext.
List<CProvincia> Provincias()	Operación para obtener la lista de provincias de una fusión determinada.

Tabla 2.23. Descripción de la clase persistente CFusion.

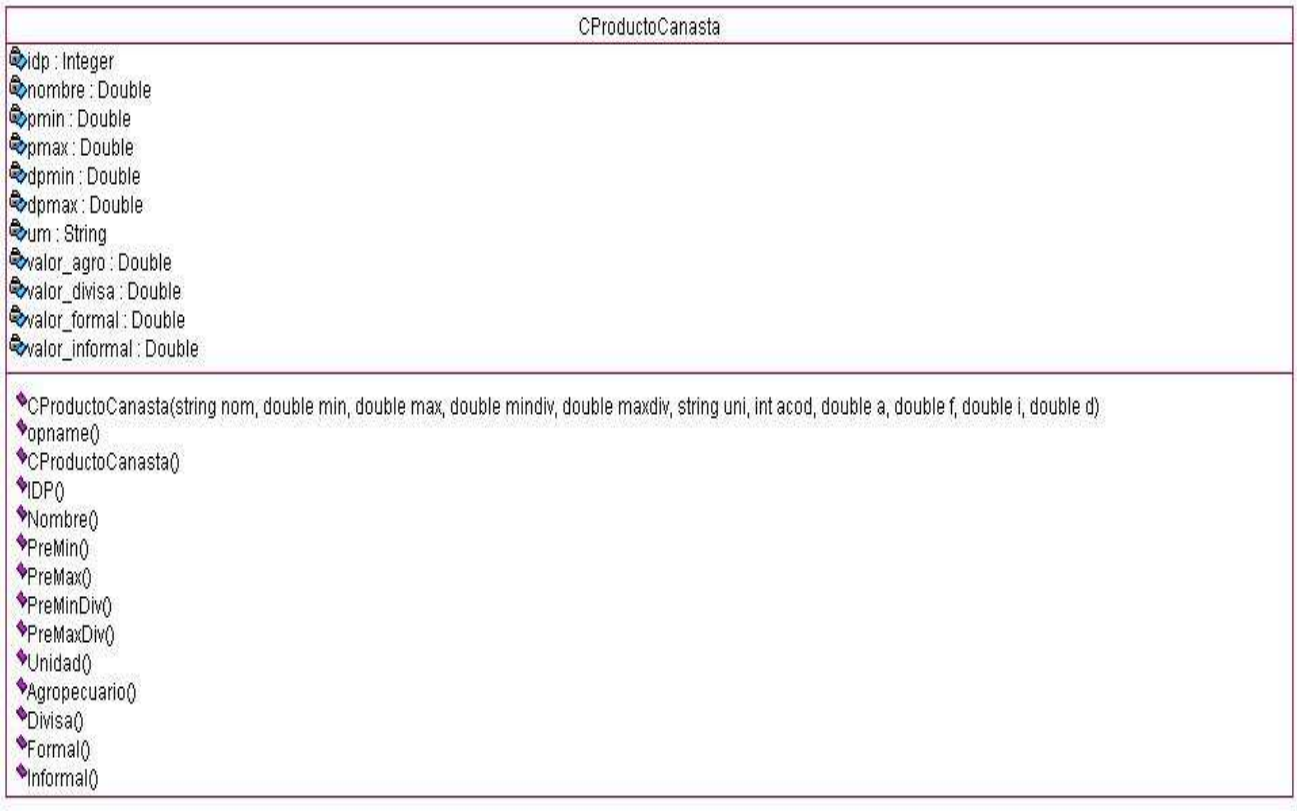


Fig.2.13. Entidad CProductoCanasta.

Nombre: CProductoCanasta	
Tipo de clase (Entidad**)**	
Atributo	**Tipo**
idp	int
nombre	string
preciomin	double
preciomax	double
preciomindiv	double
preciomaxdiv	double
unidad	string
agropecuario	double
formal	double
informal	double
divisa	double
Para cada responsabilidad:	
Nombre:	**Descripción:**
CProductoCanasta()	Constructor vacío, para la creación de objetos del tipo CProductoCanasta.
CProductoCanasta(int cod, string nom, double a, double f, f double i, double d, string uni, int cod, double a, double f, double i, double d)	Constructor por parámetros, para la creación de objetos con parámetros del tipo CProductoCanasta.
IDP	Propiedades para la asignación y obtención del atributo idp.
Nombre	Propiedades para la asignación y obtención del atributo nombre.
PreMin	Propiedades para la asignación y obtención del atributo preciomin.
PreMax	Propiedades para la asignación y obtención del atributo preciomax.
PreMinDiv	Propiedades para la asignación y obtención del atributo preciomindiv.

PreMaxDiv	Propiedades para la asignación y obtención del atributo preciomaxdiv.
Unidad	Propiedades para la asignación y obtención del atributo unidad.
Agropecuario	Propiedades para la asignación y obtención del atributo agropecuario.
Formal	Propiedades para la asignación y obtención del atributo formal.
Informal	Propiedades para la asignación y obtención del atributo informal.
Divisa	Propiedades para la asignación y obtención del atributo divisa.

Tabla 2.24. Descripción de la clase persistente CProductoCanasta.

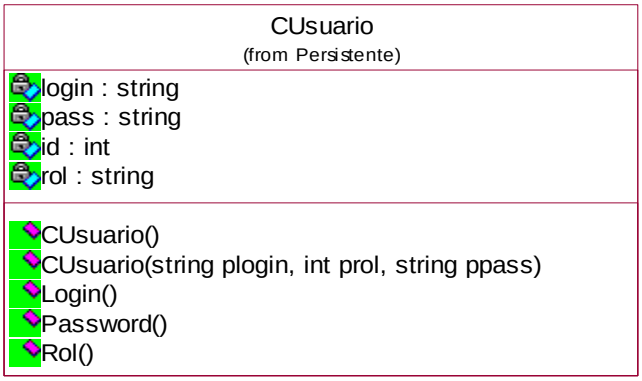


Fig.2.14. Entidad CUsuario.

Nombre: CUsuario	
Tipo de clase (Entidad**)**	
Atributo	**Tipo**
login rol password	string int string
Para cada responsabilidad:	
Nombre:	**Descripción:**
CUsuario (string plogin, int prol, string ppass)	Constructor por parámetros, para la creación de objetos con parámetros del tipo CUsuario.
Login	Propiedades para la asignación y obtención del atributo login.
Rol	Propiedades para la asignación y obtención del atributo rol.
Password	Propiedades para la asignación y obtención del atributo password.

