UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

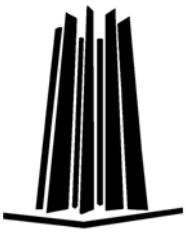
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"BASE DE DATOS PARA UN SISTEMA DE VENTAS"

T R A B A J O E S C R I T O
EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A :

E S T H E R R A M Í R E Z C H A V E R O

ASESOR: M. EN C. MARCELO PÉREZ MEDEL



MÉXICO, 2009.

AGRADECIMIENTOS

Agradezco a Dios todas las bendiciones que siempre me ha dado, pero sobre todo por haberme dado una maravillosa familia que siempre me ha apoyado.

A mi mamá por estar siempre a mi lado, apoyarme moral, sentimental y económicamente; sin ella no me hubiera sido posible terminar una carrera y por el ejemplo de vida que siempre ha sido para nosotras.

A mis hermanas por su apoyo incondicional y por siempre alentarme, ayudarme y cuidarme.

A mi tía Natalia porque siempre ha sido parte importante en mi vida y en la elección de mi carrera.

A Alfredo por apoyarme en todo momento, por ser el mejor amigo, compañero y esposo.

A mis hijos por su amor y por toda la alegría que me dan.

A la UNAM por sus excelentes instalaciones y profesores.

ÍNDICE

INTRODUCCIÓN

El estudio de los modelos de Bases de Datos son indispensables para poder desarrollar un proyecto de manera eficaz, pues en base al conocimiento que se tenga de éstas se podrá ser eficiente en el uso de recursos tecnológicos y humanos disponibles. Podemos ser más eficaces con los recursos tecnológicos, pues al saber el alcance del proyecto podemos definir qué manejador de Base de Datos es el más adecuado y sabiendo esto podemos determinar el hadware necesario para su buen funcionamiento. Se puede ser más preciso en el uso de recursos humanos, pues conociendo el alcance del proyecto se definirá la cantidad de personas necesarias para que el proyecto esté listo en el tiempo requerido.

El análisis de la Base de Datos nos ayudará en la etapa del análisis para la definición de los requerimientos y de esta manera determinar el alcance y costo del proyecto.

Dada la importancia del conocimiento y diseño de las Bases de Datos en el mundo laboral, el presente trabajo tiene la finalidad de analizar los conceptos de Bases de Datos para definir las ventajas y desventajas de sus diferentes modelos y utlizar alguno en la realización de un caso práctico que sea de utilidad para pequeñas empresas pues el objetivo es realizar un sistema eficiente y de bajo costo.

En el presente trabajo se sentarán las bases necesarias para desarrollar un sistema de ventas que tenga control sobre inventarios y que verifique el estatus de crédito de los clientes para determinar si son sujetos de ventas a crédito y los plazos, este sistema será de gran ayuda para las pequeñas empresas pues en automático el sistema le generará la entrega de mercancías si el cliente esta al corriente con sus pagos y tiene límite de crédito disponible, en caso contrario

quedaría detenido hasta que el área de crédito libere los pedidos, lo cual les ayudará pues sólo tendrán que revisar los pedidos que quedaron detenidos y analizar como se le venderá al cliente o si es necesario bloquearlo por su mal historial crediticio y en caso de estar bloqueado no se le podría generar ningún pedido y mandará mensaje para que se le informe al cliente, de esta manera se eficientarán las áreas de ventas y crédito.

CAPÍTULO 1 – BASES DE DATOS

I- Introducción

Tomé el "DIPLOMADO EN DISEÑO DE NEGOCIOS CON SQL Y ORACLE" porque para mí es muy importante el estarme preparando constantemente, para poder ofrecerle a la empresa en la cual esté trabajando mejores soluciones a los problemas que se les presenten, y de esta manera obtengan una solución que cubra sus expectativas y que su Base de Datos se mantenga trabajando de forma eficiente utlizando sus recursos de forma adecuada.

El presente trabajo presentará el desarrollo de la Base de Datos de un sistema de ventas, utilizando los conocimientos adquiridos en el diplomado, para la elaboración de la Base de Datos decidí utilizar el manejador de Base de Datos Microsoft® SQL Server.

