# Universidad de las Ciencias Informáticas

**Título: Diseño e Implementación de la Base de Datos de los Subsistemas de Inventario y Facturación del Sistema Cedrux.**

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

**Autores:** Rolando Piñeiro Vilaú.

William Alexander de Paz Zaldivar

**Tutor:** Lic. Arismayda Dorado Risco.

Ciudad de la Habana, Mayo 2009

# Agradecimientos

De William:

A mis queridos padres por estar junto a mí en la primera línea de combate a lo largo de toda mi vida, por apoyarme en todas las cosas aún cuando no estuviera actuando de la mejor forma. A mi mamá por sus llamadas día a día, por su amor incondicional, por ser como la catalogo “la mejor madre del mundo”, por preocuparse por mí como si aún fuese su niño chiquito, y querer cada noche saber como había pasado el día.

A mi papá por sus consejos a tiempo, por su cariño, por sus regaños cuando no actuaba con la responsabilidad adecuada, por ser padre en las buenas y en las malas dándome la educación que merece un hijo y el amor que siempre esperé de el.

A mi hermanita linda por quererme tan sinceramente, por escucharme cuando lo necesité, por querer siempre seguir mi ejemplo en la vida, por su inteligencia y madurez a pesar de su temprana edad, por saber que aún cuando tuviéramos las peleas propias de familia nunca saldrían a flote cosas que dañaran nuestro amor como hermanos.

A mi querida novia Lilianne por estar donde siempre la necesité, dispuesta a dejarlo todo por mi y apoyándome en lo que fuese, por saber decirme las cosas tal y como eran cuando no me daba cuenta, por su ayuda en todos estos años que llevo a su lado, por saber que en las buenas y en las malas el amor que sentíamos sobrepasaba cualquiera de las barreras posibles. Por amarme infinitamente y brindarme sin pedir nada a cambio todo esto hermoso que aún compartimos.

A mis abuelitos quienes me aportaron sabiduría y muchas enseñanzas; a mi abuela Delia y mi Abuela María.

A mi familia en general porque desde donde estuvieran consagraron tiempo para saber de mí, para preocuparse por como estaba, por que cosas necesitaba y realmente eso siempre fue importante.

A mis amigos de toda la vida que desde lejos se preocupaban por mí, a los buenos amigos de la Universidad que nunca olvidaré y de quien aprendí muchísimas cosas para ser mejor persona cada día.

A mi compañero de tesis el Rolo por aguantarme malas caras y apoyarme en todas las cosas, por ser además de compañero de tesis amigo sincero, hermano incondicional con quien siempre conté para lo que fuese.

En general a toda la gente que me quiere y que espero no haber defraudado.

De Rolando:

Comenzaré agradeciendo a mi familia que siempre ha confiando en mis decisiones y me han apoyado en todo.

A mi mamá por su preocupación por mis cosas, por mi cada vez que salía de casa camino a la UCI, por ser mi gordita que siempre a querido lo mejor para sus hijitos.

A mi papá que siempre ha estado tan orgulloso de mí y que ha sido capaz de enseñarme tantas cosas útiles e importantes en la vida.

A Tata por ser esa hermana tan dedicada y completa para mí, por ser ella la que siempre se ocupó y preocupó por todas mis cosas incluyendo las más caprichosas, por ser la razón de lo que soy hoy y de lo que llegaré a ser.

A mi novia por todo el aguante de todo este tiempo que hemos estado separados, por su comprensión, por sus llamadas que bien caras que salen, por darme todos los puntos de vista de las cosas ya sean buenas y malas. Por ser esa mujer que siempre en momentos buenos o difíciles para mí estuvo a mi lado. Por ser tan especial para mí.

A Noha y al flaco que siempre estuvieron al tanto de mis cosas, a Lucia que siempre me aconcejaba sobre mis pruebas y el estudio, a Toni que siempre me preguntaba de todos los quehaceres técnicos de aquí y de como podíamos modernizar la carpintería.

A Arismayda por ser para mí aquella amiga desinteresada, que siempre me ha malcriado y que yo siempre voy a querer mucho.

A todos mis amigos que los quiero un mundo, que los voy a echar de menos pero siempre van a estar conmigo porque nunca los voy a olvidar.

Y para terminar que no por último deja de ser importante a William que ha sido para mí un hermano con el cual siempre conté, pedí consejos, fui al gym; fue la parte de mi vida regada, fiestera, malcriada, orgullosa, en fin juntos somos los mejores.

# Dedicatoria

De William:

Dedico mi trabajo de tesis a mis queridos padres, a mi hermanita linda y a mi novia Lilianne.

De Rolando:

Dedico este trabajo a mi papá, a mi mamá, a mi hermana y a mi novia.

# Resumen

En este trabajo se realiza un acercamiento a los Sistemas de Bases de Datos en general, abordando temáticas como la arquitectura de los mismos; con las metodologías de diseño y las tendencias actuales. Además se abordan temáticas como la Integridad de la información, la seguridad de las Bases de Datos (BD), la normalización y desnormalización. Se especifican las diferentes herramientas que se usan para el diseño e implementación de la BD de los Subsistemas de Inventario y Facturación, así como la descripción de las tablas y atributos más importantes que conforman cada uno de los esquemas en los cuales fueron divididos los subsistemas anteriormente mencionados. La BD se implementó teniendo en cuenta conceptos como los de multientidad y multimoneda, conjuntamente con los requisitos funcionales y no funcionales brindados por los clientes, proceso que conllevó a una integración de datos entre los esquemas del Sistema Cedrux, siguiendo nomenclaturas y estándares definidos por el equipo de arquitectura del proyecto. Además se realizó una validación teórica y funcional del diseño.



**Palabras claves:** Sistemas de Bases de Datos, Bases de Datos, Inventario, Facturación.

# Tabla de Contenidos

# Índice de Figuras

# Introducción

En la actualidad la economía mundial está regida mayormente por los países más desarrollados en casi todas las esferas económicas, y para su propio desenvolvimiento empresarial usan técnicas y metodologías de marketing y negocio que respaldan su constante aumento de ganancias y capital.

La intensidad en la competencia global ha creado un ambiente volátil en los negocios, lo cual implica que las empresas se preocupen por tiempos de respuesta más rápidos en el desarrollo de nuevos productos y órdenes de entrega al cliente, satisfacción del cliente, diseño de productos y servicios personalizados, reducción en los costos, productos consistentes, órdenes y procesos de pago simplificados a clientes multinacionales. (Figueroa, 1998)

Es por ese motivo que el entorno empresarial actual se caracteriza por la revolución que experimenta y por el aumento de la competitividad. Estos factores hacen imprescindible llevar a cabo una gestión empresarial adecuada y conocer las funciones en que pueden fundamentarse las estrategias competitivas de las empresas.

El desarrollo del mundo del negocio implica la necesidad de sistemas digitales que permitan un mejor manejo de la gestión empresarial, es por esto que surgen los Sistemas de Planificación de Recursos de Empresas (ERP).

**Definición de Sistema ERP:**

Podemos definir por ERP (Enterprise Resource Planner) a los sistemas de planificación de recursos empresariales que integran y manejan muchas de las prácticas de los negocios asociados con las operaciones de producción y los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios, son parte del conjunto de sistemas de información gerencial que permiten tener un control de la empresa por sus directivos en tiempo. (González Brito, 2006)

**Antecedentes generales:**

- En nuestro país no existe un Sistema Informático Integral de Gestión Empresarial totalmente compatible con las necesidades de nuestras empresas, y que generalice todos los requerimientos

más importantes que son la funcionalidad, interoperabilidad y seguridad que espera el gobierno cubano.

- Existencia de numerosos ERP extranjeros que aumentan cada día los gastos en instalación, mantenimiento y capacitación del personal.
- Dichos ERP no suplen con todas las necesidades. Debido a que las empresas cubanas tienen características en su proceso de gestión empresarial que son muy propios de la economía de nuestro país.
- Sistemas informáticos desarrollados sobre plataformas envejecidas y con poco o ningún criterio de seguridad y auditoría (en el orden técnico y funcional).
- La mayoría fueron desarrollados para un ambiente multiusuario, casi ninguno bajo conceptos de informática multicapa y distribuida en la red. Lo más general es desarrollos sobre arquitectura Cliente-Servidor de base de datos.
- Son productos que se caracterizan por abordar solamente partes del problema de la gestión de la empresa o la unidad presupuestada y no soportan mecanismos estándares de integración con otras aplicaciones.

Los antecedentes generales muestran como hoy en día los Sistemas ERP en Cuba no resuelven los problemas de la gestión empresarial. Es por ello que el gobierno le da a la Universidad de las Ciencias Informáticas (UCI) la tarea de la creación de un proyecto para desarrollar un Sistema ERP llamado Cedrux que recoja las ventajas de los sistemas ya previamente usados, y que además permita mayores funcionalidades en el terreno empresarial.

Se necesita para la creación de Cedrux tener en cuenta que existen varios procesos de vital importancia dentro de cualquier entidad que no pueden dejarse a un lado a la hora de desarrollar un sistema de este tipo; estos procesos están enmarcados en el término de Logística que es el conjunto de medios y métodos necesarios para llevar a cabo la organización de una empresa, o de un servicio, especialmente de distribución. Dentro de todos estos procesos se hace énfasis en la gestión de inventario y facturación, pilares claves que permiten mayor productividad y garantía en los servicios de cualquier empresa tanto nacional como internacional.

El inventario constituye un activo fundamental dentro de la mayoría de las organizaciones; de el dependen varias funciones como son las de producción, ventas, compras, financiación, llegando a ser parte medular de un negocio.

Además de todo lo anterior explicado también es fundamental tener en cuenta que el proceso de facturación es significativo, pues el mismo recoge un conjunto de operaciones en las que se especifica la totalidad de las ventas de una empresa en un determinado período de tiempo, estas en cualquiera de las formas en las que se analice son imprescindibles para cualquier entidad, pues con el aumento de las mismas se provee de resultados a todo un período organizativo y de producción, debido a que el objetivo de cualquier empresa es la obtención de ganancias y las mismas solo se logran con un nivel de ventas que lo abalen, demandando de esta forma que los recursos utilizados siempre sean menores que las ganancias obtenidas.

Dada la necesidad de automatizar los procesos antes mencionados se hace imprescindible la gestión de un soporte técnico de nivel tres adecuado, definir herramientas para establecer reglas de precios, y de reabastecimiento (Gestión de Stocks), conjuntamente con la utilización de herramientas para la creación de recuperaciones dinámicas y formas de almacenamiento que permitan el acceso al historial de los datos.

Según lo planteado anteriormente se puede definir:

**Problema Científico:**

¿Cómo soportar los datos generados por los Subsistemas de Inventario y Facturación, integrados al Sistema Integral de Gestión Cedrux?

**Objeto de estudio:**

Los procesos de control de inventarios y facturación.

**Objetivo General:**

Diseñar e implementar la Base de Datos de los Subsistemas de Inventario y Facturación del Sistema Cedrux.

**Campo de acción:**

Los procesos de almacenamiento de los Subsistemas de Inventario y Facturación.

**Objetivos específicos:**

- Caracterizar los procesos de almacenamiento de inventario y facturación, así como las herramientas a utilizar.
- Diseñar e implementar la Base de Datos.
- Validar teórico y funcionalmente el diseño de la Base de Datos.

**Para lograr los objetivos antes mencionados se proponen las siguientes tareas:**

- Análisis de los procesos y la documentación generada por los analistas.
- Evaluación de la experiencia nacional e internacional de la aplicación de los Subsistemas de gestión de inventario y facturación.
- Métodos y herramientas para el diseño, implementación y validación de la BD.
- Estudio de los artefactos y actividades asociadas al rol de Diseñador de BD.
- Diseño del Modelo Físico de Datos.
- Implementación de la BD en el gestor PostgreSQL.
- Análisis de la integridad de la BD.
- Análisis de redundancia de información en la BD.
- Normalización de la BD.
- Implementación de disparadores y funciones de la BD.
- Análisis de la seguridad de la BD.
- Análisis de trazabilidad de la acciones.
- Búsqueda de herramientas para pruebas de carga intensiva.

El presente documento fue estructurado en 3 capítulos. A continuación se dará una breve descripción de los contenidos que serán abordados en los mismos.

**Capítulo 1:** En este capítulo se realiza la fundamentación teórica de todo el trabajo, además de la descripción de las diferentes herramientas utilizadas para desarrollar el mismo.

**Capítulo 2:** En este capítulo se da la propuesta de solución del diseño e implementación de la BD de los Subsistemas de Inventario y Facturación, además de otros elementos importantes como son los requisitos funcionales tenidos en cuenta para el diseño y algunos documentos que rigieron todo el trabajo.

**Capítulo 3:** En este capítulo se muestran los elementos por los que se validó teórico y funcionalmente el diseño de la BD, además de la valoración de los resultados de este proceso de validación.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

En el presente capítulo se da una breve reseña del tema de las Bases de Datos en la actualidad, las tendencias tecnológicas de las mismas y las metodologías de diseño con sus fases. Se abordan también temas como la seguridad, integridad de la información y normalización. Además se especificarán las herramientas empleadas para el diseño e implementación de la BD de los Subsistemas de Inventario y Facturación del Sistema Cedrux.

## 1.2 Introducción a las Bases de Datos

Comúnmente se usa el término bases de datos sin saber del todo su significado, que no es más que la persistencia de un grupo de información en un lugar determinado, con un soporte determinado y a la cual se pueda acceder. Las antiguas bibliotecas, los archivos y todo este tipo de documentación que siempre se consolidó en papel, y se almacenó en grandes estantes y locales donde la accesibilidad se hacía un poco engorrosa, sin quitar méritos a los niveles de organización eran también un tipo de bases de datos. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

### 1.2.1 Componentes de una Base de Datos

- **Hardware**: constituido por dispositivos de almacenamiento como discos, tambores, cintas, etc.
- **Software**: que es el Sistema Gestor de Bases de Datos o SGBD.
- **Datos**: los cuales están almacenados de acuerdo a la estructura externa y van a ser procesados para convertirse en información.

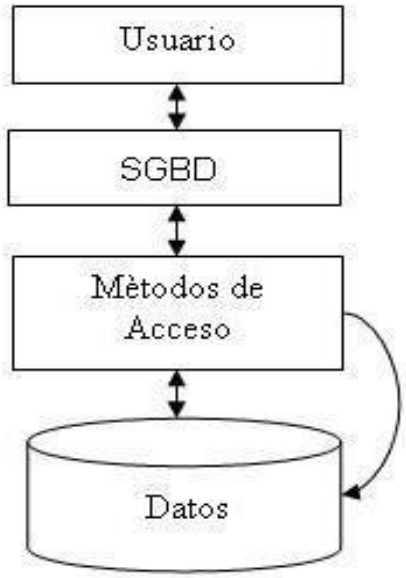



*Figura 1 Flujo de acceso a los datos en una BD.*

### 1.2.2 Tipos de Usuarios de las Bases de Datos

- **Usuario Final**: es la persona que utiliza los datos, esta persona ve datos convertidos en información.
- **Desarrollador de Aplicaciones**: es la persona que desarrolla los sistemas que interactúan con la Base de Datos.
- **Administrador de la Base de Datos**: es la persona que asegura integridad, consistencia, redundancia y seguridad de la Base de Datos, además es quien se encarga de realizar el mantenimiento diario o periódico de los datos.

## 1.3 Sistemas Gestores de Bases de Datos (SGBD)

Los Sistemas Gestores de Bases de Datos, son aplicaciones que permiten a los usuarios definir, crear y mantener la BD y proporciona un acceso controlado a la misma. Los SGBD son las aplicaciones que interactúan con los usuarios de los programas de aplicación y las BD.

Algunos de los SGBD más conocidos son: SQL, PostgreSQL, DB2, SLQ/DS, ORACLE, INGRES, INFORMIX, SYBASE, PARADOX, DBASE, ACCESS, FOXPRO, R, RM/T y RM/V2.
Entre los principales SGBD propietarios más potentes se encuentra Oracle y Microsoft SQL Server, y de software libre se tienen opciones muy completas como MySQL y PostgreSQL.

## 1.4 Criterios de clasificación de los Sistemas de Gestión de Bases de Datos

Para clasificar los SGBD se usan tres criterios. El primero de ellos es el uso de modelos lógicos los cuales son empleados con mayor frecuencia en los SGBD comerciales actuales, este tipo de clasificación por modelos lógicos se desglosa en cuatro tipos diferentes.

La mayoría de los SGBD actuales están basados en el modelo relacional, mientras que los sistemas más antiguos estaban basados en el modelo de red o el modelo jerárquico. Estos dos últimos modelos requieren que el usuario tenga conocimiento de la estructura física de la base de datos a la que se accede, mientras que el modelo relacional proporciona una mayor independencia de datos. Se dice que el modelo relacional es declarativo (se especifica qué datos se han de obtener) y los modelos de red y jerárquico son navegacionales (se especifica cómo se deben obtener los datos).

### 1.4.1 Modelo relacional

Se basa en el concepto matemático denominado “relación”, que gráficamente se puede representar como una tabla. En el modelo relacional, los datos y las relaciones existentes entre los datos se representan mediante estas relaciones matemáticas, cada una con un nombre que es único y con un conjunto de columnas.

En este modelo la base de datos es percibida por el usuario como un conjunto de tablas. Esta percepción es sólo a nivel lógico (en los niveles externo y conceptual de la arquitectura de tres niveles), ya que a nivel físico puede estar implementada mediante distintas estructuras de almacenamiento.

### 1.4.2 Modelo de red

En el modelo de red los datos se representan como colecciones de registros y las relaciones entre los datos se representan mediante conjuntos, que son punteros en la implementación física. Los registros se

organizan como un grafo: los registros son los nodos y los arcos son los conjuntos. El SGBD de red más popular es el sistema IDMS.

### 1.4.3 Modelo jerárquico

Es un tipo de modelo de red con algunas restricciones. De nuevo los datos se representan como colecciones de registros y las relaciones entre los datos se representan mediante conjuntos. Sin embargo, en el modelo jerárquico cada nodo puede tener un solo padre. Una base de datos jerárquica puede representarse mediante un árbol: los registros son los nodos, también denominados segmentos, y los arcos son los conjuntos. El SGBD jerárquico más importante es el sistema IMS.