Tabla 2.25. Descripción de la clase persistente CUsuario.

2.5.2 Clases para el acceso a la Base de Datos.

El diagrama de clases DAO es usado para modelar la estructura lógica de la capa de acceso da datos.

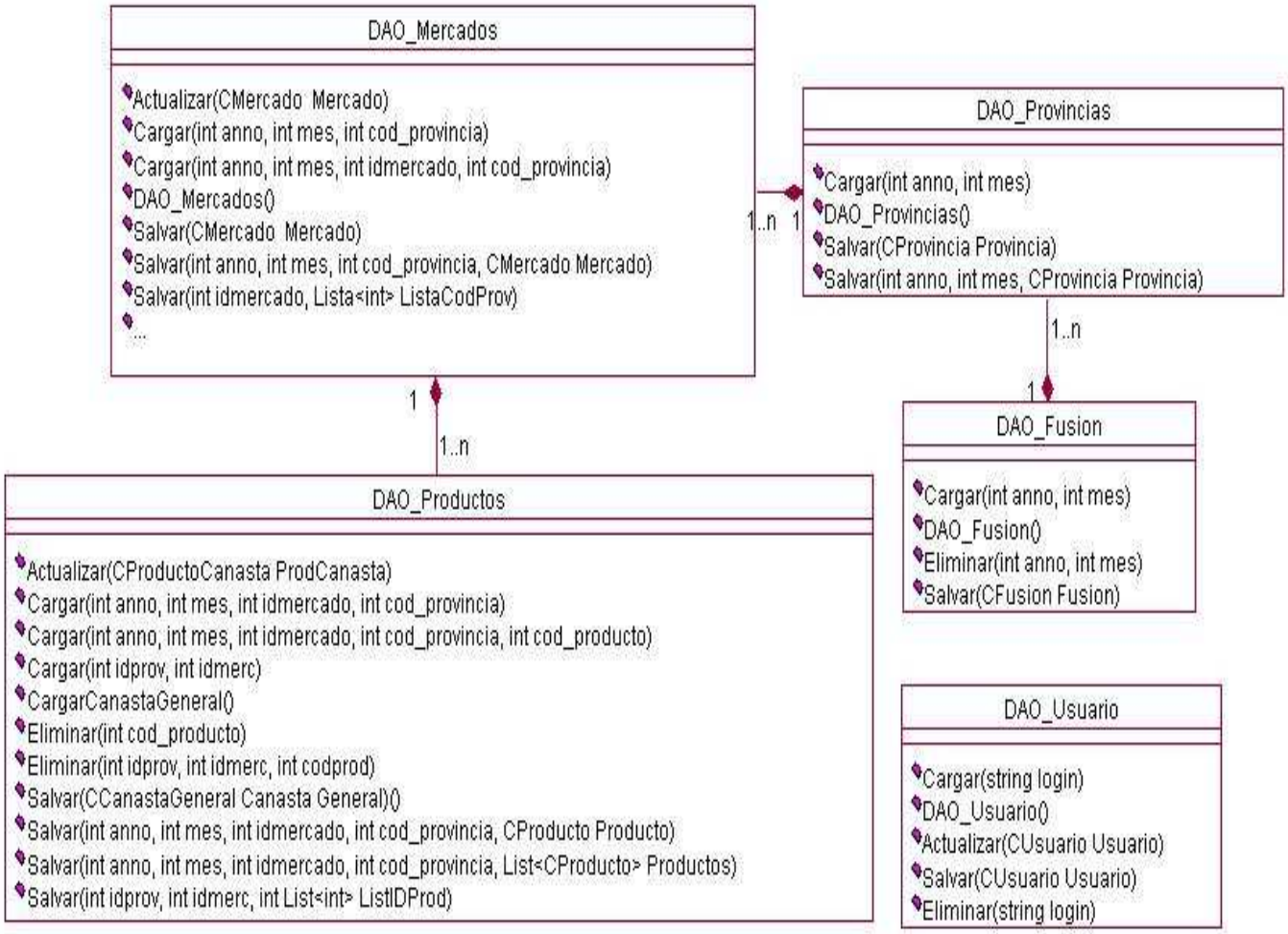


Fig.2.15. Diagramas de clases DAO para la implementación de CAD.

2.5.2.1 Descripción de las clases de acceso a datos.

A continuación se representará la descripción de las clases DAO, haciendo referencia a las clases con sus respectivos métodos, con una breve descripción de lo que realiza.

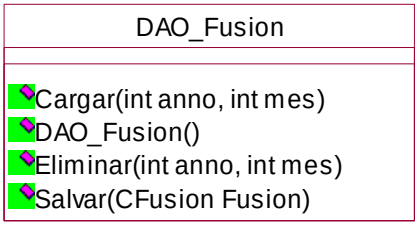


Fig.2.16. Clase DAO_Fusion.

Nombre	**DAO_Fusion**	
Descripción	Clase base para las demás clases que gestionan objetos y realiza la conexión a la base de datos.	
Atributos	**Tipo**	**Descripción**
miDAO_Provincias	DAO_Provincias	Objeto de tipo DAO_Provincias

Método	**DAO_Fusion()**	
Descripción	Constructor que crea el objeto miDAO_Provincias.	
Método	**Cargar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusión.
mes	int	Mes de la fusión.
Descripción	Devuelve un objeto CFusion de la B.Datos.	
Método	**Eliminar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusión.
mes	int	Mes de la fusión.
Descripción	Elimina un objeto CFusion en la B.Datos	

Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
pFusion	CFusion	Se crea un objeto de tipo CFusion.
Descripción	Devuelve un objeto CFusion de la B.Datos	

Tabla 2.26. Descripción de la clase DAO_Fusion.

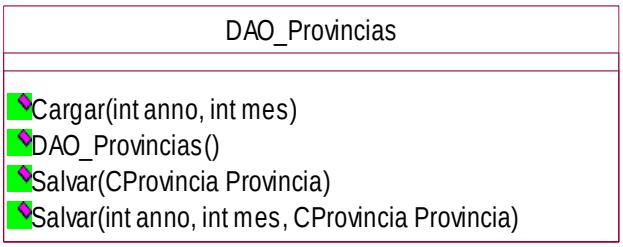


Fig.2.17. Clase DAO_Provincias.