Aunque el manejador de Base de Datos de Oracle es más robusto, también consume muchos recursos y este sistema está pensado para pequeñas empresas, por lo tanto el costo al que se desea distribuir el producto final debe ser bajo y que no implique un cambio de equipo para poder ser ejecutado, además para el correcto funcionamiento de este sistema es suficiente con las herramientas que nos proporciona Microsoft® SQL Server.

Es indispensable conocer los diferentes modelos que se pueden utilizar al generar una Base de Datos, pues conociéndolos podrémos hacer un análisis de las ventajas y desventajas de cada uno de ellos y determinar cuál es el que nos conviene utilizar para cada proyecto y/o negocio y por lo tanto los recursos serán ocupados de manera eficiente, y esto traerá como resultado que el sistema trabaje de forma rápida, ya que con una Base de Datos bien organizada es fácil poder encontrar y utilizar los datos requeridos por el usuario.

Un sistema amigable[1] es bueno porque le será más fácil al usuario trabajar con él, pero el diseño de la Base de Datos es también muy importante, debido a que de esto depende la eficiencia del sistema y esto también verá reflejado en el nivel de aceptación con el usuario puesto que si no tenemos una Base de Datos bien diseñada podemos no tener coherencia en los datos, lo cual afectará a todas las áreas involucradas con el sistema, un mal diseño también provoca que el sistema sea lento en su respuesta, lo cual provocará que el usuario no lo acepte, sin embargo si tenemos un buen diseño de la Base de Datos el usuario podrá trabajar de forma rápida y confiando en la información mostrada por el sistema, lo cual es vital para cualquier negocio.

Para el sistema de ventas, la Base de Datos será creada utilizando el modelo Relacional.

[1] Fácil de utilizar, con un diseño agradable y ayuda en todas las pantallas.

II – Fundamentos de las Bases de Datos

Desde que se empezaron a introducir ordenadores para automatizar la gestión de las empresas, las Bases de Datos han ido evolucionando constantemente, la primera generación de productos de Bases de Datos en red surgen a finales de los setentas y principios de los setentas.

En 1970 Codd[2] propuso el modelo relacional, el cual ha marcado la línea de investigación, pero IBM no explotó sus sugerencias hasta que sus rivales comerciales las pusieron en práctica. Larry Ellison[3] diseño la base de datos Oracle basándose en las ideas de Codd, este último definió las tres primeras Formas Normales que se aplican para la normalización de sistemas de Bases de Datos, aunque ahora también se encuentran los modelos orientados a objetos[4].

Una Base de Datos es un conjunto exhaustivo[5] no redundante de información, la cual debe estar estructurada y organizada independientemente de su utilización, es necesario tener en cuenta que las necesidades de los usuarios serán diferentes y por lo tanto la información requerida será diferente para cada usuario, motivo por el cual los usuarios necesitarán tener acceso a los programas que manipulan los datos que se encuentren en esta Base de Datos.

Si aplicamos el concepto de Base de Datos en una empresa, podemos definirla como una colección de datos usados por el sistema de aplicaciones de la empresa. Cabe señalar que un archivo por sí mismo no constituye una Base de Datos, lo que la constituye es la forma en que está organizada la información. Los datos deben estar organizados para poder dar servicio de forma eficiente a varias aplicaciones al mismo tiempo, debe centralizar los datos y minimizar las redundancias.

[2] Doctor en Ciencias de la Computación, nacido en Pórtland Bill (Reino Unido); trabajo como programador matemático para IBM; en 1981 recibió el Premio Turing de Ciencias de la Computación, murió en el 2003.
[3] Nació en Nueva York, es el fundador y figura principal de Oracle
[4] Este modelo trata de almacenar en la Base de Datos los objetos completos (estado y comportamiento); incorpora todos los conceptos importantes: Encapsulación, Herencia, Polimorfismo
[5] Íntegro, completo e ilimitado

Aunque si bien es cierto que la arquitectura física de una Base de Datos dependerá de la configuración del hardware, es necesario que tanto la descripción lógica como la física tendrá que adecuarse para satisfacer los requerimientos de funcionalidad y de comportamiento para el acceso de los usuarios.