### 1.4.4 Modelo orientado a objetos

Define una base de datos en términos de objetos, sus propiedades y sus operaciones. Los objetos con la misma estructura y comportamiento pertenecen a una clase, y las clases se organizan en jerarquías o grafos acíclicos. Las operaciones de cada clase se especifican en términos de procedimientos predefinidos denominados métodos. Algunos SGBD relacionales existentes en el mercado han estado extendiendo sus modelos para incorporar conceptos orientados a objetos. A estos SGBD se les conoce como sistemas objeto-relacionales.

### 1.4.5 Número de usuarios a los que da servicio el sistema

Este es el segundo criterio de clasificación de los Sistemas Gestores de Bases de Datos. El número de usuarios a los que brinda servicio el sistema se desglosa en dos tipos.

- **Sistemas monousuario**

Sólo atienden a un usuario a la vez, y su principal uso se da en los ordenadores personales.

- **Sistemas multiusuario**

La mayor parte de los SGBD usan este sistema, ya que el mismo atiende a varios usuarios al mismo tiempo.

### 1.4.6 Número de sitios en los que está distribuida la Base de Datos

Este es el tercer criterio de clasificación de los SGBD que se desglosa en dos tipos.

#### 1.4.6.1 Centralizados

Sus datos se almacenan en un solo computador. Los SGBD centralizados pueden atender a varios usuarios, pero el SGBD y la BD en sí residen por completo en una sola máquina.

#### 1.4.6.2 Distribuidos

La BD real y el propio software del SGBD pueden estar distribuidos en varios sitios conectados por una red. Muchos SGBD distribuidos emplean una arquitectura cliente-servidor y el mismo puede ser de dos tipos diferentes.

- **Distribuidos homogéneos**

Utilizan el mismo SGBD en múltiples sitios. Una tendencia reciente consiste en crear un software para tener acceso a varias bases de datos autónomas preexistentes almacenadas en un SGBD.

- **Distribuidos heterogéneos**

Esto da lugar a los SGBD federados o sistemas multibase de datos en los que los SGBD participantes tienen cierto grado de autonomía local.

## 1.5 Arquitectura de los Sistemas Gestores de Bases de Datos

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

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

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

## 1.6 Metodologías de diseño de Bases de Datos

El diseño de una BD es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas independientemente, utilizando técnicas específicas. Es por esto que el uso de metodologías de diseño de bases de datos es importante pues es una guía con pasos lógicos para lograr un aumento en la calidad del diseño.

Desafortunadamente, las metodologías de diseño de las BD no son muy populares; la mayoría de las organizaciones y de los diseñadores individuales confían muy poco en las metodologías para llevar a cabo el diseño y esto se considera, con frecuencia, una de las principales causas de fracaso en el desarrollo de los sistemas de información. Debido a la falta de enfoques estructurados para el diseño, a menudo se subestima el tiempo o los recursos necesarios para un proyecto de BD, las bases de datos son inadecuadas o ineficientes en relación a las demandas de la aplicación, la documentación es limitada y el mantenimiento es difícil.

Muchos de estos problemas se deben a la falta de una claridad que permita entender la naturaleza exacta de los datos, a un nivel conceptual y abstracto. En muchos casos, los datos se describen desde el comienzo del proyecto en términos de las estructuras finales de almacenamiento; no se da peso a un entendimiento de las propiedades estructurales de los datos que sea independiente de los detalles de la realización. (Batini, et al., 1994)

## 1.7 Fases de diseño de Bases de Datos

Para el diseño de bases de datos se usan 3 fases:

- El diseño conceptual parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la BD. El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

- El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un esquema lógico es una descripción de la estructura de la BD en términos de las estructuras de datos que puede procesar un tipo de SGBD. El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto.

- El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un esquema físico es una descripción de la implementación de una BD en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos.

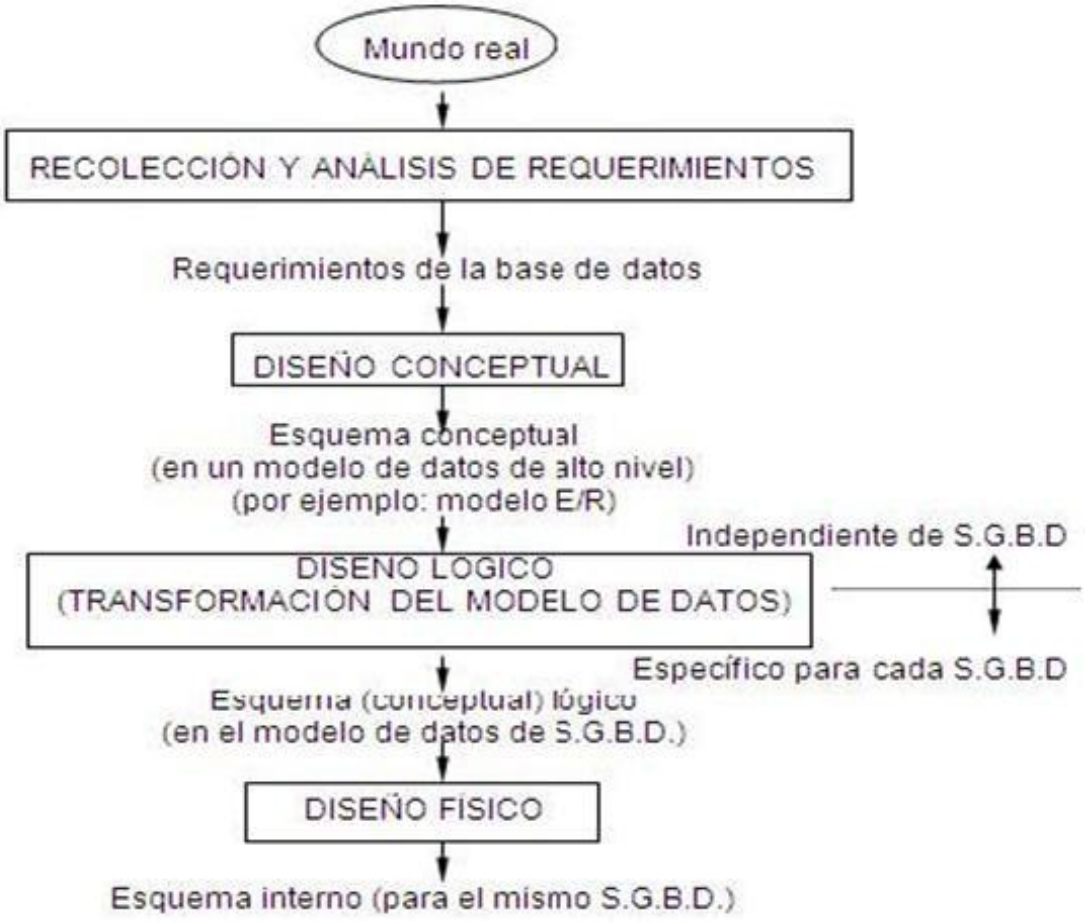



*Figura 2 Pasos para el diseño de una base de datos.*

### 1.8 Tendencias actuales y futuras

La utilización de Bases de Datos como plataforma para el desarrollo de Sistemas de Aplicación en las organizaciones se ha incrementado notablemente en nuestros días. Actualmente las BD cuentan con numerosas funcionalidades de gran importancia para el almacenamiento y la manipulación de la información en las instituciones. Esto trae consigo el desarrollo de nuevas técnicas y herramientas que brindan una solución eficiente a las solicitudes planteadas por los usuarios de las mismas.

Algunas de estas tendencias actuales y futuras de Bases de Datos se citan a continuación.

- La explotación efectiva de la información dará ventaja competitiva a las organizaciones.

- En el futuro la mayoría de las organizaciones cambiarán la forma convencional de manejo de la información al uso de base de datos debido a las ventajas que ofrecen.
- El uso de las bases de datos facilita y soporta en gran medida a los sistemas de información para la toma de decisiones.
- Los lenguajes de consulta permitirán el uso del lenguaje natural para solicitar información de la base de datos, haciendo más rápido y fácil su manejo.
- La Empresa Alemana SAP (Sistemas, Aplicaciones y Productos) ha descubierto una tecnología que podría ser devastadora para Oracle. Al menos, en teoría, los procesos in memory pueden eliminar, en algunos casos, la necesidad de las bases de datos relacionales. La tecnología **in memory** de bases de datos mantiene los datos en memoria en lugar de en un disco. Está indicada para las aplicaciones propias de compañías de telecomunicaciones y financieras, donde la velocidad es crítica. Pero además, algunos analistas de mercado apuntan que podría tener un relevante impacto en SOA, RFID, fabricación y comercio electrónico.

## 1.9 Definición de Integridad

Son restricciones que definen los estados de consistencia de la Base de Datos, esto significa que la BD o los programas que generaron su contenido, incorporen métodos que aseguren que el contenido de los datos del sistema no se rompa, así como las reglas del negocio. La integridad de los datos puede verse afectada añadiendo datos no válidos, modificando datos existentes, que se les asigne un valor incorrecto o eliminando datos que violen alguna regla. La integridad de datos presenta varias restricciones que posibilitan la declaración y comprobación de condiciones para expresar la consistencia, corrección y exactitud de datos almacenados, estas son:

Integridad de dominio, de clave, de entidad, referencial, entre otras.

### 1.9.1 Integridad de los datos

La integridad de los datos no es más que la corrección y completitud de la información en una base de datos. Cuando los contenidos se modifican con sentencias INSERT, DELETE o UPDATE, la integridad de los datos almacenados puede perderse de muchas maneras diferentes. Pueden añadirse datos no válidos a la base de datos, tales como un pedido que especifica un producto no existente.

Pueden modificarse datos existentes tomando un valor incorrecto, como por ejemplo si se reasigna un producto a un documento no existente. Los cambios en la base de datos pueden perderse debido a un error del sistema o a un fallo en el suministro de energía. Los cambios pueden ser aplicados parcialmente, como por ejemplo si se añade un pedido de un producto sin ajustar la cantidad disponible para vender. Una de las funciones importantes de un SGBD relacional es preservar la integridad de sus datos almacenados en la mayor medida posible.

La integridad de los datos esta sustentada sobre cuatro restricciones fundamentales que se explican a continuación.

**Datos Requeridos**: establece que una columna tenga un valor no NULL. Se define efectuando que 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 donde en cada columna tiene un tipo de datos y el SGBD asegura que solamente los datos del tipo especificado sean ingresados en la tabla.

**Integridad de entidad**: Establece que la clave primaria de una tabla debe tener un valor único para cada fila de la tabla; si no, la base de datos perderá su integridad. Se especifica en la sentencia CREATE TABLE. El SGBD comprueba automáticamente la unicidad del valor de la clave primaria con cada sentencia INSERT Y UPDATE. Un intento de insertar o actualizar una fila con un valor de la clave primaria ya existente fallará. Es por esto que en muchas de las tablas de cada uno de los esquemas de la Base de Datos de los Subsistemas de Inventario y Facturación sus llaves se generan de forma automática usando secuencias, garantizando así que no se cometan errores, uniformidad en la creación de la llaves, además de que el principal objetivo de la creación de las mismas está dado por el tema de las réplicas entre los servidores.

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

Eso es lo que se denomina integridad referencial y consiste en que los datos que referencian otros (claves foráneas) deben ser correctos. La integridad referencial hace que el sistema gestor de la base de datos se asegure de que no haya en las claves foráneas valores que no estén en la tabla principal.

## 1.10 Normalización de las Bases de Datos

El proceso de normalización de las BD consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad relación al modelo relacional.

Las bases de datos relacionales se normalizan para:

- Evitar la redundancia de los datos.
- Evitar problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

El proceso de Normalización define tres Formas Normales (FN) principalmente:

- Primera Forma Normal (1FN)
- Segunda Forma Normal (2FN)
- Tercera Forma Normal (3FN)
- Forma Normal de Boyce-Codd (FNBC)

Las formas normales someten el esquema de relación a una serie de pruebas para certificar si pertenecen a una cierta forma normal, cuando el esquema relacional esté en FNBC, es que ya el esquema se llevó a 3FN, y para estar en 3FN el esquema se tuvo que haber llevado a 2FN y así sucesivamente, por lo que se puede decir que la FNBC es una de las FN más deseadas para un buen diseño de BD. No obstante, no siempre que el diseño de BD esté en la FNBC podemos garantizar que sea un diseño idóneo de los datos.

### 1.10.1 Desnormalización de Bases de Datos

La desnormalización es el proceso al cual se someten los diseños de bases de datos con el fin de ganar en rendimiento. Una base de datos con la medida justa de desnormalización reduce el número de tablas que deben combinarse sin dificultar en exceso el proceso de actualización. Entonces, tendrá consecuencias de redundancia de datos, pero mejorará el rendimiento ya que se evitarían uniones y subconsultas, pero es importante tener en cuenta que sólo se debería implementar si presenta notables ventajas de performance.

El punto de vista teórico, marca la premisa en general, de evitar la redundancia de datos para obtener un diseño normalizado y optimizado. Pero si se parte desde la realidad cotidiana en la cual se desenvuelve el negocio, en la práctica, dependiendo del contexto en que se esté trabajando a veces se debe desestimar parte de la normalización, para lograr (o mejorar) el comportamiento de un sistema, para lograr un mejor performance en el funcionamiento del sistema. Se adiciona la variable costos en un ambiente real, que muchas veces tiene un impacto más ponderante que el que un buen diseñador podría desear.

Cabe destacar que a decisión de incluir redundancia en los datos (por más mínima que esta sea) debe ser perfectamente estudiada y analizada, y por otro lado los beneficios deberían ser claramente notorios para que justifique la acción.

## 1.11 Normalización de árboles en el Sistema Cedrux

Las tablas de una base de datos correlativa no son jerárquicas (como en XML), simplemente son listas llanas. Los datos jerárquicos tienen una relación padre-hijo, los cuales no se representan en la tabla de la base de datos correlativa. Los datos jerárquicos son una colección de datos donde cada artículo tiene un solo padre, puede tener muchos o ningún hijo (con la excepción del nodo raíz, este no tiene ningún padre). Estos datos pueden tener múltiples aplicaciones en la base de datos. (Lichilín Ríos, 2008)

## 1.12 Seguridad de las Bases de Datos

Las bases de datos hoy en día son de gran importancia en cada una de las aplicaciones que la requieran, es por esto que la seguridad para su acceso y manejo de la información se restringe en una jerarquía de usuarios. Los SGBD permiten mantener la seguridad mediante el establecimiento de claves para identificar al personal autorizado a utilizar la base de datos. Las autorizaciones se pueden realizar a nivel

de operaciones, de modo que un usuario puede estar autorizado a consultar ciertos datos pero no a actualizarlos. En el caso específico el gestor de bases de datos PostgreSQL presenta una serie de características que facilita el trabajo para tener un alto nivel de seguridad.

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

- Protección de los ficheros de la base de datos, ya que todos los ficheros almacenados en la base de datos están protegidos contra escritura por cualquier cuenta que no sea la del súper usuario de Postgres.

- Las conexiones de los clientes se pueden restringir por dirección IP y/o por nombre de usuario mediante el fichero pg_hba.conf situado en PG_DATA.

- Las conexiones de los clientes pueden ser autentificadas mediante otros paquetes externos.

- A cada usuario de Postgres se le asigna un nombre de usuario y (opcionalmente) una contraseña. Por defecto, los usuarios no tienen permiso de escritura a bases de datos que no hayan creado.

- Los usuarios pueden ser incluidos en grupos, y el acceso a las tablas puede restringirse en base a esos grupos.

## 1.13 Lenguaje de Modelado Unificado (UML)

El Lenguaje de Modelado Unificado (UML) es usado para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software. UML proporciona una forma estándar de escribir los planos de un sistema, cubriendo tanto los elementos conceptuales, procesos del negocio y funciones del sistema, como elementos concretos: las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes de software reutilizables. Este no garantiza el éxito de los proyectos pero sí mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios.

## 1.14 Herramientas utilizadas

Para el diseño e implementación de la Base de Datos de los Subsistemas de Inventario y Facturación del Sistema Cedrux se utilizaron diferentes herramientas que permiten la seguridad, consistencia e integridad de la información.

### 1.14.1 Visual Paradigm

Para diseñar el Modelo Entidad Relación de la Base de Datos se utilizó la Herramienta CASE (Computer-Aided Software Engineering) Visual Paradigm que ofrece un entorno de creación de diagramas para UML. El diseño es centrado en casos de uso y enfocado al negocio, lo que permite generar software de gran calidad. Esta herramienta usa un lenguaje estándar común para todo el equipo de desarrollo que facilita la comunicación. Tiene capacidad para la ingeniería directa e inversa en Java, C++, PHP, entre otros lenguajes y disponibilidad de múltiples versiones para cada necesidad. Es multiplataforma y muy útil para la generación de código fuente en PHP. Tiene la capacidad de crear el esquema de clases a partir de una Base de Datos y crear la de Base de Datos a partir del esquema de clases. Incorpora el soporte para trabajo en equipo, proporcionando que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros de equipo.

### 1.14.2 PostgreSQL

El Sistema Gestor de Bases de Datos PostgreSQL está establecido para su uso en el Sistema Cedrux por las normativas de la dirección del proyecto, la versión que se usa del mismo es la 8.3.

PostgreSQL es uno de los Sistemas Gestores de Bases de Datos más utilizados por la comunidad de software libre por las razones siguientes:

Cumple con las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) y soporta el lenguaje común de acceso a los datos: SQL.

Es multiplataforma y posee buenas interfaces de instalación y administración. Aproxima los datos a un modelo Objeto-Relacional, y es capaz de manejar completas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multiversión, soporte multiusuario, transacciones y optimización de consultas.