Nombre	**DAO_Provincias**	
Descripción	Clase base donde se gestionan objetos y realiza la conexión a la base de datos.	
miDAO_Mercados	DAO_Mercados	Objeto de tipo DAO_Provincias
Método	**DAO_Provincias()**	
Descripción	Constructor que crea el objeto miDAO_Mercado	
Método	**Cargar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusión.
mes	int	Mes de la fusión.
Descripción	Devuelve la lista de objetos CProvincia de una fusion	
Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusión.
mes	int	Mes de la fusión.

Provincia	CProvincia	Se crea un objeto de tipo CProvincia.
Descripción	Guarda un objeto CProvincia de una fusion en la BD.	
Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
provincia	CProvincia	Se crea un objeto de tipo CProvincia.
Descripción	Guarda un objeto CProvincia en la BD.	

Tabla 2.27. Descripción de la clase DAO_Provincias.

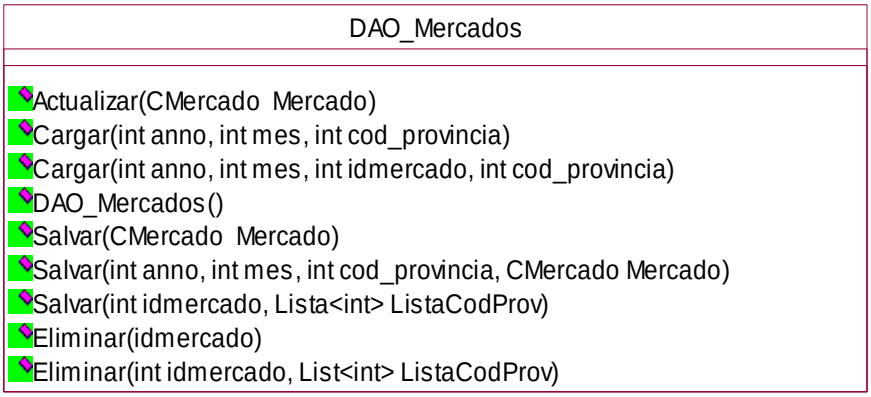


Fig.2.18. Clase DAO_Mercados.

Nombre	**DAO_Mercados**	
Descripción	Clase base donde se gestionan objetos y realiza la conexión a la base de datos.	
miDAO_Productos	DAO_Productos	Objeto de tipo DAO_Productos.
Método	**DAO_Mercados()**	
Descripción	Constructor que crea el objeto miDAO_Productos.	
Método	**Actualizar()**	

Parámetros	**Tipo**	**Descripción**
mercado	CMercado	Objeto de tipo CMercado.
Descripción	Actualiza un objeto CMercado en la BD.	
Método	**Cargar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusión.
mes	int	Mes de la fusión.
cod_provincia	int	El código de la provincia
Descripción	Devuelve una lista de objetos CMercados de una fusion.	
Método	**Cargar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusión.
mes	int	Mes de la fusión.
idmercado	int	El identificador del tipo de mercado.
cod_provincia	int	El código de la provincia.
Descripción	Devuelve un objeto CMercado de una fusion.	
Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
Mercado	CMercado	Se crea un objeto de tipo CMercado.
Descripción	Guarda un objeto CMercado en la B.Datos	
Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusion.
mes	int	Mes de la fusion.
cod_provincia	int	El código de la provincia
Mercado	CMercado	Se crea un objeto de tipo CMercado.
Descripción	Guarda un objeto CMercado de una fusion en la BD.	
Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
cod_mercado	int	El identificador del tipo de mercado.
ListaCodProv	List<int>	Una lista de provincias según su código.

Descripción	Guarda las provincias con un mercado determinado en la BD.	
Método	**Eliminar()**	
Parámetros	**Tipo**	**Descripción**
idmercado	int	El identificador del tipo de mercado.
Descripción	Elimina un objeto CMercado en la BD.	
Método	**Eliminar()**	
Parámetros	**Tipo**	**Descripción**
idmercado	int	El identificador del tipo de mercado.
ListaCodProv	List<int>	Una lista de provincias según su código.
Descripción	Elimina un mercado en las provincias especificadas en la BD.	

Tabla 2.28. Descripción de la clase DAO_Mercados.

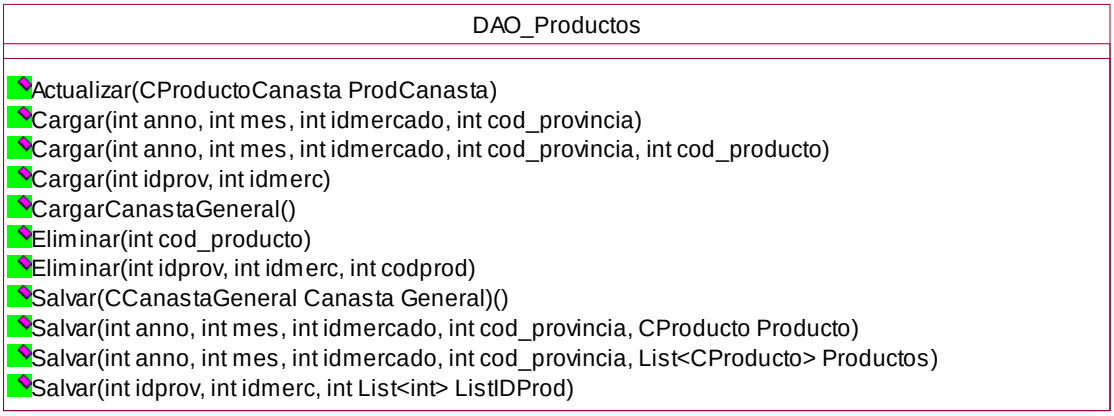


Fig.2.19. Clase DAO_Productos.