Algunas de las ventajas que tenemos al utilizar las Bases de Datos como plataformas para el desarrollo de Sistemas son:

- Independencia de datos y tratamiento[6].
- Coherencia de resultados[7]
- Reducción de redundancias
- Eficiencia en la gestión de almacenamiento
- Globalización de la información
- Permite compartir información
- Integridad en la información[8]

Para crear una Base de Datos se deben realizar los siguientes diseños:

- El diseño conceptual de una Base de Datos es el proceso de construcción de un modelo de los datos utilizados en el negocio, de forma independiente de todas las consideraciones físicas, este describe cómo se deben agrupar los elementos de datos en la Base de Datos. El proceso de diseño identifica las relaciones entre los elementos de datos y la manera más eficiente de agrupar los elementos de datos entre sí para cumplir con los requerimientos de la información.

- El diseño lógico de una Base de Datos es un modelo abstracto de la Base de Datos desde una perspectiva de negocios. Requiere de una descripción

[6] Consiste en la capacidad de modificar el esquema de una Base de Datos sin realizar cambio en las aplicaciones.
[7] Vigilar que la información que se encuentre repetida se actualice de forma simultánea
[8] Proteger los datos ante fallas de hardware, datos introducidos por usuarios, o cualquier circunstancia capaz de corromper la Base de Datos

detallada de las necesidades de información del negocio. Describe cómo los elementos en la Base de Datos han de quedar agrupados.

- El diseño físico muestra como la Base de Datos se ordena en realidad en los dispositivos de almacenamiento de acceso directo. El diseño físico debe ser llevado a cabo por los especialistas en Bases de Datos.

Entre los diferentes tipos de Bases de Datos tenemos:

- Bases de Datos documentales[9]: Son las derivadas de la necesidad de disponer de toda la información en el puesto de trabajo y de minimizar los tiempos del acceso a aquellas informaciones que no están estructuradas convenientemente. Esto se debe a que la procedencia de la información es muy variada. Las Bases de Datos documentales han experimentado un fuerte auge durante estos últimos años, impulsado sobre todo por la popularización de Internet y la consiguiente saturación de información textual que ha traido la WWW[10], así como por el reciente interés de las grandes empresas por gestionar el conocimiento almacenado en documentos.
- Bases de Datos distribuidas: Es aquella que se almacena en más de un lugar físico. Parte de la Base de Datos se almacena físicamente en un lugar y otras partes se almacenan y mantienen en otros lugares. La Base de Datos central puede ser particionada de manera que cada procesador remoto tenga los datos necesarios sobre los clientes para servir a su área local. Los cambios en los archivos pueden ser justificados en la Base de Datos central sobre las bases de lotes. Los sistemas distribuidos reducen la vulnerabilidad de un lugar único. Permiten incremento en la potencia de los sistemas al adquirir varias computadoras. Incrementan el servicio y la

[9] Ejemplos de sistemas de gestión de Bases de Datos documentales son Knosys, Inmagic, ISIS, BRS, entre otros
[10] World Wide Web

posibilidad de respuesta de los usuarios locales. Una desventaja es que los sistemas distribuidos dependen de las líneas de telecomunicaciones, las cuales a su vez son vulnerables. Existen diversas formas de afrontar el problema del diseño de la distribución. Las más usuales se muestran en la figura 1.II.1