Está basado en el proyecto POSTGRES, de la universidad de Berkeley. Es una derivación libre (OpenSource) de este proyecto, y utiliza el lenguaje SQL92/SQL99. Fue el pionero en muchos de los

conceptos existentes en el Sistema Objeto-Relacional actual, incluido más tarde en otros sistemas de gestión comerciales. Incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores e integridad transaccional.
Lleva más de una década de desarrollo, siendo hoy en día un sistema bastante avanzado, que tiene soporte nativo para los lenguajes de programación: C, C++, Java, Python, PHP y muchos más. Se encuentra bajo la licencia BSD (Berkeley Software Distribution).

**Entre otras características que presenta se encuentran las siguientes:**

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. Además permite la creación de tipos de datos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

### 1.14.3 Herramientas de desarrollo

Para la administración de la Base de Datos se usó la herramienta de desarrollo EMS SQL Manager 2005 para PostgreSQL, la misma brinda una serie de servicios que a continuación se explican.

#### 1.14.3.1 EMS SQL Manager 2005 para PostgreSQL

SQL Manager 2005 para PostgreSQL funciona con cualquier versión de PostgreSQL superior a la 8.0 y soporta las últimas características del mismo incluyendo la de los nombres de los argumentos en las funciones, los espacios de tablas y otras más; el mismo ofrece un gran número de potentes herramientas para usuarios experimentados como por ejemplo Diseñador Visual de Bases de Datos (Visual Database

Designer), Constructor de Consultas Visuales (Visual Query Builder), y el potente editor BLOB que satisface todas las necesidades del usuario.

**Características:**

- SQL Manager 2005 para PostgreSQL consta de una interfaz amigable y remodelada.
- Rápido manejo y navegación de la Base de Datos.
- Avanzadas herramientas de manipulación de datos.
- Soporte completo de versiones de PostgreSQL superior a la 8.0.
- Explorador de Bases de Datos mejorado para un mejor manejo de los objetos.
- Administración efectiva de la seguridad.
- Acceso a Servidores en PostgreSQL a través del protocolo HTTP.

### 1.14.4 Mapeador de Doctrine (Doctrine Mapping)

Para la generación de las bases y los dominios de la capa de acceso a datos se utilizó un software para el mapeo de todas las clases del diseño, esta herramienta permitió ahorrar tiempo y esfuerzos para realizar esta labor. La herramienta Doctrine Mapping se conecta a la base de datos y permite el mapeo de todas las tablas que se deseen, la única disyuntiva es que no reconoce las relaciones entre las clases y hay que redefinírselas para que el mapeo quede con la calidad requerida.

## 1.15 Conclusiones

Mundialmente los programadores conjuntamente con las grandes compañías, se empeñan en desarrollar Sistemas Gestores de Bases de Datos que sean cada vez más rápidos, eficaces, y que resuelvan los problemas que se plantean a nivel mundial en la demanda, persistencia e integridad de la información que en ellos se administra.

En la actualidad los SGBD han tenido un gran auge, principalmente en el mundo de los ERP, donde los relacionales y las Bases de Datos Distribuidas y Multidimensionales se han convertido en los líderes del desarrollo de estos.

Las técnicas seleccionadas por el equipo de arquitectura del proyecto fueron de gran ayuda para el diseño e implementación de la Base de Datos de los Subsistemas de Inventario y Facturación del Sistema Cedrux; entre ellas se destacó el SGBD PostgreSQL por su gran capacidad para garantizar la seguridad e integridad de los datos.

# CAPÍTULO 2: Propuesta de Solución

## 2.1 Introducción

La solución propuesta esta sustentada sobre la base de todo lo que se expone en este capítulo, en el cual se definen las actividades más importantes que posibilitaron la realización del diseño de la Base de Datos que dio paso a la implementación de la misma.

Exponiéndose los requisitos funcionales más importantes que se utilizaron para la creación del modelo conceptual de los Subsistemas que luego se diseñaron, además de las referencias a varios documentos que rigieron el desarrollo, conjuntamente con aspectos importantes que dan una noción clara de la vista de la arquitectura de datos y de la integración con los demás módulos del proyecto. También se brinda una descripción detallada de las entidades más importantes en cuestión de funcionalidad para los esquemas en los cuales fue dividido el diseño.

## 2.2 Argumentación de los requisitos funcionales y no funcionales del sistema propuesto

Es de vital importancia para la creación de un sistema informático llevar a cabo una elicitación de requisitos adecuada para llegar a acuerdos positivos entre los usuarios finales y los desarrolladores del sistema, es por esto que la captura de requisitos es tan importante para asegurar la calidad del producto final la cual es evaluada por los clientes.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Los no funcionales son propiedades o cualidades que el producto debe tener, permitiendo que el mismo sea atractivo, usable, rápido y confiable.

## 2.3 Requisitos Funcionales del Subsistema de Inventario dividido por módulos

Para el diseño e implementación de la Base de Datos del Subsistema de Inventario se tomaron en cuenta una serie de requisitos funcionales que a continuación se describen brevemente. Para una mayor especificación ver documentos asociados.

El Subsistema de Inventario esta estructurado en once componentes y cada uno de ellos tienen requisitos funcionales asociados. A continuación se muestran los requisitos funcionales de los componentes más importantes para el diseño conceptual.

### 2.3.1 Módulo de Inventario Físico

- Adicionar hoja de inventario.
- Registrar conteo final.
- Confirmar hoja de inventario.
- Adicionar productos.
- Adicionar plan de conteo físico.
- Confirmar plan de conteo físico.
- Evaluar plan de conteo físico (Aprobar/Rechazar).

(Morales Ortiz, et al., 2009)

### 2.3.2 Módulo de Ajuste

- Adicionar ajustes de inventario.
- Confirmar ajustes de inventario.

(Carbonell Vargas, 2009)

### 2.3.3 Módulo de Baja

- Adicionar documento de baja de equipos.
- Confirmar documento de baja de equipos.
- Contabilizar documento de baja de equipos.
- Adicionar número de serie al equipo.

(Carbonell Vargas, 2009)

### 2.3.4 Módulo Conciliación

- Adicionar documento de conciliación.
- Confirmar documento de conciliación.
- Conciliar documento de conciliación por cliente.
- Contabilizar documento de conciliación.

(Oliva Martínez, 2009)

### 2.3.5 Módulo de Despacho

- Adicionar autorizaciones de entrega.
- Adicionar cliente de la autorización de entrega.
- Confirmar autorización de entrega.
- Aprobar autorización de entrega.
- Registrar productos autorizados.
- Generar conduce.
- Contabilizar conduce.
- Generar transferencia.
- Confirmar transferencia.
- Contabilizar transferencia.

(Oliva Martínez, et al., 2009)

### 2.3.6 Módulo de Nomencladores

- Adicionar la descripción de calidad del producto en el nomenclador.
- Adicionar la categoría del producto en el nomenclador.

- Adicionar concepto al nomenclador de conceptos.
- Adicionar productos al nomenclador.

(Borges López, 2009)

### 2.3.7 Módulo de Recepción

- Crear informes de diferencias.
- Registrar lotes del producto.
- Adicionar número de serie del equipo.
- Adicionar productos al informe de recepción.
- Registrar diferencias en el informe de recepción.
- Crear tarjeta de estiba.
- Crear informe de recepción.
- Contabilizar informe de recepción.
- Adicionar productos al informe de diferencias.
- Asignar ubicación al producto.

(Borges López, et al., 2009)

### 2.3.8 Módulo Configuración

- Adicionar estructura de ubicación de un producto.
- Modificar formato de la estructura de ubicación de los productos.
- El requisito funcional Desbloqueo de documento hace referencia a la funcionalidad del sistema para permitir utilizar un documento que está siendo utilizado por otro usuario en determinada situación.

(Oliva Martínez, 2009)

## 2.4 Modelo Conceptual del Subsistema de Inventario

Después de analizar los requisitos funcionales del Subsistema de Inventario conjuntamente con los analistas de sistema, se elaboró el modelo conceptual como artefacto de gran importancia para iniciar el diseño de un modelo relacional de entidades persistentes.

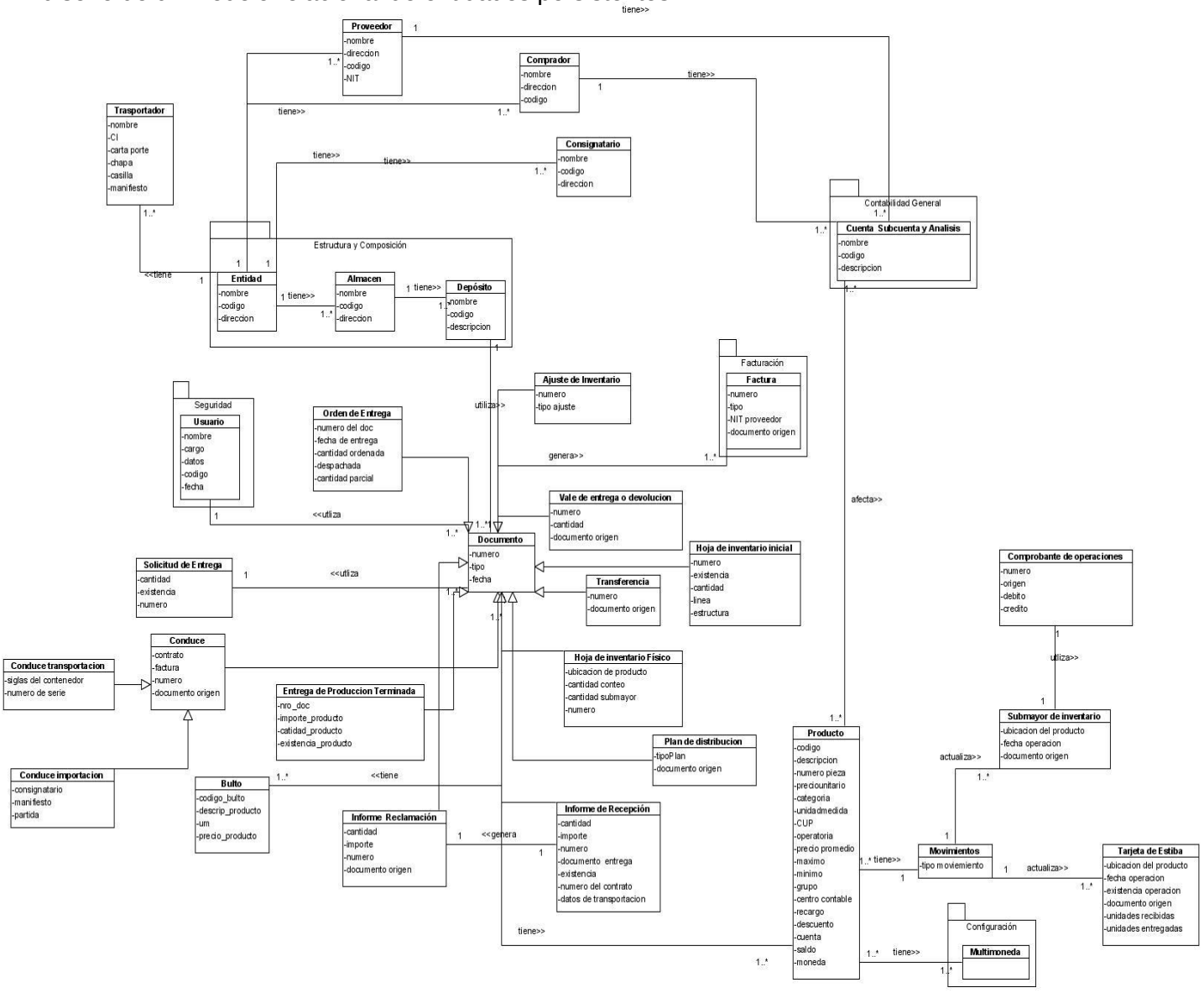



*Figura 3. Modelo Conceptual del Subsistema de Inventario*

## 2.5 Requisitos Funcionales del Subsistema de Facturación dividido por módulos

Para el diseño e implementación de la Base de Datos del Subsistema de Facturación se tomaron en cuenta una serie de requisitos funcionales que a continuación se describen brevemente. Para una mayor especificación ver documentos asociados.

El Subsistema de Facturación está estructurado en cinco componentes y cada uno de ellos tiene requisitos funcionales asociados. A continuación se muestran los requisitos funcionales de los componentes más importantes para el diseño conceptual.

### 2.5.1 Módulo de Factura

- Crear factura.
- Confirmar factura.
- Generar comprobante de operaciones.
- Contabilizar factura.
- Adición de servicios a la factura de servicio.

### 2.5.2 Módulo de Oferta

- Crear oferta.
- Adicionar productos a la oferta de inventario.
- Adicionar servicios a la oferta de servicio.
- Confirmar oferta.
- Convertir la oferta en factura.

### 2.5.3 Módulo de Configuración

- Adicionar precios de venta de los diferentes productos que se facturen. Este precio se gestionará por grupo de entidades.
- Adicionar las tarifas que tendrán los diferentes servicios que se facturen. Estas tarifas se gestionarán por grupo de entidades.
- Mostrar en el modelo de la factura el logotipo de la empresa y cargarlo a la factura.

### 2.5.4 Módulo de Servicio

- Adicionar los diferentes servicios que se van a facturar (los servicios de la empresa en general).

### 2.5.5 Módulo de Nomencladores

- Requisito funcional que va ha gestionar los conceptos que van a estar vigentes en una entidad determinada, los mismo pueden ser: Estatales, Privados, UBPC, etc. Adiciona, Modifica y Elimina conceptos.

- Requisito que gestiona los conceptos por clientes asociados a una entidad determinada. Adiciona y Elimina concepto al cliente.

## 2.6 Modelo Conceptual del Subsistema de Facturación

Después de analizar los requisitos funcionales del Subsistema de Inventario conjuntamente con los analistas de sistema, se elaboró el modelo conceptual como artefacto de gran importancia para iniciar el diseño de un modelo relacional de entidades persistentes.

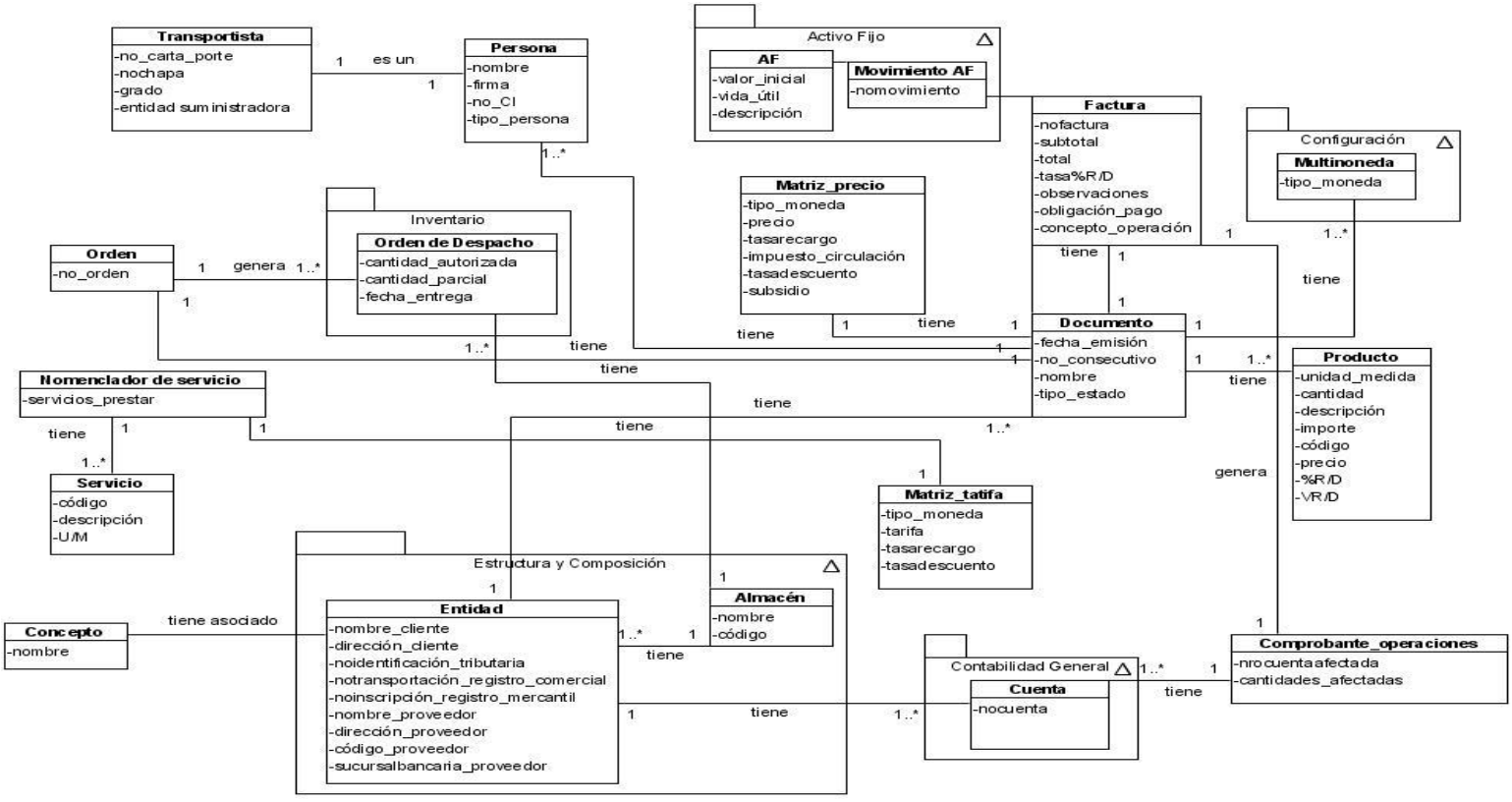



*Figura 4 Modelo Conceptual del Subsistema de Facturación*

## 2.7 Requisitos no Funcionales

**Rendimiento:**

Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

**Soporte:**

La aplicación contará antes de su puesta en marcha con un período de pruebas, se le dará mantenimiento, configuración y se brindará el servicio de instalación.

**Portabilidad:**

El sistema debe ser multiplataforma, haciendo énfasis en Linux y Windows.

**Seguridad:**

Confidencialidad: La información manejada por el sistema esta protegida de acceso no autorizado y divulgación. Mediante la autenticación de los usuarios (Contraseña de acceso).