Nombre	**DAO_Productos**	
Descripción	Clase base donde se gestionan objetos y realiza la conexión a la base de datos.	
Método	**DAO_Productos()**	
Descripción	Constructor sin parámetros.	
Método	**Actualizar()**	
Parámetros	**Tipo**	**Descripción**
ProductoCanasta	CProductoCanasta	Se crea un objeto de tipo CProductoCanasta.
Descripción	Actualiza un objeto CProductoCanasta en la BD.	
Método	**Cargar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusion.
mes	int	Mes de la fusion.
idmercado	int	El identificador del tipo de mercado.
cod_provincia	int	El código de la provincia.
Descripción	Devuelve una lista de objetos CProductos de una fusion.	
Método	**Cargar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusion.
mes	int	Mes de la fusion.
idmercado	int	El identificador del tipo de mercado.
cod_provincia	int	El código de la provincia.
cod_producto	int	El código del producto.
Descripción	Devuelve un objeto CProducto de una fusion	
Método	**Cargar()**	
Parámetros	**Tipo**	**Descripción**
idprov	int	El código de la provincia.
idmerc	int	El código del producto.
Descripción	Devuelve todos los objetos CProductoCanasta de un mercado en una provincia.	
Método	**CargarCanastaGeneral()**	
Descripción	Devuelve el objeto CCanastaGeneral de la BD.	

Método	**Eliminar()**	
Parámetros	**Tipo**	**Descripción**
cod_producto	int	El código del producto.
Descripción	Elimina un producto de acuerdo a su código.	
Método	**Eliminar()**	
Parámetros	**Tipo**	**Descripción**
idprov	int	El código de la provincia.
idmerc	int	El identificador del tipo de mercado.
idprod	int	El código del producto.
Descripción	Elimina un producto del mercado de una provincia.	
Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
CanastaGeneral	CCanastaGeneral	Se creó un objeto de tipo CCanastaGeneral.
Descripción	Guarda un objeto CCanastaGeneral en la BD.	
Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusion.
mes	int	Mes de la fusion.
idmercado	int	El identificador del tipo de mercado.
cod_provincia	int	El código de la provincia.
Producto	CProducto	Se crea un objeto de tipo CProducto.
Descripción	Guarda un objeto CProducto de una fusion en la BD.	
Método	**Salvar()**	
Parámetros	**Tipo**	**Descripción**
ano	int	Año de la fusion.
mes	int	Mes de la fusion.
idmercado	int	El identificador del tipo de mercado.
cod_provincia	int	El código de la provincia.
Productos	List<CProducto>	Se crea una lista de productos.
Descripción	Guarda una lista de objetos CProductos de una fusion en la BD.	
Método	**Salvar()**	

Parámetros	Tipo	Descripción
idprov	int	El código de la provincia.
idmerc	int	El identificador del tipo de mercado.
ListaIdProd	List<int>	Una lista de productos según su código.
Descripción	Guarda los productos existentes en un mercado de una provincia en la BD.	

Tabla 2.29. Descripción de la clase DAO_Productos.

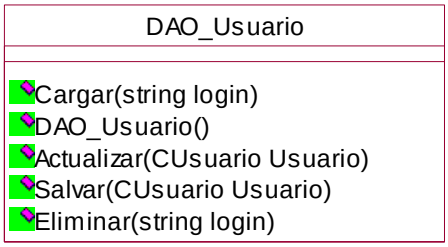


Fig.2.20. Clase DAO_Usuario.

Nombre	DAO_Usuario	
Descripción	Clase base donde se gestionan objetos y realiza la conexión a la base de datos.	
Método	**DAO_Usuario()**	
Descripción	Constructor sin parámetros.	
Método	**Cargar()**	
Parámetros	**Tipo**	**Descripción**
login	string	El login de un usuario.
Descripción	Devuelve un objeto CUsuario de la BD.	
Método	**Actualizar()**	
Parámetros	**Tipo**	**Descripción**
usuario	CUsuario	Se crea un objeto de tipo CUsuario.
Descripción	Actualiza un objeto CUsuario en la BD.	
Método	**Salvar()**	

Parámetros	**Tipo**	**Descripción**
usuario	CUsuario	Se crea un objeto de tipo CUsuario.
Descripción	Guarda un objeto CUsuario en la BD.	
Método	**Eliminar()**	
Parámetros	**Tipo**	**Descripción**
login	string	Se entra el login del usuario.
Descripción	Elimina un objeto CUsuario en la BD.	

Tabla 2.30. Descripción de la clase DAO_Usuario.

En el paquete MAP están las clases que se generan al ser mapeadas las tablas de la base de datos, donde por cada tabla se generan tres clases. A continuación se hace énfasis a un ejemplo de una tabla mapeda de la BD, ver figura 2.21.

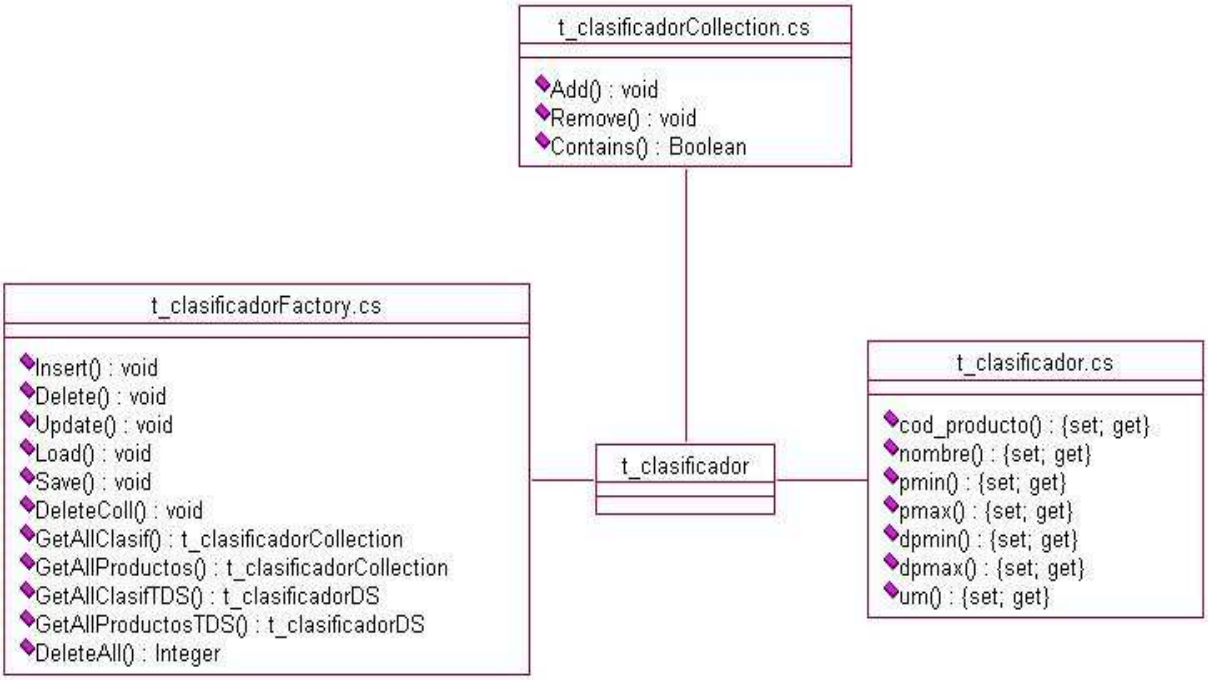


Fig. 2.21. Ejemplo de tabla mapeada.