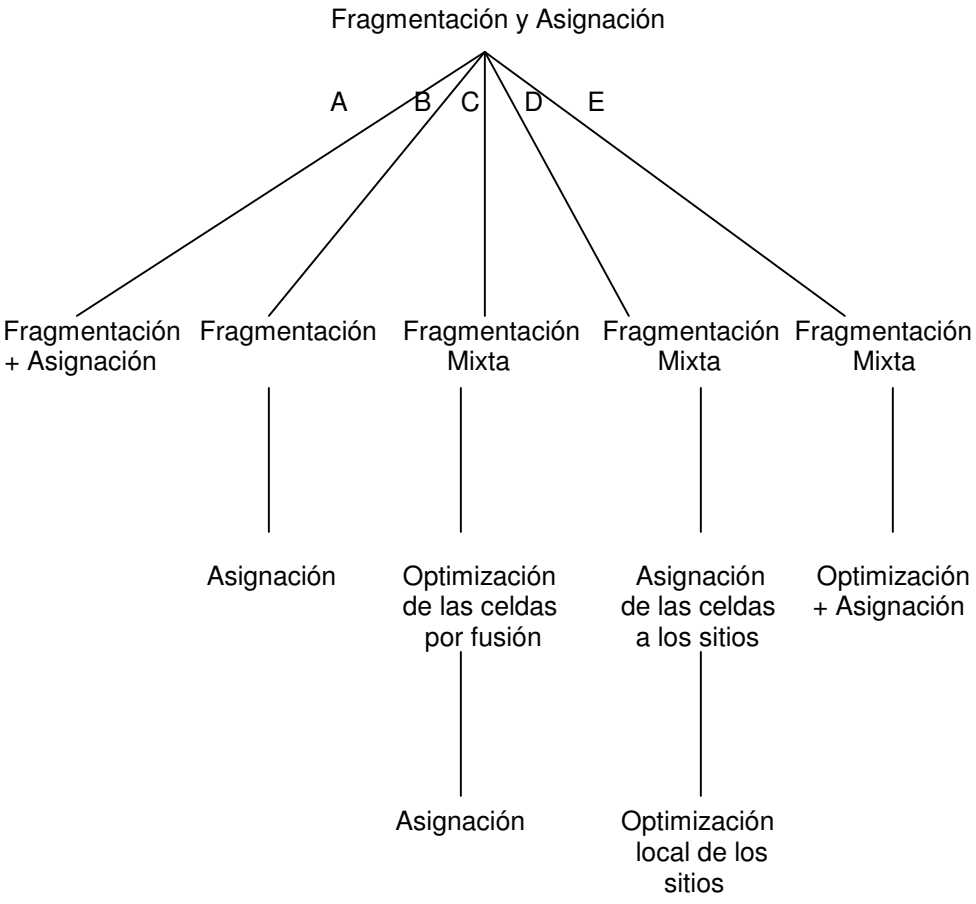


Figura 1.II.1

- Bases de Datos orientadas a objetos[11]: Éstas son capaces de almacenar tanto procesos como datos. Por este motivo las bases orientadas al objeto deben tener la capacidad de almacenar información no convencional. Este

[11] Algunos de los productos de las BDOO son: Gemstone, Itasca, Objectivity, Object Store, Ontos, Versant

tipo de Bases de Datos deriva directamente de la programación orientada a objetos. Los principales conceptos que se utilizan en las BDOO[12] son: Identidad de objetos, Constructores de tipos, Encapsulamiento, Compatibilidad con los lenguajes de programación, Jerarquías de tipos y herencia, Manejo de objetos complejos, Polimorfismo y sobre carga de operadores, Creación de versiones.

III - Modelos de Bases de Datos

Existen distintos modos de organizar la información y representar las relaciones entre los datos en una Base de Datos. Los Sistemas administradores de Bases de Datos usan uno de los tres modelos lógicos de Bases de Datos para hacer seguimiento de las entidades, atributos y relaciones. Los tres modelos lógicos de Bases de Datos son: el jerárquico, de redes y el relacional.

- Modelo Relacional de Datos: Representa al mundo real mediante tablas relacionadas entre sí por columnas, como sencillas tablas de dos dimensiones llamadas relaciones, son semejantes a los archivos planos, pero la información en más de un archivo puede ser fácilmente extraída y combinada. Cada línea se llama tupla, las columnas se llaman atributos. Este modelo es el más empleado, debido a las ventajas que ofrece sobre los otros dos modelos. Figura 1.III.1