Integridad: la información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, mediante el mantenimiento de las bases de datos así como la salva de la información se realizará de forma centralizada por el administrador.

Disponibilidad: Significa que los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado de manera que se les dará a los usuarios autorización para tener atribución respecto a sus funciones de trabajo.

### 2.7.1 Requisitos no funcionales para el Servidor de Base de Datos

**Ambiente Windows:**

- Sistema operativo servidor: Microsoft Windows Server 2003.

- Servidor Datos: PostgreSQL 3.0.
- Sistemas para generación de salvas automáticas: Por definir.

**Ambiente Linux:**

- Sistema operativo servidor: Debian 4.0.
- Servidor Datos: PostgreSQL 3.0.
- Sistemas para generación de salvas automáticas: Bacula 2.2.4.

**Requerimientos de Hardware:**

- SERVIDOR al menos P4.
- 2 GB RAM.
- Al menos 3.0 GHz de velocidad para el CPU.
- 2 Discos Duros al menos con 160 GB cada uno.
- Tarjeta RAID
- Switch - 1 L2.
- Cableado UTP.
- UPS para servidor y Router.
- Router, conectividad con el exterior (Al menos 128 Kbs), se solicita que el Router presente opciones de seguridad o se garantice a través de proveedor de red. También para los temas de replicas entre servidores de Base de Datos.

## 2.8 Diseño Arquitectónico de los datos

El diseño arquitectónico de los datos va a contar con dos puntos importantes a seguir:

- **Vista de datos de la arquitectura**

La vista de datos de la arquitectura va a estar sustentada sobre la base de un documento el cual tiene como objetivo la estandarización del trabajo en el diseño y el desarrollo de todas las Base de Datos y los modelos de datos de las líneas de trabajo en las cuales esta dividido el proyecto ERP, generalizando estándares de nomenclatura, tipos de datos, generación de llaves primarias únicas en cada entidad de la base de datos por la importancia que esto representa para la replicación de datos hacia otros servidores, secuencia en estructura de árboles conjuntamente con la normalización de los mismos además de tratarse temas como la concurrencia y el mantenimiento de la base de datos. (UCI, 2008)

- **Estrategia de integración de la solución con otros módulos a nivel de datos**

De acuerdo con las normativas trazadas en el documento “Integración de Datos” elaborado por el equipo de Arquitectura de Datos del proyecto en el mismo plantean la estrategia de integración de la solución con otros módulos en cinco epígrafes.

**Entradas y salidas:** Se representa el flujo de la información entre los módulos donde las entradas representan el flujo de información que entra a un módulo es decir el que se necesita para su correcto funcionamiento y las salidas representan aquella información que contienen en sí que es significativa para el funcionamiento de otro.

**Modelos de integración:** La integración de todos los módulos en cuales esta fraccionado el sistema del ERP cubano, es vital si se quiere tener toda la información en una misma base de datos, pero para ello es necesario integrar cada esquema en los cuales se está desarrollando para que no exista duplicación de datos y mala manipulación de los mismos. (UCI, 2008)

**Matriz de traza de datos:** Se representa la trazabilidad de los datos usando los conceptos de entradas y salidas por líneas de trabajo en las que está dividido el proyecto con una configuración de la matriz que permite el entendimiento de la misma y que refleja la interrelación del trabajo entre las líneas.

**Multi-empresa:** Unos de los temas importantes a tener en cuenta en el desarrollo de un sistema de planificación de recursos empresariales para que se pueda implantar en cualquier entidad dentro de un

propio país, es en cuanto a el tema de la Multi-Empresa, temática que dentro del proyecto se le da tratamiento en uno de los módulos en los que está fraccionado el sistema y se trata del Módulo de Estructura y Composición. (UCI, 2008)

**Integración subsistema entidad:** Este epígrafe muestra como va a ser la interrelación entre los subsistemas y la entidad, para que cada subsistema sepa cual es su código en específico. (UCI, 2008)

## 2.9 Modelo entidad relación general de la Base de Datos

El Modelo entidad relación de la Base de Datos de los Subsistemas de Inventario y Facturación esta diseñado conformando 4 esquemas que completan un total de 82 tablas. (Ver Anexo 1)

## 2.10 Descripción general del modelo entidad relación del Subsistema de Inventario

El Subsistema de Inventario está conformado físicamente en la base de datos con un esquema de Inventario que cuenta con un total de 32 tablas de las cuales 5 son nomencladores y 27 tablas de datos, complementando sus funcionalidades con los esquemas de Documento de 28 tablas, de las cuales 8 son nomencladores, 2 de seguridad y el resto de datos.

### Detalles de las tablas más importantes de los esquemas de Inventario y Documento.

**dat_movimiento**

| Nombre | Valor |
|---|---|
| Descripción | Contiene la relación del producto y el documento conformando así un movimiento. |
| Esquema | mod_inventario |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idmovimiento | decimal(19) | PK | No | Identificador que se genera automáticamente con la secuencia sec_datmovimiento_seq. |

| | | | | |
|---|---|---|---|---|
| idproducto | decimal(19) | FK | Si | Llave foránea que viene de la tabla dat_producto vinculando esta con la llave foránea que viene de dat_documento constituyendo así la tabla dat_movimiento. |
| iddocumento | decimal(19) | FK | No | Llave foránea que viene de la tabla dat_documento vinculando esta con la llave foránea que viene de dat_producto constituyendo así la tabla dat_movimiento. |
| idoperacion | decimal(19) | | Si | Llave foránea producto de la integración con el esquema de datos maestros, la cual especifica la operación que se va a hacer con ese producto. |
| idmonedabase | decimal(19) | | Si | Llave foránea que proviene de datos maestros del nomenclador de monedas. Moneda original con la que entra el producto al almacén. |
| cuenta | varchar(2) | | Si | Número de la cuenta que afecta la compra del producto. |
| recargodescuento | decimal(25) | | Si | El recargodescuento no es más que una acción monetaria que se hace sobre una factura, en este caso con la moneda original. |
| cantidad | decimal(25) | | Si | Cantidad de productos que entraron en el almacén registrados en el documento. |
| ubicacion | decimal(1) | | Si | Atributo que dice si el producto está ubicado o no dentro del almacén, si es 0 es que no está ubicado y si es 1 es que está ubicado. |
| cantidadbulto | decimal(25) | | Si | Si el producto viene en bultos aquí se guarda la cantidad de bultos en los que vino. |
| codigosum | varchar(20) | | Si | Código de la entidad suministradora. |
| version | decimal(6) | | Si | Este campo se usa para el tratamiento de la concurrencia de acceso a los datos. |

| | | | | |
|---|---|---|---|---|
| existenciaactual | decimal(25) | | Si | Es la existencia actual de la cantidad de productos en el almacén. |
| dualidadmonetaria | decimal(1) | | Si | Atributo que dice si el producto presente en el documento entró al almacén con una doble moneda. Además indica el llenado o no de la tabla dat_movdualidadmonetaria. Si está en 0 es que no tiene dualidad monetaria y si está en 1 viceversa. |
| preciocontable | decimal(25) | | Si | Es el precio con que entra el producto al almacén. |
| importecontable | decimal(25) | | Si | Es el importe con que entra el producto al almacén. |
| fechavencimiento | date(0) | | Si | Esta es la fecha de vencimiento del producto. |
| condiferencia | decimal(1) | | Si | Atributo que indica si existen diferencias entre las cantidades de productos. |
| preciopromedio | decimal(25) | | Si | Precio promedio del producto. |
| importetotal | decimal(25) | | Si | Precio por la cantidad de productos. |
| porcientord | decimal(1) | | Si | Atributo que almacena el recargo o el descuento de un producto cuando entra al almacén mediante el proceso de la recepción. |
| cronologia | decimal(19) | | Si | Para saber el orden en que se van contabilizando los documentos. |

**dat_movimientodiferencia**

| Nombre | Valor |
|---|---|
| Descripción | Recoge las diferencias que pueden existir en los diferentes documentos relacionados con los productos donde esto constituye un movimiento y este así se relaciona con la tabla de las diferencias. |
| Esquema | mod_inventario |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idmovdiferencia | decimal(19) | PK | No | Identificador que se genera automáticamente con la secuencia sec_datmovimientodiferencia_seq. |
| idcalidad | decimal(3) | FK | Si | Llave foránea que viene del nomenclador de calidad de movimiento diferencia. |
| idmovimiento | decimal(19) | FK | No | Llave foránea que viene de dat_movimiento, vinculándola con la diferencia existente en los productos si los mismos la tienen. |
| cantidad | decimal(25) | | Si | Cantidad de diferencias. |
| cantidadcalidad | decimal(25) | | Si | Cantidad de calidad deficiente. |
| preciodiferencia | decimal(25) | | Si | Precio de la cantidad de diferencia. |
| importediferencia | decimal(25) | | Si | Importe de la cantidad de diferencia. |
| version | decimal(6) | | Si | Este campo se usa para el tratamiento de la concurrencia de acceso a los datos. |
| motivo | decimal(2) | | Si | Si es sobrante es 0, faltante 1 y sino por defecto -1. |

**dat_producto**

| Nombre | Valor |
|---|---|
| Descripción | Contiene todos los productos de la entidad. |
| Esquema | mod_inventario |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idproducto | decimal(19) | PK | No | Identificador que se genera automáticamente con la secuencia |

| | | | | |
|---|---|---|---|---|
| | | | | sec_datproducto_seq. |
| idprod | decimal(19) | FK | No | Llave foránea que viene del nomenclador de los productos. |
| idumedida | decimal(6) | FK | Si | Llave foránea que viene del grupo de las unidades de medida, relación que muestra la unidad de medida en que se encuentra el producto. |
| idgrupo | decimal(6) | FK | Si | Llave foránea que viene del nomenclador de grupos de productos. |
| idestructuraop | decimal(19) | FK | Si | Identificador que proviene de la tabla de las entidades (nom_filaestruc) del esquema de estructura y composición que permite poner en práctica el concepto de la multientidad. Es el atributo llave que dice según id a que entidad pertenece. |
| numrm | decimal(19) | | Si | |
| preciopromedio | decimal(25) | | Si | Precio promedio que tienen los productos. |
| existencia | decimal(25) | | Si | Cantidad de productos que existen en la entidad. |
| activo | decimal(1) | | Si | Atributo que dice si el producto está en uso en la entidad. |
| cantidaddisponible | decimal(25) | | Si | Cantidad disponible de un producto determinado. |
| version | decimal(6) | | Si | Este campo se usa para el tratamiento de la concurrencia de acceso a los datos. |
| observacion | varchar(255) | | Si | Observaciones que se le hacen al producto. |
| fechacontable | date(0) | | Si | Esta es la fecha en que se adquiere el producto. |
| importetotal | decimal(25) | | Si | Este es el importe total del tipo de producto. |
| idcategoriaproducto | decimal(3) | FK | Si | Llave foránea que viene del nomenclador de las categorías de los productos, da una referencia de la categoría del mismo. |

| | | | | |
|---|---|---|---|---|
| idcuenta | decimal(19) | | Si | Atributo que guarda el valor de la cuenta afectada por la compra del producto. |

**nom_producto**

| Nombre | Valor |
|---|---|
| Descripción | Nomenclador que contiene todos los productos con características generales de los mismos. |
| Esquema | mod_inventario |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idprod | decimal(19) | PK | No | Este campo es el identificador de los productos. |
| idpadre | decimal(19) | FK | Si | Atributo que guarda el identificador del producto del cual es hijo o se deriva; ayuda a formar una jerarquía de forma arbólica. |
| idumedida | decimal(6) | | Si | Identificador que viene del esquema de las Unidades de Medida. Que da la unidad de medida de los productos. |
| idnacionalidad | decimal(19) | FK | Si | Llave foránea que viene de nom_pais, por su relación da la nacionalidad del producto. |
| precio | decimal(25) | | Si | Este es el campo que contiene el precio del producto originalmente. |
| Tipo | decimal(3) | | Si | Este es el campo que determina si el producto es un equipo o no. |
| actual | decimal(1) | | Si | Campo para verificar si está en uso o no el nomenclador. |
| tipocontrol | varchar(255) | | Si | Este es el campo que se usa para el control de los equipos, es decir si es obligatoria o no la especificación del número de serie o no del |

| | | | | |
|---|---|---|---|---|
| | | | | mismo. |
| lft | decimal(9) | | Si | Este es el campo que determina el lado izquierdo de esta tabla ya que la misma tiene una estructura arbólica. Este facilita realizar los recorridos del mismo. |
| rgt | decimal(9) | | Si | Este es el campo que determina el lado derecho de esta tabla ya que la misma tiene una estructura arbólica. Este facilita realizar los recorridos del mismo. |
| codigoprod | varchar(10) | | No | Este campo es el código que tiene un producto. |
| descripcion | varchar(70) | | Si | Este es el campo que describe el nombre del producto. |
| nropieza | varchar(20) | | Si | Este campo es otro atributo identificador del producto. |
| nivel | decimal(1) | | Si | Este es el campo que define el nivel de las ramas del árbol. |
| idestructura | decimal(19) | FK | Si | Identificador que proviene de la tabla de las entidades (nom_filaestruc) del esquema de estructura y composición que permite poner en práctica el concepto de la multientidad. Es el atributo llave que dice según id a que entidad pertenece. |

## 2.11 Descripción general del modelo entidad relación del Subsistema de Facturación

El Subsistema de Facturación está conformado físicamente en la Base de Datos con un esquema de Facturación que cuenta con un total de 17 tablas de las cuales 5 son nomencladores y 12 tablas de datos, complementando sus funcionalidades con los esquemas de Documento formado por 28 tablas de las cuales tributa al esquema de Facturación 1 tabla, y el esquema de Servicio con 5 tablas.

**Detalles de las tablas más importantes de los esquemas de Facturación, Documento y Servicio.**

**dat_matrizprecio**

| Nombre | Valor |
|---|---|
| Descripción | Tabla de datos que almacena el id del producto, el id del concepto por entidad y el id de la entidad a la que corresponde el producto. Tabla que permite la relación de todos ellos y el almacenamiento de los precios de los productos por concepto. |
| Esquema | mod_facturacion |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idproducto | decimal(19) | FK | Si | Llave foránea que viene de dat_producto. Dada la relación del producto con la matriz de precio en el proceso de facturación. |
| idconceptocliente | decimal(19) | FK | Si | Llave foránea que viene del nomenclador del concepto de los clientes, relación dada por la diferencia de precios según conceptos (Batalla de Ideas, CIMEX, etc.). |
| idmatrizprecio | decimal(19) | PK | No | Identificador que se genera automáticamente con la secuencia sec_datmatrizprecio_seq. |
| idestructuracomun | decimal(19) | FK | Si | Identificador que proviene de la tabla de las entidades (nom_filaestruc) del esquema de estructura y composición que permite poner en práctica el concepto de la multientidad. Es el atributo llave que dice según id a que entidad pertenece. |

### dat_matrizpreciodualidadmonetaria

| Nombre | Valor |
|---|---|
| Descripción | Tabla que contiene la estructura requerida para almacenar los datos de un producto asociado a una entidad por concepto donde su valor esta dado en más de una moneda. |
| Esquema | mod_facturacion |

**Atributos**

| Nombre | Tipo de dato | PK/ FK | ¿Es Null ? | Descripción |
|---|---|---|---|---|
| idmoneda | decimal(19) | FK | Si | Llave foránea que viene del nomenclador de monedas en el esquema de datos maestros. |
| idmatrizprecio | decimal(19) | FK | Si | Llave foránea que viene de dat_matrizprecio dado que el precio de la misma pudiera estar en otro tipo de moneda. |
| idmatrizpreciodualidadmonetaria | decimal(19) | PK | No | Identificador que se genera automáticamente con la secuencia sec_datmatrizpreciodualidadmonetaria_seq |
| precioventa | decimal(25) | | Si | Precio de venta de los productos. |

### dat_precio

| Nombre | Valor |
|---|---|
| Descripción | Tabla que guarda los precios definidos para los productos de una entidad. |
| Esquema | mod_facturacion |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idprecio | decimal(19) | PK | No | Identificador que se genera automáticamente con la secuencia sec_datprecio_seq. |
| idprecioentidad | decimal(19) | FK | No | Identifidor de la operación a realizar en la entidad. |
| idmovdualidadmonetaria | decimal(19) | FK | No | Atributo que guarda la dualidad con la que va a trabajar un precio en específico. |
| valoranterior | decimal(25) | | Si | Valor que había anteriormente almacenado en el atributo valor de la tabla en cuestión. |
| valor | decimal(25) | | No | Valor actual del precio del producto. |

**dat_movservicio**

| Nombre | Valor |
|---|---|
| Descripción | Tabla donde se relacionan el documento dat_factura con el nomenclador de los servicios de una entidad. |
| Esquema | mod_servicio |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idoperacion | decimal(19) | | Si | Para generar el comprobante de operaciones. |
| fechavencimiento | date(0) | | Si | Fecha de vencimiento del servicio. |
| idmovservicio | decimal(19) | PK | No | Llave primaria que se genera automáticamente con la secuencia sec_datmovservicio_seq. |
| idserv | decimal(19) | FK | Si | Llave foránea que viene del |

| | | | | |
|---|---|---|---|---|
| | | | | nomenclador de los servicios. |
| idfactura | decimal(19) | | Si | Llave foránea que viene del esquema de Documentos de la tabla dat_factura. |
| cantidad | decimal(25) | | Si | Cantidad de servicios. |
| porcientord | decimal(1) | | Si | Atributo que dice si el servicio tiene porciento de recargo o descuento. |

**nom_categoriaprecio**

| Nombre | Valor |
|---|---|
| Descripción | Nomenclador que contiene los precios con sus descripciones por entidad. |
| Esquema | mod_facturacion |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idcategoriaprecio | decimal(19) | PK | No | Identificador que se genera automáticamente con la secuencia sec_nomcategoriaprecio_seq. |
| descripcion | varchar(255) | | No | Nombre de la categoría del precio. |
| concepto | varchar(1) | | No | |
| actual | decimal(1) | | No | Este campo dice si el nomenclador está en uso o no. |

**nom_servicio**

| Nombre | Valor |
|---|---|
| Descripción | Nomenclador que contiene todos los servicios que presta una entidad. |
| Esquema | mod_servicio |

**Atributos**

| Nombre | Tipo de dato | PK/FK | ¿Es Null? | Descripción |
|---|---|---|---|---|
| idserv | decimal(19) | PK | No | Identificador del nomenclador de servicio. |
| idpadre | decimal(19) | FK | Si | Este es el campo identificador que permite saber el producto base al cual pertenece un producto determinado. |
| descripcion | varchar(255) | | Si | Nombre del servicio. |
| lft | decimal(9) | | Si | Este es el campo que determina el lado izquierdo de esta tabla ya que la misma tiene una estructura arbólica. Este facilita realizar los recorridos del mismo. |
| rgt | decimal(9) | | Si | Este es el campo que determina el lado derecho de esta tabla ya que la misma tiene una estructura arbólica. Este facilita realizar los recorridos del mismo. |
| nivel | decimal(1) | | Si | Este es el campo que define el nivel de las ramas del árbol. |
| codigoservicio | varchar(15) | | Si | Código del servicio. |
| idumedida | decimal(6) | FK | Si | Llave foránea que viene del esquema de las Unidades de Medida. |
| idestructura | decimal(19) | | Si | Identificador que proviene de la tabla de las entidades (nom_filaestruc) del esquema de estructura y composición que permite poner en práctica el concepto de la multientidad. Es el atributo llave que dice según id a que entidad pertenece. |

## 2.12 Argumentación de generación de bases y dominios de la Capa de Acceso a los Datos

La capa de acceso a datos está implementada según las normativas de arquitectura y las propuestas hechas para el uso del Framework de Doctrine que dado su estructura este consta entre otras cosas para su integración y funcionamiento con Zend Studio con dos tipos de ficheros los Bases y los Dominios donde en los dominios se lleva a cabo la implementación de funciones en el lenguaje DQL.