2.5.3 Implementación de la CAD

En el desarrollo de la implementación de la capa de acceso a datos se tuvo presente la utilización de los diferentes componentes de prototipos .dll o librerías, de enlace dinámico; las cueles son: ONE.Clases.dll, DAO.dll, MAP.dll y EnterpriseNonServiceLib.dll.

En ONE.Clases, se recogen las clases persistentes; para acceder a los datos se hace necesaria la utilización de la librería DAO, que son las clases que proporcionan a la capa de negocio el acceso a la base de datos esta a su vez utiliza las librerías MAP y EnterpriseNonServiceLib, las cuales se encargan del mapeo y el acceso físico a la base de datos, ver figura 2.22.

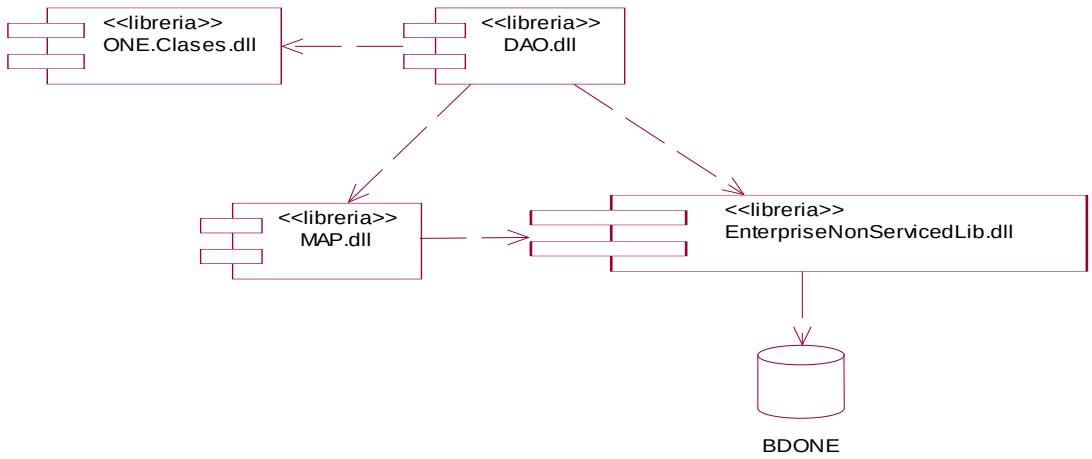


Fig.2.22. Diagrama de Implementación de CAD.

2.6 Conclusiones

Se crearon los diversos modelos de diseño, de implementación, así como también las descripciones de cada entidad o clase necesaria para dichos modelos, logrando obtener finalmente la construcción de la base de datos propuesta, a través del gestor de base de datos seleccionado para ello, en este caso Microsoft SQL Server 2000.

Además, se analizó e implementó la estructura de la capa de acceso a datos utilizando la herramienta TierDeveloper.

CAPITULO 3 VALIDACION DE LOS DATOS EN LA BASE DE DATOS

Introducción

En el presente capítulo se hará referencia a la validación de los datos, basado en la implementación de la base de datos, teniendo cuenta la descripción de la integridad relacional de datos, la normalización de la base de datos, al análisis de la redundancia de la información que la conforma, además de hacer alusión a las políticas de seguridad aplicadas.

Se plantean otros aspectos de importancias tales como la validación funcional según el llenado voluminoso e inteligente de la base de datos, además de la realización de las pruebas de cargas, permitiendo mejorar la ejecución de las consultas (querys) y el procesamiento de los datos, con el objetivo de alcanzar un óptimo funcionamiento de la BD.

3.1 Validación teórica del diseño:

3.1.1 Integridad

La implementación de la integridad relacional en una BD consiste en establecer las reglas de consistencia correspondientes a los datos requeridos de las tablas, el chequeo de los valores válidos y únicos, así como la integridad de las llaves primarias y foráneas.

- **Tipos de Restricciones de Integridad en BD Relacionales.**

  - **Datos Requeridos:** Establece que una columna tenga un valor no nulo. Se define efectuando la declaración de una columna es NOT NULL cuando la tabla que contiene las columnas se crea por primera vez, como parte de la sentencia CREATE TABLE.

  - **Chequeo de Validez:** Cuando se crea una tabla cada columna tiene un tipo de datos y el SGBD asegura que solamente los datos del tipo especificado sean ingresados en la tabla.

  - **La Integridad de entidades**

    Exige que los atributos que componen una llave primaria tengan que estar definidos.

- **La Integridad referencial**

  Los atributos que hacen referencia a una llave primaria de otra relación tienen que tomar uno de los valores definidos para esa llave primaria o estar indefinidos (relaciones padre/hijo). [25]

3.1.1.1 Descripción de las reglas de integridad aplicadas para los diseños de BD:

A continuación se representará la descripción de integridad de los datos a partir de los datos requeridos y chequeos de validez para cada entidad del diseño relacional de la BD.

Tabla	**t_provincia**				
Columna	**No Nulo**	**Llave**	**Tabla Foránea**	**OnUpdate**	**OnDelete**
cod_provincia	✓	PK			
descripción	✓	Unique			
Columnas Unitarias	**cod_provincia** **descripción**				
Chequeo	**Descripción**				
cod_provincia >0	En el código de la provincia no se admiten valores negativos.				

Tabla 3.1. Validación de t_provincia.

Tabla	**t_mercados_provincias**				
Columna	**No Nulo**	**Llave**	**Tabla Foránea**	**OnUpdate**	**OnDelete**
idmercado	✓	PFK	t_mercados	Cascade	Cascade
cod_provincia	✓	PKF	t_provincia	Cascade	Cascade
Columnas Unitarias	**idmercado** **cod_producto**				

Tabla 3.2. Validación de t_ mercados _ provincias.