[12] Bases de Datos Orientadas a Objetos

Atributo 1	Atributo 2	Atributo 3	Atributo 4	Atributo 5	
Clave	Nombre	Dirección	RFC	Teléfono	Tupla 1
1	XXXX	XXXX	XXXXX	XXXXX	Tupla 2
2	XXXX	XXXX	XXXXX	XXXXX	Tupla 3
3	XXXX	XXXX	XXXXX	XXXXX	Tupla 4
4	XXXX	XXXX	XXXXX	XXXXX	Tupla 5

Figura 1.III.1

- Modelo de Red: En este modelo se representa al mundo real como registros lógicos que representan a una entidad y que se relacionan entre sí. Este modelo permite la representación de muchos a muchos, de tal forma que cualquier registro dentro de la Base de Datos puede tener varias ocurrencias superiores a él. El modelo de red evita redundancia en la información, a través de la incorporación de un tipo de registro denominado el conector. Figura 1.III.2

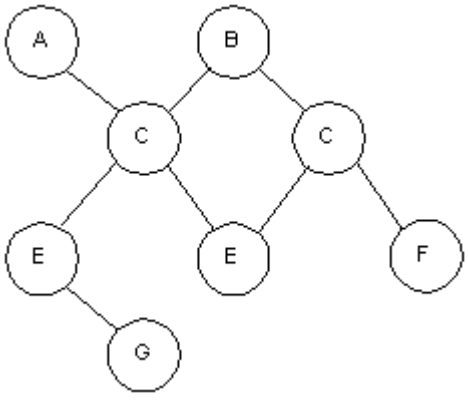


Figura 1.III.2

- Modelo Jerárquico[13]: Los primeros sistemas administradores de Bases de Datos eran jerárquicos. Tiene forma de árbol invertido. Un padre puede tener varios hijos, pero cada hijo sólo puede tener un padre. El modelo jerárquico sólo admite relaciones 1 : 1 ó 1 : N. Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos; almacena la información en una estructura jerárquica que enlaza los registros en forma de estructura de árbol. Figura 1.III.3 Esta relación jerárquica no es estrictamente obligatoria, de manera que puede establecerse relaciones entre nodos hermanos. En este caso la estructura en forma de árbol se convierte en una estructura en forma de grafo dirigido, y entonces se convierte en el modelo de red.

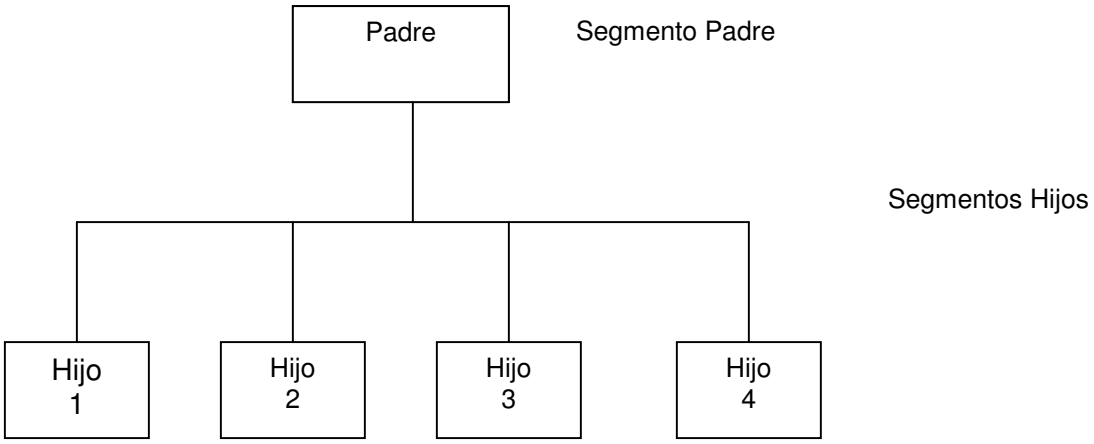


Figura 1.III.3

[13] Las Bases de Datos Jerárquicas fueron concebidas en los años '60. El primer modelo de Base de Datos fue la Base de Datos en red

IV - Capas de un sistema de Información

Los Sistemas de Base de Datos pueden ser estudiadas desde tres niveles distintos:

1. Nivel Físico. Es el nivel real de los datos almacenados, es decir, cómo se almacenan los datos. Este nivel es usado por muy pocas personas que deben estar cualificadas para ello. Este nivel lleva asociada una representación de los datos, que es lo que denominamos Esquema Físico.
2. Nivel Conceptual. Es el correspondiente a una visión de la base de datos desde el punto de visto del mundo real. Tratamos con la entidad u objeto representado, sin importarnos como está representado o almacenado. Este nivel lleva asociado el Esquema Conceptual.
3. Nivel Visión. Son partes del esquema conceptual. El nivel conceptual presenta toda la base de datos, mientras que los usuarios por lo general sólo tienen acceso a pequeñas parcelas de ésta. El nivel visión es el encargado de dividir estas parcelas.

Los tres niveles vistos, componen lo que conocemos como arquitectura de base de datos a tres niveles, figura 1.iv.1

A menudo el nivel físico no es facilitado por muchos DBMS[14], esto es, no permiten al usuario elegir cómo se almacenan sus datos y vienen con una forma estándar de almacenamiento y manipulación de los datos

La arquitectura a tres niveles se puede representar como sigue (figura 1.iv.1):

- Esquema de Visión.
- Esquema Conceptual
- Esquema Físico

[14] En inglés Database Management System o en español SGBD (Sistemas de Gestión de Bases de Datos)

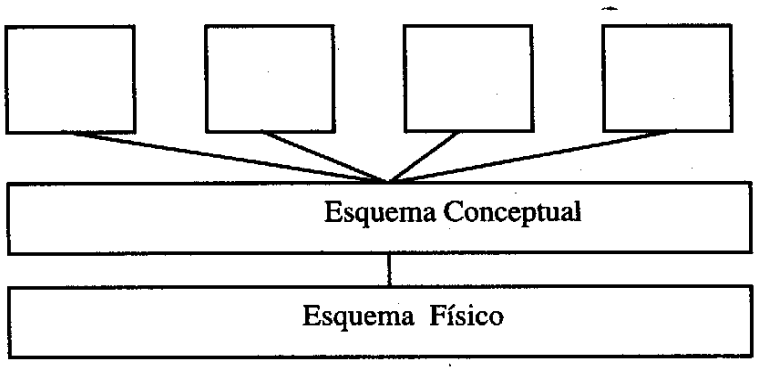


Figura 1.iv.1

V - Sistema de Base de Datos

Un Sistema de Base de Datos estará formado por:

- Personas
- Máquinas
- Programas: Son los encargados de manejar los datos, son conocidos como DBMS (Data Base Management System) o también SGBD[15] (Sistema Gestor de Base de Datos). Los DBMS tienen dos funciones principales que son:
    - La definición de las estructuras para almacenar los datos.
    - La manipulación de los datos.
- Datos: Es lo que se conoce como Base de Datos propiamente dicha. Para manejar estos datos utilizamos una serie de programas

La Base de Datos es una colección estructurada de tablas que nos permiten guardar grandes cantidades de datos en las cuales se almacena información de cualquier tipo. En dichas tablas la información se guarda en campos, aunque la Base de Datos no es sólo la colección de tablas en donde están los datos, sino que en dichas tablas se encuentra la estructura de los datos, por ejemplo para conocer la longitud de cada campo, hay que conocer como se llama el campo así como el tipo de datos en dicho campo, porque puede contener letras, números, imágenes, dependiendo de la estructura de la Base y del sistema gestor.

El DBMS (Data Base Management System) es un conjunto de numerosas rutinas de software interrelacionadas y cada una de ellas es responsable de una determinada tarea.

[15] Son un tipo de Software muy específico dedicado a servir de interfaz entre la Base de Datos, el usuario y las aplicaciones que la utilizan.

El DBMS (Data Base Management System): son sistemas desarrollados para hacer posible el acceder a datos integrados que atraviesan los límites operacionales, funcionales u organizacionales de una empresa, es decir son un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos y esta implementado por:

DDL: Lenguaje de Definición de Datos

DML: Lenguaje de Manipulación de Datos

SQL[16] : Lenguaje de Consulta.