Los ficheros Bases contienen todos los atributos de las tablas correspondientes en la BD y los dominios tienen las relaciones de las tablas, ambos ficheros tienen una relación por herencia. Estos ficheros son producto del llamado Mapeo de la Base de Datos.

En los ficheros de dominio a parte de las relaciones entre las tablas se encuentran implementadas cuatro funciones donde además de estas en algunos por ser importantes en cuanto a la funcionalidad de la capa y del negocio se implementan otras.

### Ejemplo de un fichero Base:

```
<?php
abstract class BaseDatMatrizpreciodualidadmonetaria extends Doctrine_Record
 {
   public function setTableDefinition()
    {
      $this->setTableName('dat_matrizpreciodualidadmonetaria');
      $this->hasColumn('idmatrizpreciodualidadmonetaria', 'numeric', null, array('notnull' => true, 'primary' => true,
'sequence' => 'mod_facturacion.sec_datmatrizpreciodualidadmonetaria_seq'));
      $this->hasColumn('idmatrizprecio', 'numeric', null, array('notnull' => falso, 'primary' => falso));
      $this->hasColumn('idmoneda', 'numeric', null, array('notnull' => falso, 'primary' => falso));
      $this->hasColumn('precioventa', 'numeric', null, array('notnull' => falso, 'primary' => falso));
    }
   public function Setup()
    {
      parent::setUp();
    }
 }
?>
```

**Ejemplo de un fichero Dominio:**

```
<?php
class DatMatriztarfdualmon extends BaseDatMatriztarfdualmon
 {
   public function setUp()
     {   parent::setUp();
       $this->hasMany ('DatTarifa', array ('local'=>'idmatriztarfdualmon','foreign'=>'iddualidadtarifa'));
       $this->hasOne ('DatMatriztarifa', array ('local'=>'idmatriztarifa','foreign'=>'idmatriztarifa'));
     }
    public function GetLLave()
     {
       $query = new Doctrine_Query ();
       $result = $query ->select ('MAX ('idmatriztarfdualmon')') ->from ('DatMatriztarfdualmon')->execute();
       $arr = $result->toArray ();
       return (is_array($arr) ? $arr[0]['MAX'] + 1 : 1);
     }
 public function Buscar ($idmatriztarfdualmon)
     {
       $temp = $this->conn->getTable ('DatMatriztarfdualmon')->find($idmatriztarfdualmon);
        return $temp->toArray();
     }
  public function GetTodos()
   {
       $query = new Doctrine_Query ();
       $result = $query->from('DatMatriztarfdualmon')->execute ();
       return $result->toArray();
   }
  public function GetPorLimite($limite = 10,$inicio = 0)
  {
       $query = new Doctrine_Query ();
       $result = $query->from('DatMatriztarfdualmon')->limit($limite = 10)->offset($inicio = 0)->execute ();
       return $result->toArray();
  }
```

## 2.13 Conclusiones

Se ha representado a través del diseño propuesto de la BD cada uno de los requerimientos que de alguna forma muestran la persistencia de los datos dentro del sistema, donde algunos fueron significativos para la funcionalidad del mismo. Además la implementación de las BD se ha realizado según lo propuesto por Arquitectura.

# Capítulo 3: Validación del Diseño

## 3.1 Introducción

En el capítulo anterior se planteó una propuesta de solución para el diseño e implementación de la Base de Datos de los Subsistemas de Inventario y Facturación del Sistema Cedrux, en el presente capítulo se realiza una validación de la misma. Abordando aspectos como la integridad, seguridad y normalización de la BD, así como el análisis de la redundancia de la información y la trazabilidad de las acciones. Es objetivo de este capítulo validar el diseño realizado de forma teórica y funcional.

## 3.2 Validación teórica del diseño

Para realizar la implementación de una BD se debe tener primeramente un diseño acorde con las funcionalidades que requiere el sistema, pero esto solo no es suficiente; es muy importante tener en cuenta aspectos claves como son la consistencia, integridad y seguridad de los datos almacenados.

### 3.2.1 Restricciones fundamentales para la integridad de los datos

- **Datos Requeridos**

Por ejemplo:

```
CREATE TABLE "mod_inventario"."nom_producto" (“idprod" NUMERIC (19, 0) DEFAULT nextval
('sec_nomproducto_seq'::regclass) NOT NULL,
  "idpadre" NUMERIC(19,0),
  "idumedida" NUMERIC(6,0),
  "idnacionalidad" NUMERIC(9,0),
  "idespecialidad" NUMERIC(9,0),
  "precio" NUMERIC(25,6),
  "tipo" NUMERIC(3,0),
  "actual" NUMERIC(1,0),
  "tipocontrol" VARCHAR(255),
  "lft" NUMERIC(9,0),
  "rgt" NUMERIC(9,0),
  "codigoprod" VARCHAR(10) NOT NULL,
  "descripcion" VARCHAR(70),
  "nropieza" VARCHAR(20),
```

"nivel" NUMERIC(1,0),
"idestructura" NUMERIC(19,0)

- **Chequeo de Validez**

A continuación se muestra un ejemplo donde a la hora de insertar un elemento en la base de datos se lleva a cabo el chequeo de validez, devolviendo un mensaje de error.

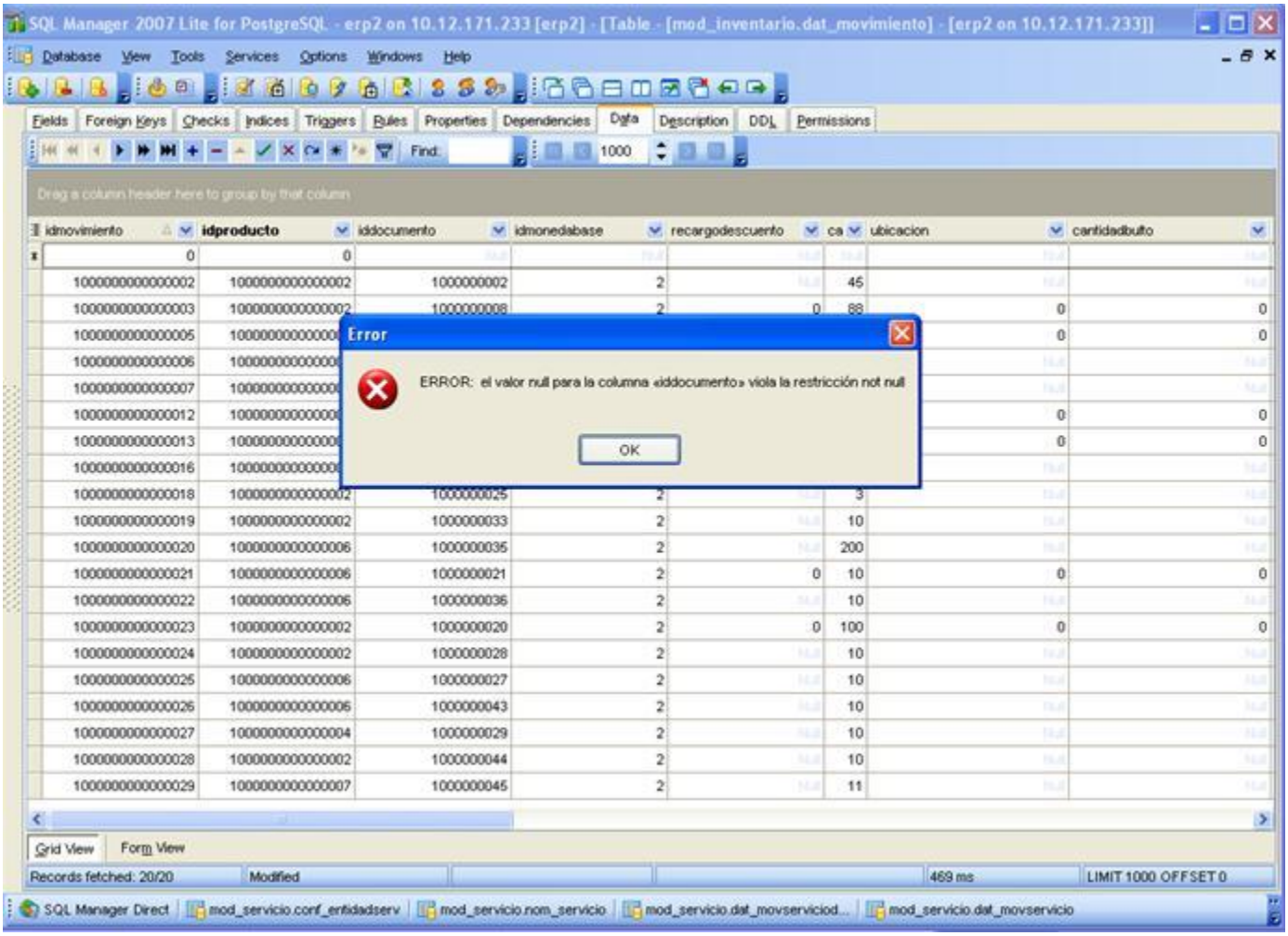



*Figura 5. Ejemplo del chequeo de validez al insertar elementos en la tabla dat_movimiento*

- **Integridad de entidad**

Ejemplo de secuencia para la generación de llave primaria de forma automática:

```
CREATE SEQUENCE "mod_inventario"."sec_datproducto_seq"
INCREMENT 1  MINVALUE 1000000000000000
MAXVALUE 1999999999999999  START 1000000000000012
CACHE 1;
```

- **Integridad Referencial**

Para poner de manifiesto la integridad referencial hay que tener en cuenta que cuando se define una columna como clave foránea, las filas de la tabla pueden contener en esa columna o bien el valor nulo (ningún valor), o bien un valor que existe en la otra tabla, un error sería asignar a un documento un producto que no está en la tabla de productos.

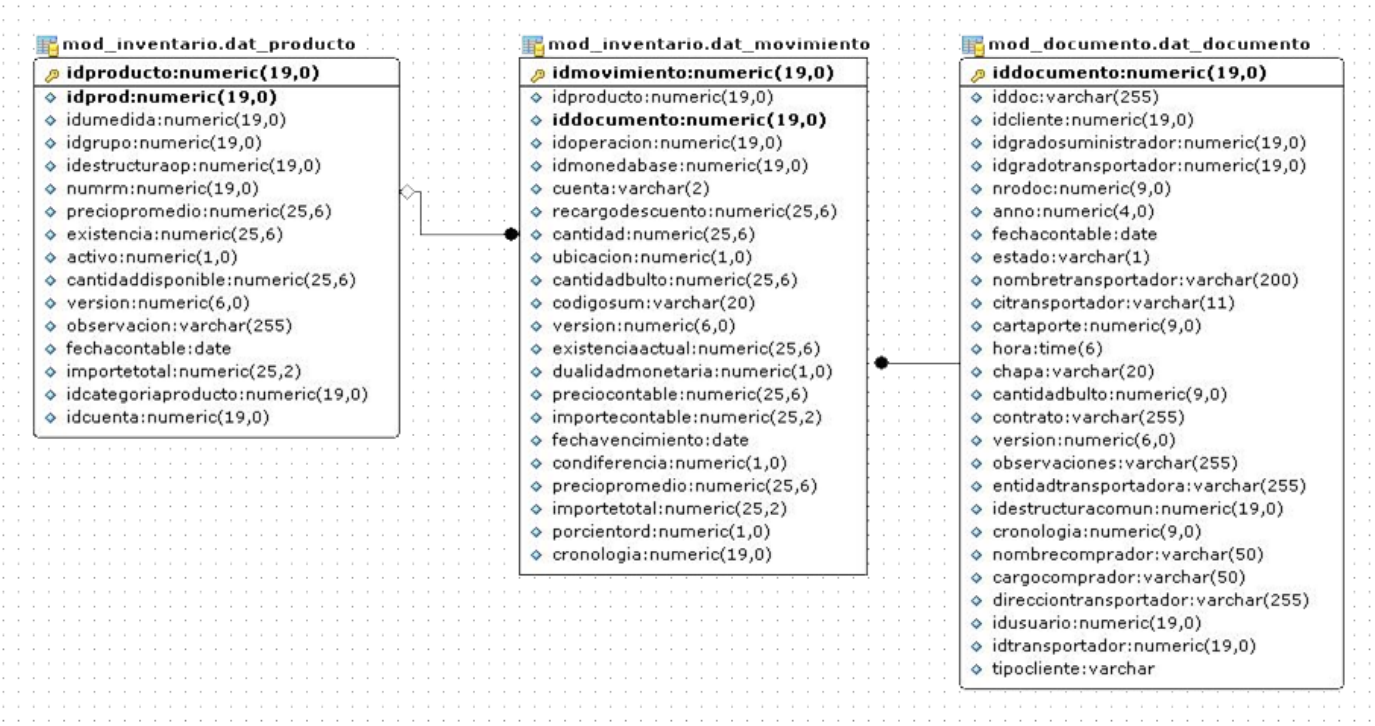



*Figura 6 Integridad referencial entre tablas de esquemas diferentes.*

### 3.2.2 Aplicación de la Normalización y la Desnormalización de la Base de Datos

En el caso específico de la BD de los Subsistemas de Inventario y Facturación inicialmente se tuvo en cuenta la normalización para el diseño de la misma, llegando hasta la tercera forma normal; pero como la base de datos tributa a un sistema empresarial que no está definido para una única entidad o para un solo tipo de moneda, al aplicarle los conceptos de multientidad y multimoneda entre otras se quebranta la normalización, lo que no quita que el diseño sea bueno y eficiente.

**Ejemplos que muestran la desnormalización:**

Existe redundancia en algunos atributos específicos en algunas tablas, como por ejemplo el campo versión que se encuentra en casi todas las tablas de datos para tratar el tema de la concurrencia. El identificador de las entidades que es el idestructura es otro de los campos que redunda.

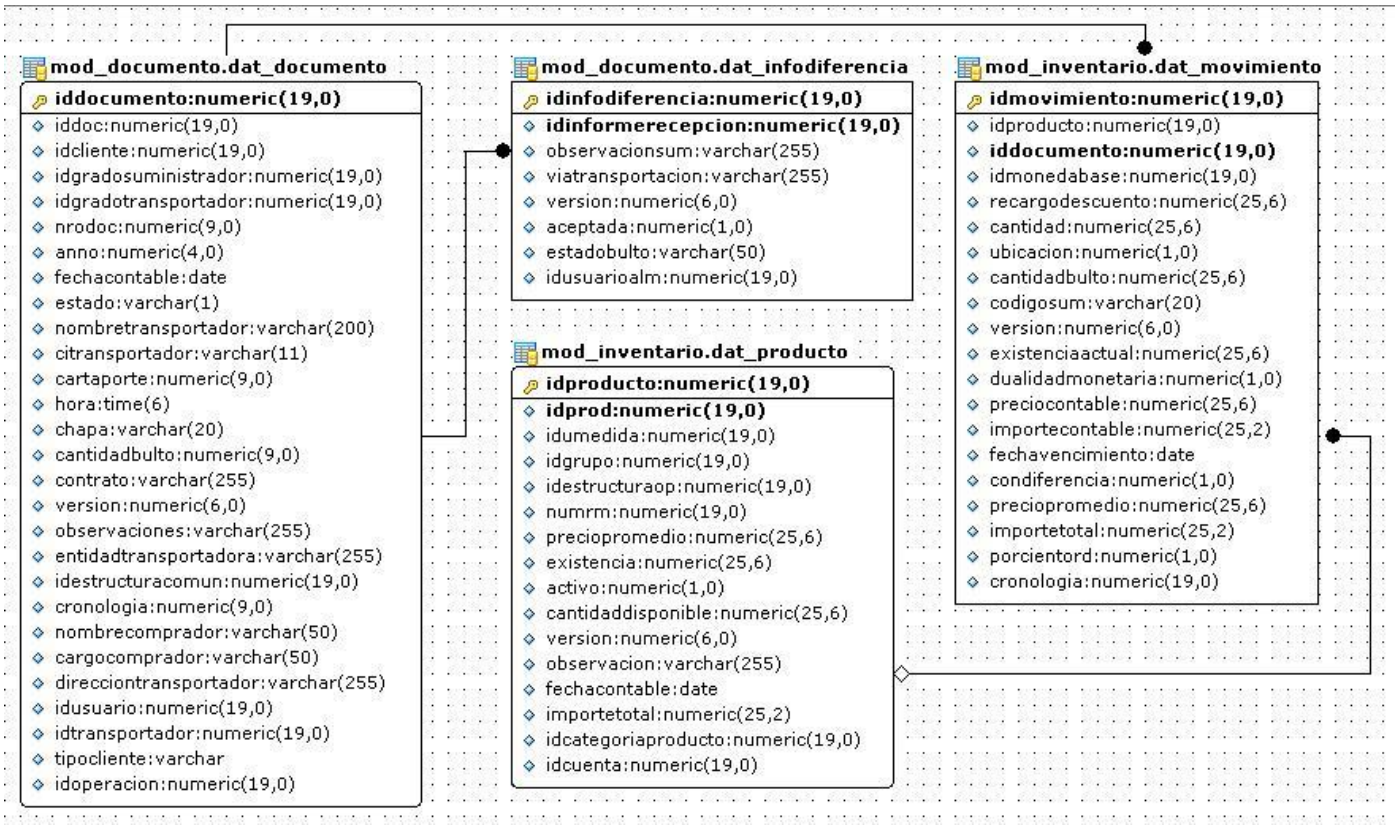



*Figura 7 Ejemplo de desnormalización con el identificador de estructura*

### 3.2.3 Aplicación de la Normalización de árboles en el Subsistema Inventario

Un ejemplo de estas aplicaciones consta en los esquemas de Inventario y Servicio con dos tablas de estructura jerárquica en inventario que son: dat_estructuraubicacion y nom_producto, y una tabla en el esquema de servicio que es: nom_servicio.