[25] Obra Citada [4]

Tabla	**t_mercados**				
Columna	**No Nulo**	**Llave**	**Tabla Foránea**	**OnUpdate**	**OnDelete**
idmercado	✓	PK		Cascade	Cascade
tipo	✓	Unique			
descripción	✓	Unique			
Columnas Unitarias	**idmercado** **tipo** **descripción**				

Tabla 3.3. Validación de t_mercados.

Tabla	**t_prod_merc_prov**				
Columna	**No Nulo**	**Llave**	**Tabla Foránea**	**OnUpdate**	**OnDelete**
idmercado	✓	PK		Cascade	Cascade
cod_provincia	✓	Unique		Cascade	Cascade
cod_producto	✓	Unique		Cascade	Cascade
Columnas Unitarias	**idmercado** **cod_provincia** **cod_producto**				

Tabla 3.4. Validación de t_prod_merc_prov.

Tabla	**t_clasificador**				
Columna	**No Nulo**	**Llave**	**Tabla Foránea**	**OnUpdate**	**OnDelete**
cod_producto	✓	PK			
nombre	✓	Unique			
pmin	✓				
pmax	✓				
dpmin	✓				
dpmax	✓				
um	✓				
Columnas Unitarias	**cod_producto** **nombre**				
Chequeo	**Descripción**				
cod_producto > 0	En el código del producto no se admite con valores negativos.				
(pmax >= pmin AND dpmax >= dpmin)	Los valores de los precios máximos (pmax y dpmax) debe ser mayor o igual que los valores de los precios mínimos (pmin y dpmin).				

Tabla 3.5. Validación de t_clasificador.

Tabla	**t_fusion_prod_merc_prov**				
Columna	**No Nulo**	**Llave**	**Tabla Foránea**	**OnUpdate**	**OnDelete**
ano	✓	PK		Restrict	Cascade
mes	✓	PK		Restrict	Cascade
idmercado	✓	PFK	t_mercados	Cascade	Cascade
cod_provincia	✓	PFK	t_provincia	Cascade	Cascade
cod_producto	✓	PFK	t_clasificador	Cascade	Cascade
precio	✓				
Columnas	ano, mes, idmercado				

Unitarias	cod_provincia, cod_producto
Chequeo	**Descripción**
ano>0 AND (ano like '[0-9][0-9][0-9][0-9]')	En el ano no se aceptan valores negativos, además de que en cada carácter que sea insertado, esté en un intervalo de 0 a 9.
(mes >=1 AND mes <=12)	En el mes, no se aceptan valores negativos y se debe insertar en un intervalo desde 1 hasta 12.
Precio (elaboración de un trigger)	El producto contenga cota de precios y que el precio a introducir o modificar debe estar contenido en los intervalos de precios de la tabla t_clasficicador.

Tabla 3.6. Validación de t_fusion_prod_merc_prov.

Para asegurar que en la columna de los precios tuvieran los valores predefinidos, se desarrollo un Trigger, debido a la posibilidad de ocurrir un suceso específico en el momento de manejar un precio, en este caso, se aseguro que un determinado producto según su código, contuviera un rango de precios y que dicho precio debiera estar en los intervalos de precios de la tabla t_clasficicador.

Evento	**Trigger**
Insert o Update	trig_validar_precio_producto

Tabla 3.8. Validación del trigger.

En el trigger llamado validar_precio_producto se rectifica y almacena el precio adecuado, según la cota de precios definida.

De esta manera se cumple la validación de los precios utilizando este trigger en caso que no cumpla con la condición requerida, y se desarrolló con vista a que se almacene la información apropiada.

Tabla	**t_roles**				
Columna	**No Nulo**	**Llave**	**Tabla Foránea**	**OnUpdate**	**OnDelete**
rol	✓	PK			
descripción	✓	Unique			
Columnas Unitarias	**rol** **descripción**				
Chequeo	**Descripción**				
rol > 0	En el rol no se permiten valores negativos.				

Tabla 3.9. Validación de t_roles.

Tabla	**t_usuarios**				
Columna	**No Nulo**	**Llave**	**Tabla Foránea**	**OnUpdate**	**OnDelete**
login	✓	PK		Cascade	Cascade
rol	✓	PFK	t_roles	Cascade	Restrict
passowrd	✓				
Columnas Unitarias	**login** **rol**				

Tabla 3.10. Validación de t_usuarios.

3.1.2 Normalización de la Base de Datos

La normalización de la BD es el proceso mediante el cual se dividen las relaciones insatisfactorias distribuyendo sus atributos en relaciones mucho más pequeñas que presentan características deseables. El proceso de normalización es aplicable a los modelos entidad relación y a los modelos relacionales. Este proceso permite, una vez terminado, que en las BDs relacionales no existan datos repetidos en las tablas, que se proteja la integridad de los datos y además contribuyen a evitar los problemas de actualización de los datos en las tablas. [26]

Para los esquemas de relación propuesto para la base de datos se tuvo en cuenta la normalización donde se aplico al diagrama Modelo Entidad Relación (DER), basado en sus formas normales con el objetivo de analizar la relación de la BD, las dependencias funcionales entre sus atributos, de cumplir con una serie de requisitos para evitar las anomalías en las actualización, inserción, y eliminación, asegurando la consistencia de los datos, con vista a garantizar un adecuado diseño de la BD.

Para normalizar una BD es necesario conocer y aplicar una serie de reglas que van a permitir que el diseño realizado sea correcto. Existen 6 Formas Normales que garantizan un buen diseño de la BD. Básicamente se considera una BD bien diseñada si se encuentra en la Tercera Forma Normal (3FN).

Desarrollado el proceso de normalización, se puede comprobar que satisface la primera forma normal (1FN), ya que se demuestra que no hay presencia de atributos multivaluados, y que sólo se permiten valores atómicos, esto implica que cada tabla contenga para cada atributo un sólo dominio y que tiene un valor único para cada fila y posibilita a que no existen tuplas duplicadas y que cada tupla contiene un valor para cada atributo.

Para lo expuesto en el diseño propuesto tanto para el Módulo de Datos IPC como para el Módulo de Seguridad del Sistema se concluye que todas las relaciones de la base de datos se hallan en 1FN. Con esto se garantiza una atomicidad de todos los atributos, evitando que existan atributos con más de un

[26] Normalización de una base de datos., [26 de marzo de 2007]. Disponible en: http://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_una_base_de_datos.

valor para cada una de las tuplas.