**Estructura arbólica del nomenclador de producto:**

```
CREATE TABLE "mod_inventario"."nom_producto" (
"idprod" NUMERIC(19,0) DEFAULT nextval('sec_nomproducto_seq'::regclass) NOT NULL,
"idpadre" NUMERIC(19,0),
"idumedida" NUMERIC(6,0),
"idnacionalidad" NUMERIC(9,0),
"idespecialidad" NUMERIC(9,0),
"precio" NUMERIC(25,6),
"tipo" NUMERIC(3,0),
"actual" NUMERIC(1,0),
"tipocontrol" VARCHAR(255),
"lft" NUMERIC(9,0),
"rgt" NUMERIC(9,0),
"codigoprod" VARCHAR(10) NOT NULL,
"descripcion" VARCHAR(70),
"nropieza" VARCHAR(20),
"nivel" NUMERIC(1,0),
"idestructura" NUMERIC(19,0),
CONSTRAINT "nom_producto_pkey" PRIMARY KEY("idprod"),
CONSTRAINT "fk_nprod_especialidad" FOREIGN KEY ("idespecialidad")
REFERENCES "mod_nomencladores"."nom_especialidad"("idespecialidad")
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE,
CONSTRAINT "fk_nprod_estructura" FOREIGN KEY ("idestructura")
REFERENCES "mod_estructuracomp"."nom_filaestruc"("idfila")
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE,
CONSTRAINT "fk_nprod_nacionalidad" FOREIGN KEY ("idnacionalidad")
REFERENCES "mod_nomencladores"."nom_pais"("idpais")
ON DELETE NO ACTION
ON UPDATE NO ACTION
```

```
NOT DEFERRABLE,
CONSTRAINT "fknom_produc235338" FOREIGN KEY ("idpadre")
REFERENCES "mod_inventario"."nom_producto"("idprod")
ON DELETE CASCADE
ON UPDATE NO ACTION
DEFERRABLE
INITIALLY DEFERRED
) WITH OIDS
```

### 3.3 Seguridad de la Base de Datos

La autenticación de usuarios, el control de acceso y los privilegios del mismo fueron puestos en práctica por los Arquitectos de Datos principales del proyecto, aumentando así la seguridad de la BD y la protección de la integridad de la información. Algunas de las medidas tomadas fueron:

- Únicamente el dueño (o el súper-usuario Postgres) puede hacer cualquier cosa con los objetos de esa base de datos.

- Se otorgaron privilegios de acceso para otros usuarios que solo interactuarían con los objetos correspondientes.

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

- Se crearon diferentes usuarios en la base de datos para cada aspecto de la aplicación, con privilegios muy limitados sobre los objetos de la base de datos. Estos usuarios se utilizarán en el archivo de configuración para la conexión a la BD. Con esto se garantiza que si un intruso gana acceso a la base de datos usando una de éstas credenciales, él solo puede acarrear el menor daño posible a la misma.

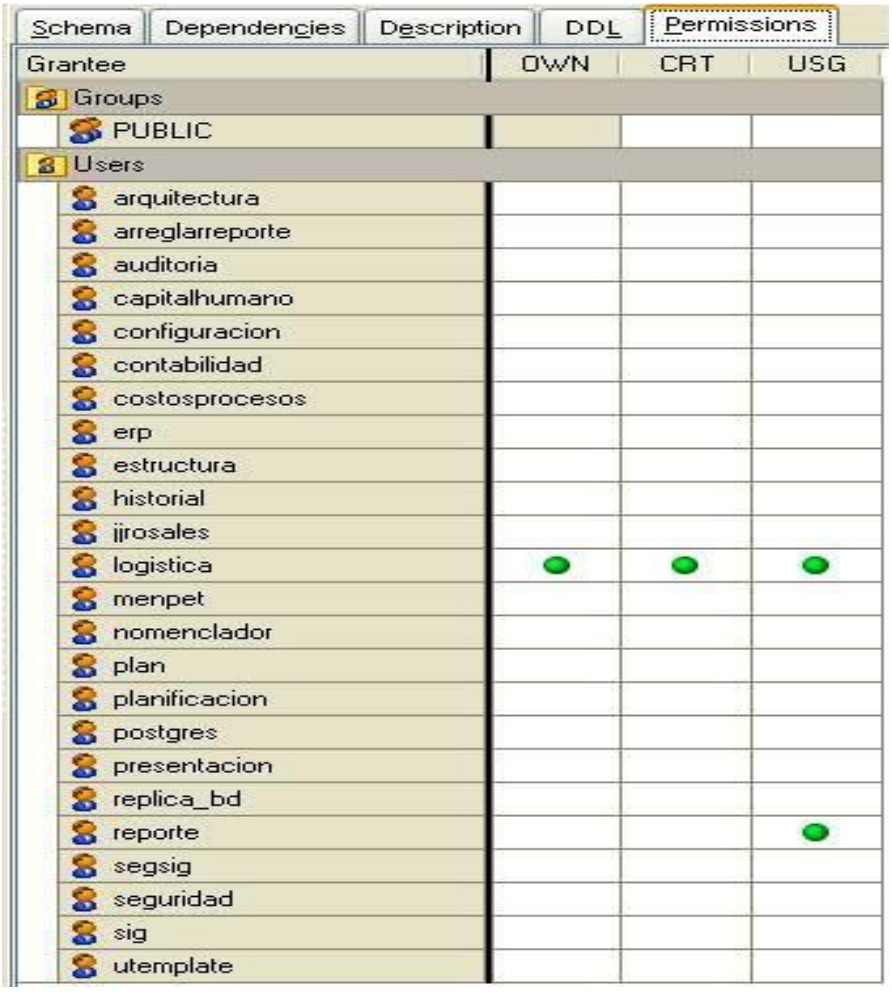



*Figura 8 Tabla de usuarios con sus privilegios de acceso*

### 3.4 Validación funcional del diseño

Para la validación funcional del diseño se realizaron una serie de pruebas enfocadas al rendimiento de algunas consultas hechas a la BD que manejaban gran volumen de información, donde para lograr hacer estas pruebas primeramente se hizo una gran carga de datos inicial a las tablas implicadas, usando la herramienta EMS Data Generator para PostgreSQL 2005 la cual tiene una buena integración con postgreSQL y permite generar hasta 10 000 valores por columna, genera datos para una o varias tablas a la vez, definiendo para cada campo el rango de valores admisibles, dependiendo del tipo, además de la cantidad de tuplas que se desean generar, validando de forma automática la integridad referencial. Un

gran volumen de información que se necesitaba almacenar en la BD estaba consolidada en ficheros .csv y para cargar los mismo se hizo necesario utilizar la funcionalidad para importación de datos que tiene el EMS PostgreSQL 2005 por defecto, lo que permite disminuir el tiempo de llenado de las tablas y evitar equivocaciones en el momento de insertar los datos existentes en la aplicación anterior.

**Las pruebas de validación normalmente se centran en las operaciones siguientes:**

- La carga de la Base de Datos se realiza sin violar la integridad de los datos.
- La correcta interfaz de las aplicaciones con la Base de Datos.
- El rendimiento del sistema satisface las necesidades para las que se adquirió el SGBD.

Para realizar las pruebas se ejecutaron unas de las consultas de mayor envergadura, las cuales tomarían en relación al volumen de información mayores tiempos de respuesta:

La primera de las pruebas que se hizo fue con la función del submayor de inventario, que se usa para la recuperación de todos los datos de un submayor pasando como parámetro el identificador del producto.

```
CREATE OR REPLACE FUNCTION "mod_inventario"."f_submayor_inventario" (idproducto numeric) RETURNS SETOF
"mod_inventario"."td_submayor_inv" AS
$body$
declare
all_rec mod_inventario.td_submayor_inv;
begin
    for all_rec in
    (SELECT DISTINCT
"mod_estructuracomp"."dat_estructura"."denominacion",
"mod_estructuracomp"."dat_estructura"."codigo",
"mod_estructuracomp"."dat_estructuraop"."denominacion",
"mod_inventario"."nom_producto"."codigoprod",
"mod_inventario"."nom_producto"."nropieza",
"mod_umedida"."nom_unidadsi"."abreviatura",
"mod_inventario"."nom_categoriaproducto"."descripcion",
"mod_inventario"."nom_grupoproducto"."descripcion",
ddd."fechacontable",
"mod_documento"."nom_documento"."descripcion",
ddd."nrodoc",
"mod_nomencladores"."nom_clientesproveedores"."nombre",
```

```
case when ("mod_documento"."nom_documento"."concepto" ='+' and ddd.estado <> '7') or ddd.operacion='+' or
("mod_documento"."nom_documento"."concepto" ='-' and ddd.estado = '7') then "dat_movimiento"."cantidad"
end as cantidad,
case when ("mod_documento"."nom_documento"."concepto" ='+' and ddd.estado <> '7') or ddd.operacion='+' or
("mod_documento"."nom_documento"."concepto" ='-' and ddd.estado = '7') then
"dat_movimiento"."importecontable" end as importe,
case when "mod_documento"."nom_documento"."concepto" ='-' or ddd.operacion='-' or
("mod_documento"."nom_documento"."concepto" ='+' and ddd.estado = '7') then "dat_movimiento"."cantidad"
end as cantidad,
case when "mod_documento"."nom_documento"."concepto" ='-' or ddd.operacion='-' or
("mod_documento"."nom_documento"."concepto" ='+' and ddd.estado = '7') then
"dat_movimiento"."importecontable" end as importe,
"mod_inventario"."dat_movimiento"."existenciaactual",
"mod_inventario"."dat_movimiento"."importetotal",
"mod_inventario"."dat_movimiento"."preciopromedio",
ddd."cronologia",
ddd."iddocumento",
ddd.iddoc,
"mod_inventario"."nom_producto"."descripcion",
"mod_inventario"."dat_producto"."fechacontable",
"mod_inventario"."dat_producto".numrm,
"mod_inventario"."dat_movimiento"."preciocontable",
"mod_inventario"."dat_movimiento"."recargodescuento",
ddd."estado"
FROM
"mod_estructuracomp"."dat_estructura"
Inner Join "mod_estructuracomp"."dat_estructuraop" ON "mod_estructuracomp"."dat_estructura"."idestructura" =
"mod_estructuracomp"."dat_estructuraop"."idestructura"
Inner Join
   ("mod_documento"."dat_documento" d
      left join ("mod_documento"."dat_ajuste" da
            inner join "mod_documento"."nom_ajuste" n on da.idajust=n.idajust) a
      on d.iddocumento=a.idajuste) as ddd
 ON "mod_estructuracomp"."dat_estructuraop"."idestructuraop" = ddd."idestructuracomun"
Inner Join "mod_inventario"."dat_producto" ON "mod_estructuracomp"."dat_estructuraop"."idestructuraop" =
"mod_inventario"."dat_producto"."idestructuraop"
Left Join "mod_umedida"."nom_unidadmedcomun" ON
"mod_umedida"."nom_unidadmedcomun"."idumedidadcomun" = "mod_inventario"."dat_producto"."idumedida"
Left Join "mod_umedida"."nom_unidadsi" ON "mod_umedida"."nom_unidadsi"."idunidadsi" =
"mod_umedida"."nom_unidadmedcomun"."idumedidadcomun"
Left Join "mod_inventario"."nom_producto" ON "mod_inventario"."nom_producto"."idprod" =
"mod_inventario"."dat_producto"."idprod"
```

```
Inner Join "mod_inventario"."dat_movimiento" ON ddd."iddocumento" =
"mod_inventario"."dat_movimiento"."iddocumento" AND "mod_inventario"."dat_producto"."idproducto" =
"mod_inventario"."dat_movimiento"."idproducto"
Left Join "mod_documento"."nom_documento" ON "mod_documento"."nom_documento"."iddoc" = ddd."iddoc"
Left Join "mod_nomencladores"."nom_clientesproveedores" ON
"mod_nomencladores"."nom_clientesproveedores"."idclientesproveedores" = ddd."idcliente"
Left Join "mod_inventario"."nom_categoriaproducto" ON
"mod_inventario"."nom_categoriaproducto"."idcategoriaproducto" =
"mod_inventario"."dat_producto"."idcategoriaproducto"
Left Join "mod_inventario"."nom_grupoproducto" ON "mod_inventario"."nom_grupoproducto"."idgrupo" =
"mod_inventario"."dat_producto"."idgrupo"
WHERE
"mod_inventario"."dat_producto"."idproducto" = $1 AND
"mod_documento"."nom_documento"."concepto" in ('+', '-', 'N') AND
(ddd."cronologia" <> 0 and ddd.estado in ('3','5','7','9'))
ORDER BY ddd."cronologia" ASC
    )
    loop
    return next all_rec;
    end loop;
end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100 ROWS 1000;
```

El resultado obtenido a partir de esta consulta fue en un tiempo de 0,09 segundos.

La otra función con la que se hicieron pruebas es la que se usa para la recuperación de todos los datos de un documento de ajuste, pasando como parámetro el identificador del documento y el identificador de la entidad.

```
CREATE OR REPLACE FUNCTION "mod_documento"."f_ajuste" (iddocumento numeric, idestructuracomun numeric)
RETURNS SETOF "mod_documento"."td_ajuste" AS
$body$
declare
all_rec mod_documento.td_ajuste;
begin
```

```
    for all_rec in
    (SELECT DISTINCT
"mod_estructuracomp"."dat_estructura"."denominacion" AS "Unidad",
"mod_estructuracomp"."dat_estructuraop"."denominacion" AS "Almacen",
"mod_documento"."dat_documento"."nrodoc" AS "No. Ajuste",
"mod_documento"."dat_documento"."fechacontable" AS "Fecha Ajuste",
"mod_documento"."nom_ajuste"."descripcion" AS "Motivo",
"mod_inventario"."nom_producto"."codigoprod" AS "Codigo",
"mod_inventario"."nom_producto"."descripcion" AS "Descripcion",
"mod_umedida"."nom_unidadsi"."abreviatura" AS "U/M",
"mod_inventario"."dat_movimiento"."cantidad" AS "Cantidad",
"mod_inventario"."dat_movimiento"."preciocontable" AS "Precio",
"mod_inventario"."dat_movimiento"."importecontable" AS "Importe",
"mod_inventario"."dat_producto"."existencia" AS "Saldo existencia",
"mod_nomencladores"."nom_especialidad"."denespecialidad" AS "Especialidad",
"mod_inventario"."nom_categoriaproducto".descripcion,
"mod_documento"."nom_ajuste"."idajust" AS "idmotivo",
"mod_documento"."dat_documento"."observaciones",
"mod_estructuracomp"."dat_estructura"."codigo" as "codigoest"
FROM
"mod_estructuracomp"."dat_estructura"
Inner Join "mod_estructuracomp"."dat_estructuraop" ON "mod_estructuracomp"."dat_estructura"."idestructura" =
"mod_estructuracomp"."dat_estructuraop"."idestructura"
Inner Join "mod_documento"."dat_documento" ON "mod_estructuracomp"."dat_estructuraop"."idestructuraop" =
"mod_documento"."dat_documento"."idestructuracomun"
Inner Join "mod_inventario"."dat_producto" ON "mod_estructuracomp"."dat_estructuraop"."idestructuraop" =
"mod_inventario"."dat_producto"."idestructuraop"
Inner Join "mod_umedida"."nom_unidadmedcomun" ON
"mod_umedida"."nom_unidadmedcomun"."idumedidadcomun" = "mod_inventario"."dat_producto"."idumedida"
Inner Join "mod_umedida"."nom_unidadsi" ON "mod_umedida"."nom_unidadsi"."idunidadsi" =
"mod_umedida"."nom_unidadmedcomun"."idumedidadcomun"
Inner Join "mod_inventario"."nom_producto" ON "mod_inventario"."nom_producto"."idprod" =
"mod_inventario"."dat_producto"."idprod"
Inner Join "mod_inventario"."dat_movimiento" ON "mod_documento"."dat_documento"."iddocumento" =
"mod_inventario"."dat_movimiento"."iddocumento" AND "mod_inventario"."dat_producto"."idproducto" =
"mod_inventario"."dat_movimiento"."idproducto"
Inner Join "mod_documento"."dat_ajuste" ON "mod_documento"."dat_documento"."iddocumento" =
"mod_documento"."dat_ajuste"."idajuste"
Inner Join "mod_documento"."nom_ajuste" ON "mod_documento"."nom_ajuste"."idajust" =
"mod_documento"."dat_ajuste"."idajust"
```

```
Inner Join "mod_nomencladores"."nom_especialidad" ON
"mod_nomencladores"."nom_especialidad"."idespecialidad" =
"mod_estructuracomp"."dat_estructura"."idespecialidad"
Inner join mod_inventario.nom_categoriaproducto on "mod_inventario"."dat_producto".idcategoriaproducto =
mod_inventario.nom_categoriaproducto.idcategoriaproducto
WHERE
"mod_documento"."dat_documento"."iddocumento" = $1 AND
"mod_documento"."dat_documento"."idestructuracomun" = $2
    )
    loop
    return next all_rec;
    end loop;
end;
$body$
LANGUAGE 'plpgsql'
VOLATILE
CALLED ON NULL INPUT
SECURITY INVOKER
COST 100 ROWS 1000;
```

El resultado obtenido a partir de esta consulta fue en un tiempo de 0.12 segundos.

Las pruebas realizadas a una BD nunca pueden tomarse como resultados reales, aunque dan un aproximado al mismo, dependiendo del grado de acercamiento que se utilice a los procesos de la vida real. Aún así, la implantación y utilización de una BD, está marcada por factores como, cantidad de usuarios a conectarse y el grado de concurrencia de las solicitudes hechas por los mismos, los cuales hay que tener en cuenta siempre. Además, es necesario velar el cumplimiento de todos los requerimientos no funcionales propuestos para la BD, ya que validan el funcionamiento correcto y máximo rendimiento de la misma. Aunque las pruebas realizadas brinden resultados que puedan parecer alentadores, no pueden ser motivo de confianza, todo lo contrario, la mejor prueba que se le puede realizar a cualquier sistema informático lo constituye la interacción directa del usuario, de aquí los procesos de mantenimiento de software y la realización de versiones del mismo buscando mejorar o resolver posibles errores que se detecten durante su vida útil.

## 3.5 Valoración de resultados

De los resultados de las pruebas realizadas a la BD podemos afirmar que el diseño e implementación de la misma cumple con las perspectivas esperadas, para esto se tuvieron en cuenta factores importantes como el tiempo de realización de las operaciones, confidencialidad, seguridad, disponibilidad e integridad de los datos almacenados, además de parámetros como los requisitos capturados por los analistas del sistema. Se verificó que las relaciones entre las tablas fueron construidas adecuadamente, y se disminuyó en gran medida la redundancia de la información.

## 3.6 Conclusiones

La certificación del diseño se ha podido culminar debido a la puesta en acción de validaciones tanto teóricas como funcionales, incluyendo dentro de la validación teórica aspectos muy importantes como la integridad de la información almacenada, la seguridad, la normalización y desnormalización de la Base de Datos; la misma se validó poniendo en práctica las experiencias adquiridas como diseñadores de BD, sobre la base de los requerimientos del negocio y conceptos como los de multientidad y multimoneda. La validación funcional estuvo respaldada por dos funciones implementadas en el gestor de base de datos, las cuales pusieron a prueba la optimización y rendimiento del diseño y la capacidad de repuesta de este ante la petición concurrente de datos al sistema, permitiendo todo esto hacer finalmente una valoración del diseño y llegar a conclusiones satisfactorias.

# Conclusiones Generales

Los Sistemas Gestores de Bases de Datos son cada día más usados en el cúmulo de aplicaciones que se desarrollan, por lo que la implementación de los diseños en los mismos tienen que ser mejores y más eficaces. Es por esto que se caracterizó Inventario y Facturación como proceso y sistemas, se evaluaron sus experiencias nacionales e internacionales, se mantuvo un contacto directo con funcionales de la actividad rectora de la logística en Cuba, además de personal calificado que trabajan directamente en almacenes llevando a cabo cada uno de estos procesos. Se tuvieron en cuenta los requisitos funcionales y no funcionales que fueron importantes para el diseño. Se definieron todas las entidades que van a persistir conjuntamente con las relaciones entre estas, siguiendo metodologías de diseño, cumpliendo con estándares definidos por la directiva de Arquitectura del proyecto y la experiencia de los diseñadores.

Todos estos aspectos que se tuvieron en cuenta para el diseño permitieron la obtención del modelo entidad relación que se implementó en el gestor PostgreSQL, generando de esta forma físicamente la Base de Datos de los Subsistemas de Inventario y Facturación, que según pruebas realizadas a la misma y al diseño se evidencia que cumple con los requisitos de los clientes y que la misma está apta para su funcionamiento, de acuerdo a su grado de integridad y seguridad.

# Recomendaciones

- Ampliar el diseño e implementación de la BD para soportar la información generada por las nuevas funcionalidades.
- Proporcionar un mantenimiento y soporte regular a la BD enfocados a su mejor funcionamiento.

# Referencias Bibliográficas

**Alomá Santos, Karen, et al. 2009.** *Especificación de requisitos de software Gestionar números de serie de los equipos.* Ciudad de la Habana : s.n., 2009.

**Álvarez Márquez, Yaima, Alomá Santos, Karen and Borges López, Leosdanis. 2009.** *Especificación de requisitos de software Gestionar informe de recepción.* Ciudad de la Habana : s.n., 2009.

Arquitectura de los SGBD. [Online] [Cited: Febrero 19, 2009.] http://ggomez.files.wordpress.com/2008/09/sesion3.pdf.

**Batini, C, Ceri, S and Navathe, S.B. 1994.** *Diseño Conceptual de Bases de Datos. Un enfoque de entidades-interrelaciones.* s.l. : Addison-Wesley/Diaz de Santos, 1994.

**Borges López, Leosdanis. 2009.** *Especificación de requisitos de software Gestionar categorías del producto Inventario.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de requisitos de software Gestionar descripción de calidad del producto Inventario.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de Requisitos de Software Gestionar estructura de ubicación.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de Requisitos de Software Gestionar formato de estructura de ubicación.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de requisitos de software Gestionar grupos del producto.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de requisitos de software Gestionar nomenclador de productos.* Ciudad de la Habana : s.n., 2009.

**Borges López, Leosdanis, et al. 2009.** *Especificación de requisitos de software Gestionar informe de diferencias.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de requisitos de software Gestionar lotes del producto.* Ciudad de la Habana : s.n., 2009.

**Borges López, Leosdanis, Pupo Leyva, Iliannis and Alomá Santos, Karen. 2008.** *Especificación de requisitos de software Ubicar productos en las áreas de almacenamiento.* Ciudad de la Habana : s.n., 2008.

Build Quality Applications Faster, Better and Cheaper. [Online] [Cited: Febrero 3, 2009.] http://www.visual-paradigm.com/product/vpuml/.

**C.J, Date. 2003.** Introducción a los Sistemas de Base de Datos. Ciudad de la Habana : Félix Varela, 2003. Vol. I.

**Carbonell Vargas, Celia Yadira. 2009.** *Especificación de requisitos de software Gestionar documento de baja.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de requisitos de software Gestionar documentos de ajuste de inventario.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de requisitos de software Gestionar números de serie de los equipos.* Ciudad de la Habana : s.n., 2009.

Clasificación de los sistemas de gestión de bases de datos. [Online] [Cited: Marzo 20, 2009.] http://www3.uji.es/~mmarques/f47/apun/node38.html.

Conceptos básicos de integridad referencial. [Online] [Cited: Abril 5, 2009.] http://www.aulaclic.es/sql/b_8_1_1.htm.

**Correa, Ramírez. 2004.** *Rol y Contribución de los Sistemas de Planificación de los Recursos de la Empresa ERP.* Universidad de Sevilla : s.n., 2004.

**Cuba, Arquitectos de datos. ERP. 2008.** *Vista de datos de la Arquitectura de ERP.* Ciudad de La Habana : s.n., 2008.

Doctrine. *PHP Object Relational Mapper.* [Online] [Cited: Febrero 1, 2009.] http://www.doctrine-project.org/documentation/manual/1_1/en/introduction.

El Rincón del Vago. *Lógica Proposicional. Algebra Relacional. Cálculo Relacional. Dependencias Funcionales. Esquema Lógico. SQL (Structured Query Language). Atributos.* [Online] [Cited: Marzo 20, 2009.] http://html.rincondelvago.com/bases-de-datos_5.html.

**Figueroa, José Luis. 1998.** [Online] 1998. [Cited: Enero 10, 2009.] http://www.gestiopolis.com.

**González Brito, Henry Raúl. 2006.** *ERP cubano, un paso estratégico para la consolidación del Software Libre en Cuba.* Ciudad de la Habana : s.n., Octubre 2006.

**Hernández León, Rolando Alfredo and Coello González, Sayda. 2002.** *El paradigma cuantitativo de la investigación científica.* s.l. : Editorial Universitaria (EDUNIV), 2002.

**Lichilín Ríos, Yoandy. 2008.** *Documento de normalización de árboles.* Ciudad de la Habana : s.n., 2008.

Metodología de diseño de bases de datos . [Online] [Cited: Marzo 13, 2009.] http://www3.uji.es/~mmarques/f47/apun/node81.html.

Monografias.com. [Online] [Cited: Abril 2, 2009.] http://www.monografias.com/trabajos5/norbad/norbad.shtml.

**Morales Ortiz, Aylín and Carbonell Vargas, Celia Yadira. 2009.** *Especificación de requisitos de software Gestionar hoja de inventario físico.* Ciudad de la Habana : s.n., 2009.

**—. 2009.** *Especificación de requisitos de software Gestionar productos a la hoja de inventario.* Ciudad de la Habana : s.n., 2009.

**Morales Ortiz, Aylín and Carbonell Vargas., Celia Yadira. 2009.** *Especificación de requisitos de software Gestionar Plan de conteo .* Ciudad de la Habana : s.n., 2009.

Normalización de Bases de Datos y Técnicas de diseño. [Online] [Cited: Abril 5, 2009.] http://www.abcdatos.com/webmasters/tutorial/l1873.html.

**Oliva Martínez, Saily and Pérez Pérez, Elisabeth. 2009.** *Especificación de Requisitos de Software Gestionar autorización de entrega.* Ciudad de la Habana : s.n., 2009.

**Oliva Martínez, Saily. 2009.** *Especificación de Requisitos de Software Desbloqueo de documentos.* Ciudad de la Habana : s.n., 2009.

**—. 2008.** *Especificación de Requisitos de Software Generar conduce.* Ciudad de la Habana : s.n., 2008.

**—. 2008.** *Especificación de Requisitos de Software Generar transferencia.* Ciudad de la Habana : s.n., 2008.

**—. 2009.** *Especificación de Requisitos de Software Gestionar documentos de conciliación.* Ciudad de la Habana : s.n., 2009.

**Oliva Martínez, Saily, Pupo Leyva, Iliannis and Alomá Santos, Karen. 2008.** *Especificación de requisitos de software Gestionar productos al informe de diferencias.* Ciudad de la Habana : s.n., 2008.

**Pupo Leyva, Iliannis, et al. 2009.** *Especificación de requisitos de software Gestionar productos al informe de recepción.* Ciudad de la Habana : s.n., 2009.

**Rosario, Javier y Gil, F.A.** Departamento de Computación de la Facultad Experimental de Ciencias y Tecnología (FaCyT) de la Universidad de Carabobo. [En línea] http://alfa.facyt.uc.edu.ve/computacion/pensum/cs0347/download/exposiciones/1/SGDB.pdf.

SAP - Wikipedia, La enciclopedia libre. [Online] [Cited: Febrero 10, 2009.] http://es.wikipedia.org/wiki/SAP.

SQL Manager.Net. *EMS Database Management Solutions.* [Online] [Cited: Enero 15, 2009.] http://www.sqlmanager.net/products/postgresql/manager.

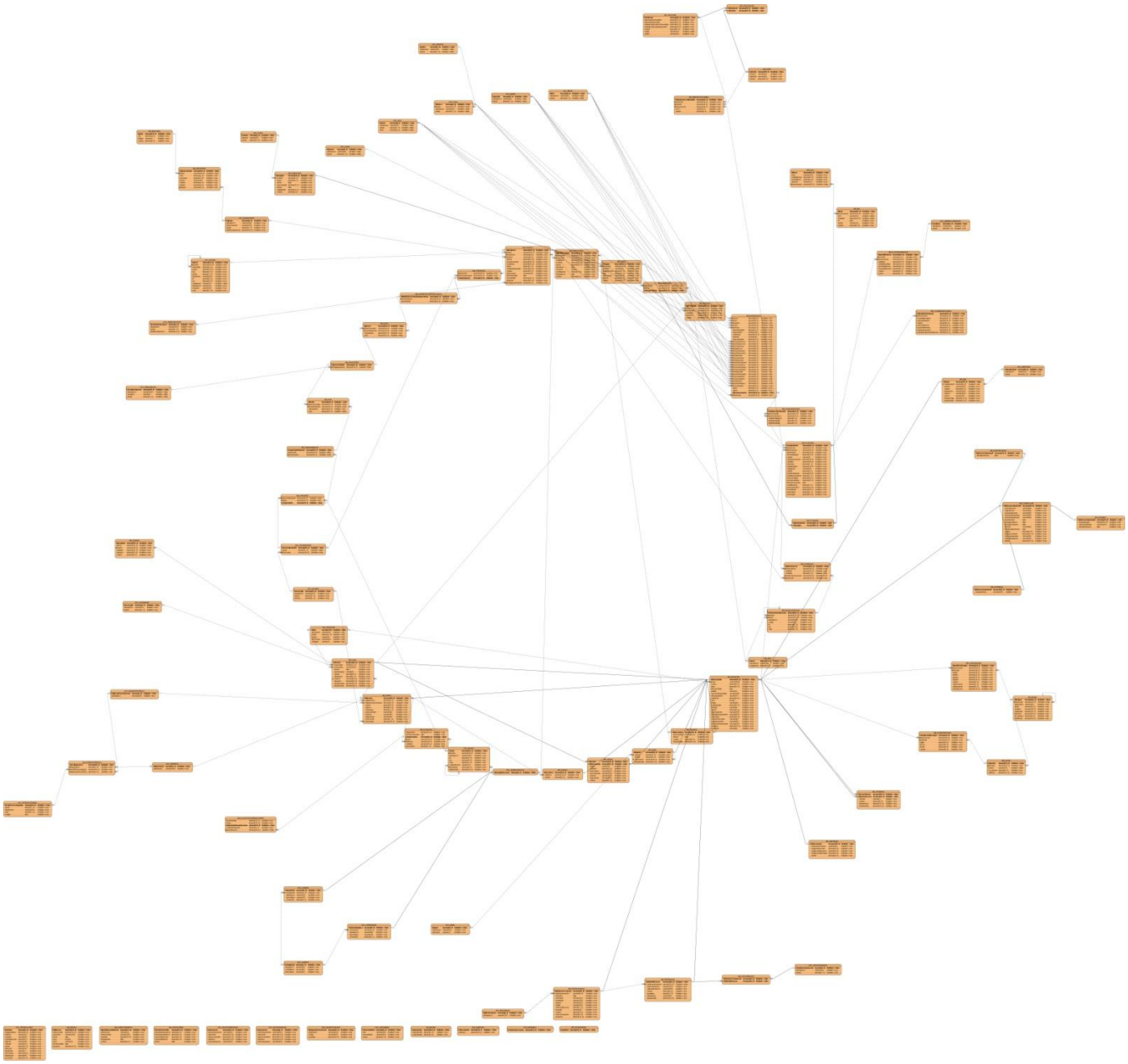
TechWEEk.es. *SAP tiene a Oracle en su punto de mira .* [Online] [Cited: Marzo 15, 2009.] http://www.techweek.es/erp/analisis/1000689002801/sap-tiene-oracle-punto-mira.1.html.

**UCI, Arquitectos de datos. 2008.** *Vista de datos de la Arquitectura de ERP.* Ciudad de la Habana : s.n., 2008.

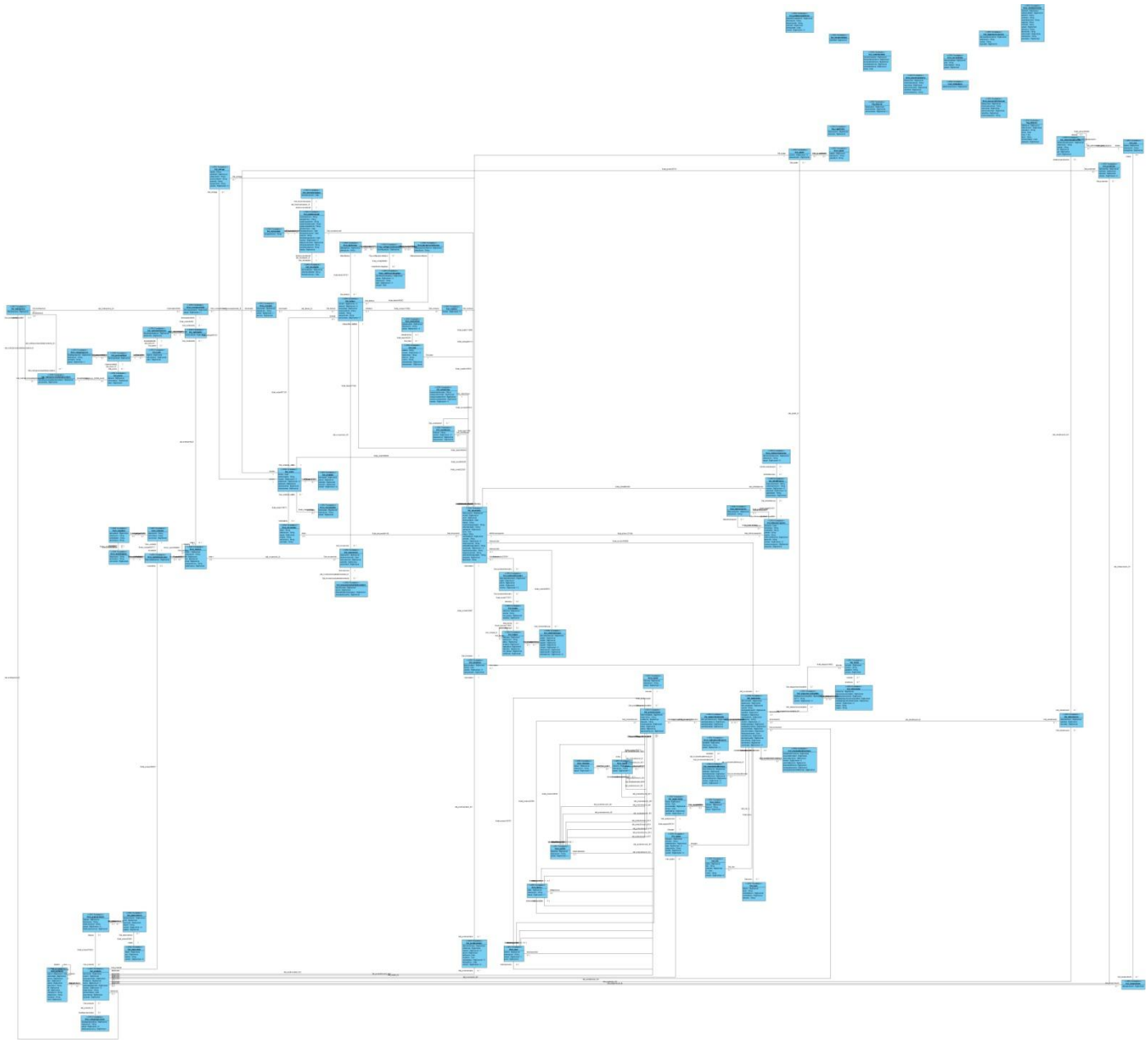
**UCI, Línea de Arquitectura. 2008.** *Documento de Integración de Datos.* Ciudad de la Habana : s.n., 2008.