Para la segunda forma normal (2FN) se tuvo en cuenta primeramente que estuviera en 1FN y que existieran dependencias totales, es decir, que todos los atributos no llaves dependen de todos los atributos que componen la llave y no de solo una parte de ellos.
En el análisis de los diseños propuestos se encuentran en 2FN, debido a que todos los atributos no llaves tienen una dependencia total de los atributos llaves, ejemplo de esto en la tabla t_clasificador, para cada código del producto (que es quien la identifica como llave primaria), y de esta llave depende totalmente el nombre del producto, las cotas de precios, la unidad de medida y los valores promedios o inflados según el mercado que se trate, por lo que esto significa que en caso que se elimina uno de los atributos que componen la llave la dependencia pierde validez.

Para la tercera forma normal (3FN), se analizó primeramente que estuviera en 2FN (los atributos no llaves deben depender de toda la llave de manera directa) y no porque dependan de otro atributo no llave de la relación (ejemplo una llave foránea) que a su vez depende de la llave primaria. [27]
Ejemplo de esto en la tabla t_fusion_prod_merc_prov, el atributo precio depende tanto de las llaves primarias como de las llaves primarias compuesta, debido a que para una determinada fusion de una provincia, recogerá además de la fecha, los mercados con sus productos y sus respectivos precios.

A pesar de que no se haya aplicado la forma normal FNBC (Forma Normal de Boyce -Codd) debido a que se alcanzo cumplir con la 3FN, se considero que se desarrolló un adecuado diseño para la BD, manteniendo una conservación de la dependencias y un acople sin perdida de información.

[27] Normalización de bases de datos relacionales., [27 de marzo de 2007]. Disponible en: http://teleformacion.uci.cu/file.php/45/CLASES/Conferencias/Conferencia_4.

3.1.3 Análisis de redundancia de información.

El diseño de una BD debe ser realizado cuidadosamente, procurando cumplir con permitir un fácil acceso a la información. El sistema debe facilitar un alto rendimiento, una buena velocidad de respuesta y una gran consistencia de los datos. Al diseñar una BD se debe tener en cuenta que la información almacenada ocupará irremediablemente un espacio en memoria. Por ello es de vital importancia eliminar la posibilidad de almacenar datos repetidos ya que adicionalmente podrían conducir a que exista inconsistencia en la información. A este almacenamiento de información repetida se le conoce como redundancia de la información. [28]

Eliminar la redundancia de la información es uno de los objetivos principales de la normalización. Esto no es más que eliminar, o por lo menos tener una mínima redundancia de información en la BD.

Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula. No obstante a lo anterior planteado, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias. En ocasiones resulta mejor perder en redundancia por ganar en simplicidad de las consultas y lograr un mejor tiempo de respuesta.[29]

Después de lograrse la normalización de la base de datos se tuvo en cuenta la eliminación de la redundancia en la información que fuese almacenada, específicamente para el Módulo de Datos IPC, que debido a su complejidad se garantizo eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas.

[28] AVENDAÑO, D. E. P. Base de Datos, [27 de marzo de 2007]. Disponible en: http://www.cs.buap.mx/~dpinto/bd

[29] Sistema de gestión de base de datos., [27 de marzo de 2007]. Disponible en: http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos

3.2 Validación funcional del diseño:

3.2.1 Llenado automático de los datos y las pruebas de cargas en la BD.

Para llevar a cabo el llenado de los datos en las tablas de la BD se utilizó la herramienta EMS SQL Data Generador, con el propósito de almacenar datos aleatorios, teniendo en cuenta la validez de los datos en cada campo de las tablas.

Una vez que la base de datos está completamente construida, se realizarán las llamadas "pruebas de carga". El objetivo de estas pruebas es poder simular una carga de producción real, conexiones del usuario y observar como se comportan la aplicación y la base de datos bajo cargas extremas.

Esto permite solucionar los problemas potenciales de rendimiento, antes de que se ponga la base de datos en producción. El propósito general es ejecutar las consultas y procedimientos almacenados más complejos, con vistas a solucionar los problemas existentes con respecto a las consultas más grandes y que requieren un uso de las transacciones de los datos, para finalmente alcanzar una mayor velocidad en la captura y procesamiento de los datos.

3.3 Conclusiones: Valoración de resultados y propuesta de iteración en el diseño.

Con la implementación de la base de datos, se tuvo presente la integridad de los datos, a través de la validación de los mismos, así como las validaciones funcionales con el llenado de las tablas y pruebas de sus funcionalidades según la carga de dicha BD. Con vista a que se desarrolle un diseño óptimo, que se tengan presentes toda la información que se recibe de las distintas provincias, ante la posibilidad de que en un futuro sean utilizados, se propone una nueva iteración del diseño existente, en el cual debe tenerse en cuenta que se encuentre en las distintas formas normales y que cumplan con las validaciones correspondientes.

CONCLUSIONES

Durante el desarrollo de este trabajo se expuso la necesidad de implementar en base al diseño una base de datos para el Módulo IPC que de soporte a la gestión de los procesos de índice de precios para la automatización de los datos estadísticos de ONE.

Luego del estudio realizado de los procesos de IPC que tienen lugar actualmente en el ONE en aras de lograr informatizar para dicho sector, así como el cumplimiento de de los objetivos y tareas trazados, se arribó a las siguientes conclusiones:

- Se realizó un estudio concreto de las tendencias y tecnologías actuales, permitiendo seleccionar las herramientas adecuadas para el desarrollo de la solución propuesta.

- Se modeló los diagramas relacionales (teniendo en cuenta la representación de la estructura lógica y física) de la base de datos según los módulos propuestos, resaltándose la descripción de cada entidad y sus relaciones

- Se llevo a cabo la implementación de la base datos a través de generar el Script de la estructura física

- Se desarrolló la propuesta de construir una base de datos usando como SGBD a Microsoft SQL Server, para implementar la BD.

- Se diseñó el diagrama de clases persistentes, con sus respectivas descripciones y haciendo uso de la misma se especifico la arquitectura de la capa de acceso a datos.

- Se empleó el TierDeveloper para el acceso a datos, con el mapeo de las tablas de la BD, además de que se generan procedimientos almacenados específicos, además de describirse dichos procedimientos presentes para cada módulo.

- Se tuvo en cuenta la integridad de la información, las validaciones de los campos de cada tabla,

además de desarrollarse triggers para la inserción o modificación de datos, además de implementarse la capa de acceso a datos, verificándose si la información es correctamente almacenada y todo con vista a las funcionalidades principales de la base de datos para el módulos de Datos IPC y Módulo de Seguridad del Sistema.