# Anexos

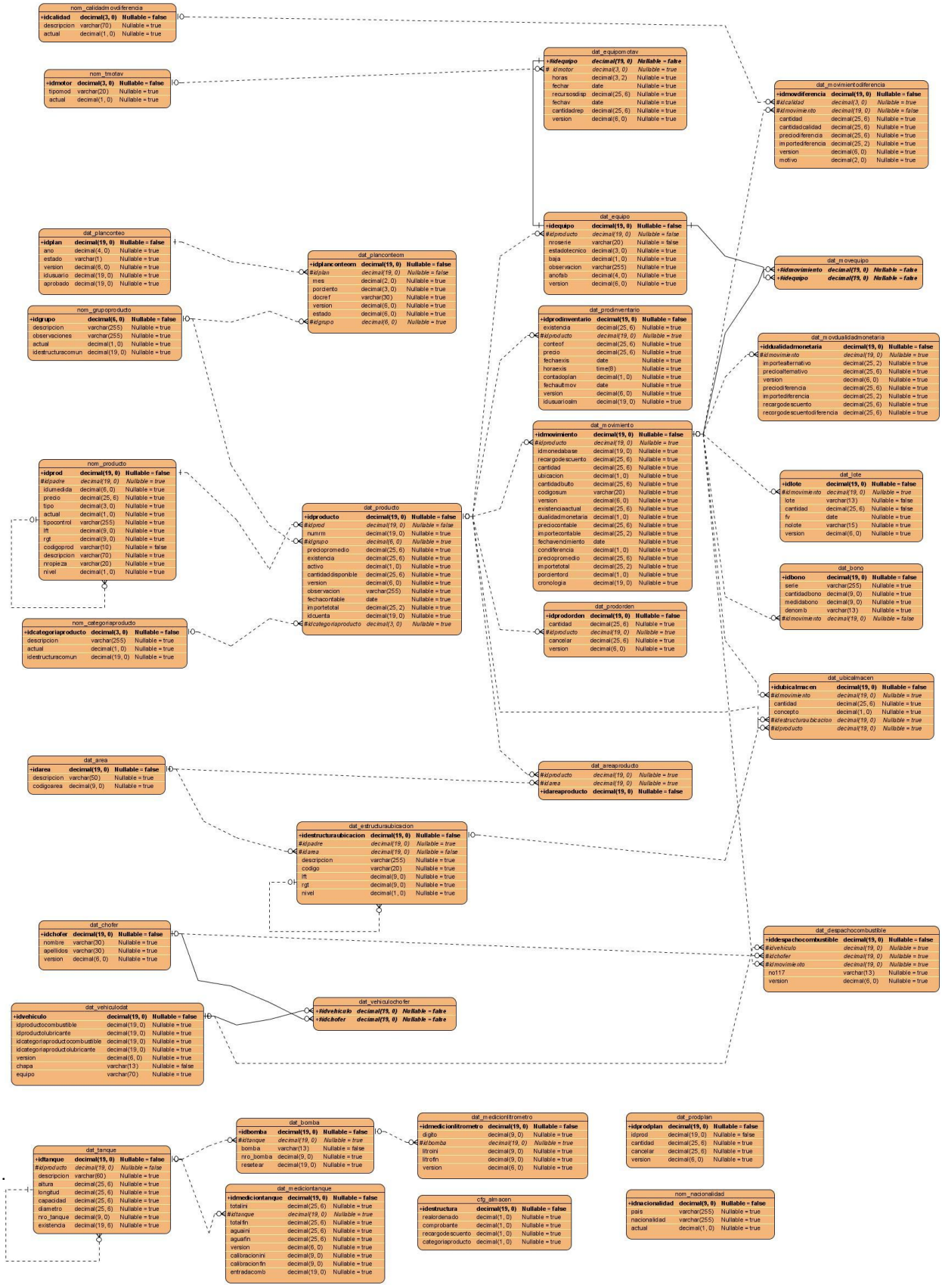
**Anexo 1.** Diagrama entidad relación general de la Base de Datos de la Línea Logística.



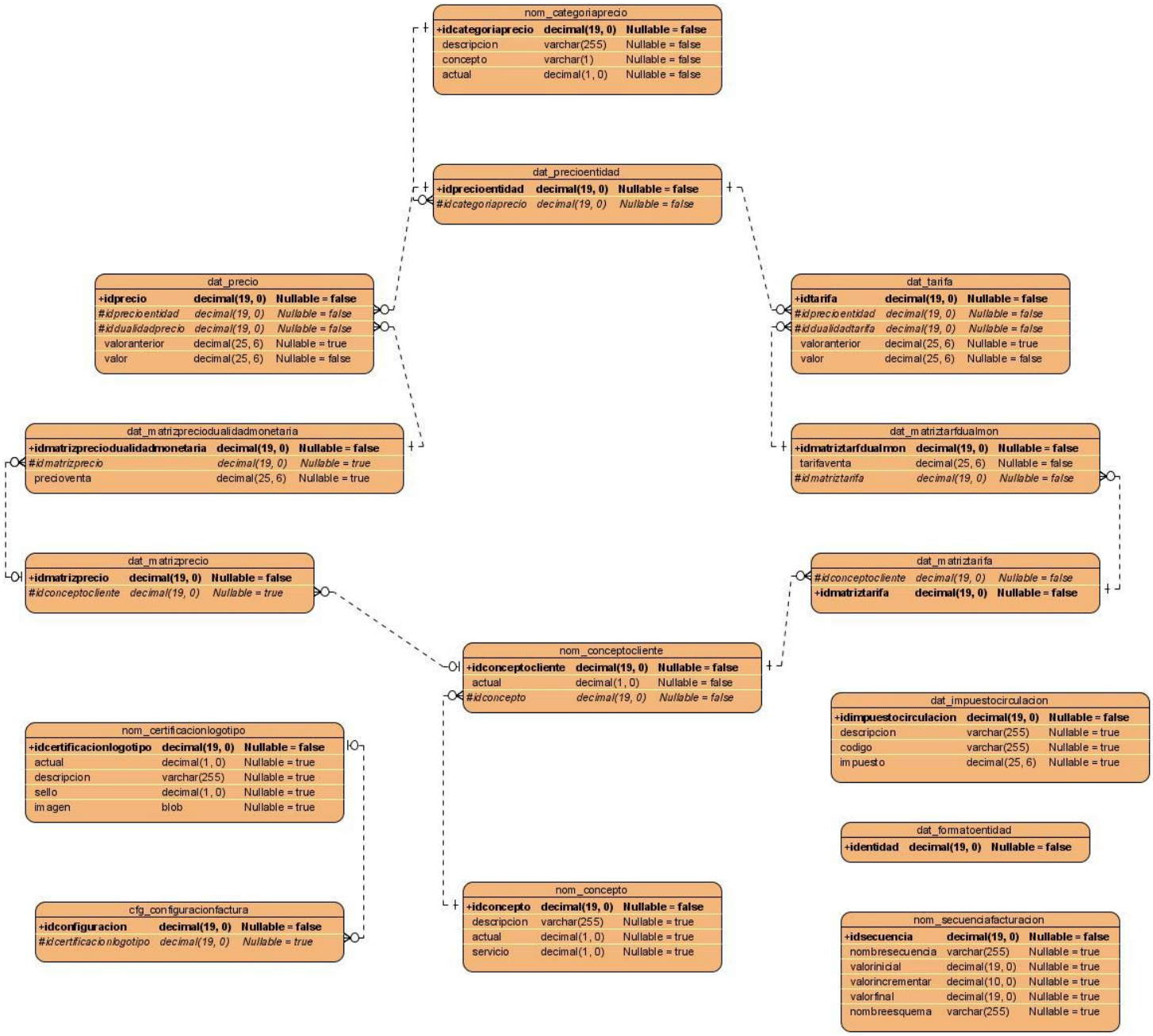
**Anexo 2.** Diagrama de Clases general de la Base de Datos.



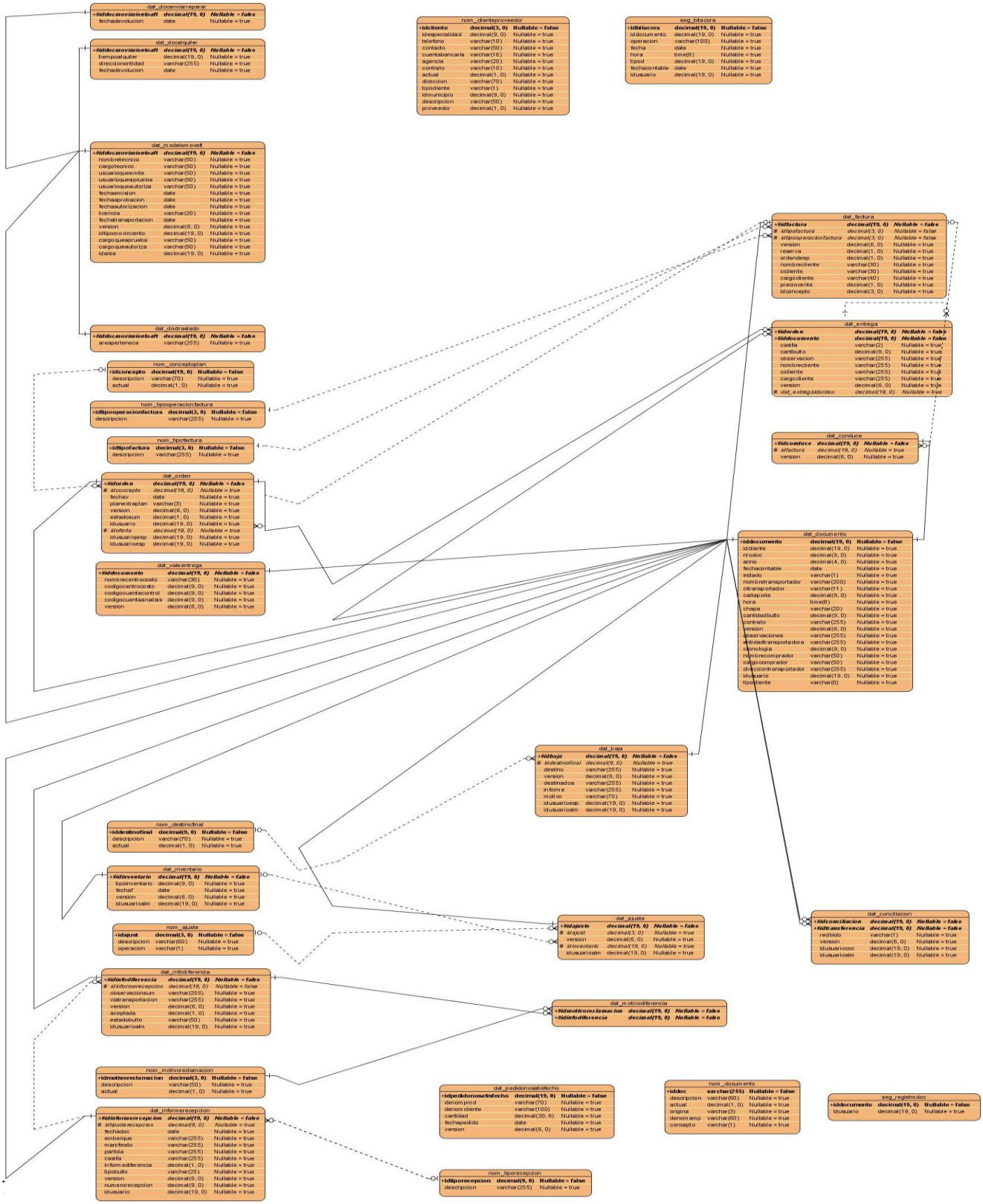
**Anexo 3.** Modelo entidad relación del esquema de Inventario.



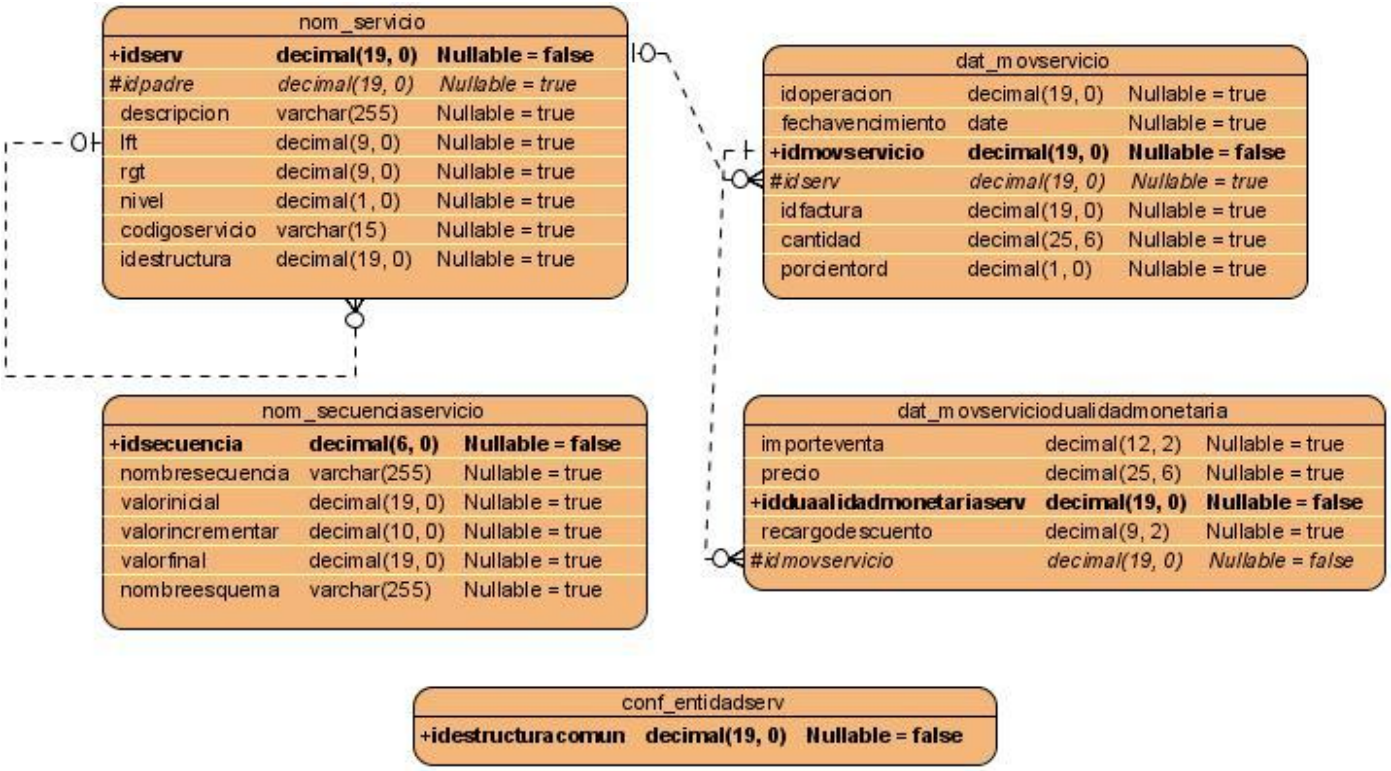
**Anexo 4.** Modelo entidad relación del esquema de Facturación.



**Anexo 5.** Modelo entidad relación del esquema de Documento.



**Anexo 6.** Modelo entidad relación del esquema de Servicio.



# Glosario de términos

**-Inventario:** Relación o estado detallado de los bienes qué constituyen el patrimonio de una persona o entidad.

**-Facturación:** Proceso de gestión empresarial que genera la factura.

**-Factura:** Documento que expide el vendedor y entrega al comprador en el que consta fecha de la operación, nombre de comprador y vendedor, importe, cantidad y descripción del producto o servicio, etc.

**-Módulo:** Es un componente autocontrolado de un sistema, el cual posee una interfaz bien definida hacia otros componentes; algo es modular si es construido de manera tal que se facilite su ensamblaje, acomodamiento flexible y reparación de sus componentes.

**-Componente:** Agrupación de elementos que hacen parte de un subsistema.

**-Subsistema:** Parte del sistema global con una interfaz razonablemente bien definida.

**-Modelo entidad relación:** Un diagrama o modelo entidad-relación (a veces denominado por su siglas, E-R "Entity relationship", o, "DER" Diagrama de Entidad Relación) es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

**-SAP AG** (Systeme, Anwendungen und Produkte) (Sistemas, Aplicaciones y Productos), con sede en Walldorf (Alemania), es el segundo proveedor de software empresarial en el mundo, después de Oracle. Como empresa, comercializa un conjunto de aplicaciones de software para soluciones integradas de negocios, entre ellas mySAP Business Suite, que provee soluciones escalables, es decir posibles de futura modificación, con más de 1.000 procesos de negocio, que la empresa clama se encuentran entre las mejores prácticas empresariales.