- Se utilizó el EMS SQL Data Generator encaminada al llenado de datos de las tablas de la base de datos, y se logro realizar pruebas de cargas, con la perspectiva de probar el funcionamiento de la BD.

- Se integró en su uso la plataforma .NET, y en especial el lenguaje C# para desarrollar su conectividad con BD.

Luego de estos procesos de trabajo realizado se considera que se han cumplidos los objetivos planteados y se puede concluir que la Base de Datos para el Módulo de IPC dará solución a la situación problemática planteada y su implantación supondrá una mejora de consideración en la gestión de los procesos de IPC.

RECOMENDACIONES

- Mantener sobre la base de datos un estricto cumplimiento de proceso de mantenimiento y copias de seguridad periódicas, logrando así que se mantenga la fiabilidad y funcionamiento óptimo de la base de datos.
- Lograr la integración a otros módulos o subsistemas de la Oficina Nacional de Estadísticas.
- Posibilitar la migración hacia software libre según las políticas actuales del país y la Universidad de la Ciencias Informáticas, durante la investigación se contacto un gran volumen de documentación y herramientas de desarrollo que existen y las potencialidades que brindan las mismas, para lo cual se propone el uso de herramientas y tecnologías libres para la elaboración de una base de datos para ONE.
- Se recomienda el uso de NHibernate como futura implementación de la capa de acceso a datos, pues permite cambiar fácilmente de gestor lo que le brinda al sistema una mayor independencia del gestor de base de datos.

BIBLIOGRAFIA

- CORPORATION, M. SQL Server 2000, 2007 [Disponible en: http://www.microsoft.com/spanish/msdn/articulos/architectema/tema/sql.asp
- Elmasri, R.; Navathe, S.B. Fundamentos de Sistemas de Bases de Datos. 3ª Edición. Addison-Wesley, Pearson Educación, 2002.
- Elmasri, R.; Navathe, S.B. Sistemas de bases de datos. Conceptos fundamentales. 2ª Edición. Addison-Wesley Iberoamericana, 1997.
- Date C. J. Introducción a los sistemas de Bases de Datos, primera parte. La Habana, Editorial Félix Varela, 2003.
- Date C. J. Introducción a los sistemas de Bases de Datos, segunda parte. La Habana, Editorial Félix Varela, 2003.
- Date C. J. Introducción a los sistemas de Bases de Datos, tercera parte. La Habana, Editorial Félix Varela, 2003.
- Introducción al SQL. Sentencias de definición DDL y restricciones. Catálogo. Disponible en: http://teleformacion.uci.cu/file.php/45/CLASES/Conferencias/Conferencia_6/C6-Introduccion_al_SQL.Sentencias_de_definicion_DDL.pdf
- Sentencias de manipulación de datos (DML): INSERT, UPDATE, DELETE, SELECT. Disponible en: http://teleformacion.uci.cu/file.php/45/CLASES/Conferencias/Conferencia_7/C7_Sentencias_de_manipulacion_de_datos_DML.pdf
- Guía del desarrollador de .NET Framework. Acceso a datos con ADO.NET. 2007. [Disponible en: http://msdn.microsoft.com/library/spa/default.asp?url=/library/SPA/cpguide/html/cpconaccessingdatawithadonet.asp.

- HERRERA, M. Programación en Capas Primera Parte (Capa de Acceso a Datos). Disponible en: http://72.14.205.104/search?q=cache:Tnh6NZ1d6x4J:jmhogua.blogspot.com/2007/01/programacin-en-capas-primera-parte-capa.html+Capa+de+Acceso+a+Datos+en+C%23&hl=es&ct=clnk&cd=1&gl=cu&lr=lang_es
- QUIRQUE, G. Introducción a SQL Server 2000, 2006. [Disponible en: http://www.palermo.edu.ar/ingenieria/downloads/Introduccion_a_SQL_Server_2000.ppt
- ROJAS, R. G. A. Seguridad en SQL Server 2000, 2004. [Disponible en: http://www.informatizate.net/articulos/seguridad_en_sql_server_2000_30052004.html
- ALGORRY, I. A. M. Como Asegurar la Disponibilidad de los Datos en SQL SERVER 2000, 2001. [Disponible en: http://www.sqlmax.com/DatosenSQLSERVER.asp
- SOLANO, A. Seguridad al 100% con SQL Server 2000, 2003. [Disponible en: http://www.netveloper.com/contenido2.aspx?IDC=73_0
- MONOGRAFIAS.COM. Bases de datos. Disponible en: http://www.monografias.com/trabajos11/basda/basda.shtml
- CORPORATION, D. Modelado de bases de datos, 2006. [Disponible en: http://www.danysoft.info/free/model2.pdf
- Cinco pasos al desarrollo rápido con TierDeveloper 4.0. 2006. [Disponible en: http://www.vaitman.com/tr/es/?q=/software/4144.php
- CORPORATION, M. SQL Server 2000 vs. SQL Server 2005. 2007. [Disponible en: http://download.microsoft.com/download/9/7/a/97a913c8-2057-434d-aad5-b409363e1c6e/SQL%20Server%202000%20vs.%20SQL%20Server%202005.pdf
- MICROSOFTTECHNET. Diseño de bases de datos de SQL Server. Guía de arquitectura de referencia. Disponible en: http://www.microsoft.com/latam/technet/articulos/idc/idc5/default.asp

GLOSARIO DE TERMINOS

ONE: Oficina Nacional de Estadísticas.

IPC: Índice de Precio al Consumidor.

SQL: Structured Query Language. Lenguaje estándar de comunicación con bases de datos.

CASE: *Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador.*

PK: llave primaria.

PFK: llave primaria compuesta

BD: Base de datos.

ANSI SQL: Estándar de codificación SQL para sistemas gestores de bases de datos.

APIs: Applications Programing Interfaces. Conjunto de subrutinas que aportan los sistemas operativos a los programas de aplicación para el acceso a los recursos y servicios prestados por el ordenador.

XML: Extensible Markup Language (Lenguaje extensible de etiquetas) Es un meta-lenguaje que permite definir lenguajes de marcado adecuado a usos determinados. Se propone como lenguaje de bajo nivel (a nivel de aplicación, no de programación) para intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, y casi cualquier cosa que se pueda pensar.

IBM: International Business Machines.