Una de las ventajas del DBMS es que puede ser invocado desde programas de aplicación que pertenecen a Sistemas Transaccionales escritos en algún lenguaje de alto nivel, para la creación o actualización de las Bases de Datos, o bien para efectos de consulta a través de lenguajes propios que tienen las Bases de Datos o lenguajes de cuarta generación.

Los objetivos en el uso de un DBMS (Data Base Management System) son:

- La eficiencia y la eficacia para el manejo de datos
- Disponibilidad, permitiendo la accesibilidad de datos
- Consistencias, disponibilidad y calidad de datos
- Evolución, para adaptarse al entorno
- Integridad, en el nivel de los datos así como en el sistema.

Las ventajas de utilizar un DBMS (Data Base Management System) son:

- Independencia de datos

[16] Structured Query Language

- Accesibilidad limitada
- Datos al día y sin redundancias
- Consistencia
- Interfaz única
- Entrada directa a los datos
- Recuperación por diferentes accesos
- Función completa de interrogantes
- Estandarización
- Seguridad

Existen cuatro productos básicos en los Sistema de Gestión de Base de Datos para sistemas propietarios. Estos son:

- Repositorio / diccionario / directorio / enciclopedia
- Desarrollador de aplicaciones (Prototipo, Lenguaje(s) de programación, basado en interpretes).
- Lenguaje de consulta
- Lenguaje reporteador

En los sistemas de procesamiento de archivos la información necesita ser guardada y manipulada para que sea útil. El sistema de Procesamiento de Archivos: Tiene una serie de inconvenientes que son reducidos en los SGBD (Sistemas de Gestión de Base de Datos), a continuación mencionaré algunos:

a) Dificultad de Acceso a ciertos datos o información: Si no existen programas para acceder o calcular cierta información, no puede accederse a ella.
b) Aislamiento de Datos: Los datos pueden estar en varios archivos con distintos formatos, que complican la creación de programas nuevos.
c) Falta de Integridad: Es complicado mantener ciertas condiciones en la información.

d) Problemas en las operaciones: A veces es esencial que para la consistencia de la Base de Datos se efectúen varias operaciones como si fueran una única operación, evitando que se produzcan fallos en medio de dicha operación.
e) Problemas en el Acceso Concurrente: Si varios usuarios acceden a la vez a un dato pueden producirse errores.
f) Problemas de Seguridad: Dificultad para controlar que ciertos usuarios no accedan a ciertos datos

Viendo la necesidad de mejorar este estándar se desarrollaron los "Sistemas gestores de Base de Datos Relaciónales" (SGBDR) cuyas características hacen al sistema mucho más eficiente que los sistemas de manejos de archivos. Algunas de las características son que existe sólo una copia de los datos para que todos los programas trabajen con ella, esto es lo que se denomina obtención de redundancia mínima y de esta manera se podrá eliminar la inconsistencia de los datos.

Una característica importantes que tiene un modelo de Base de Datos relacional es la capacidad de interactuar en un ambiente cliente / servidor donde los usuarios pueden trabajar con un conjunto único de datos alojados en un servidor y donde varios clientes podrían estar trabajando al mismo tiempo, y el procesamiento puede ser en lína o por lotes.

A continuación describiré brevemente en que consiste el procesamiento en línea y por lotes:

a) En un sistema de procesamiento por lotes, organizar la memoria en particiones fijas es simple y efectivo: cada trabajo se carga en la memoria cuando le toque su turno, y se queda en memoria hasta que termine.

b) Procesamiento en línea: el procesamiento en línea implica que los programas se ejecuten de tal forma que los datos se actualicen de inmediato en los archivos de la computadora. A este tipo de procesamiento se le conoce también como tiempo real.

Las aplicaciones de tiempo real son indispensables en aquellos casos en que los datos contenidos en los archivos se modifican varias veces en el transcurso de un día y se consultan en forma casi inmediata con las modificaciones que se efectuaron.

La diferencia principal entre un sistema de procesamiento de archivos y un DBMS (Data Base Management System) radica en la manera de almacenar, recuperar y actualizar los datos; lo anterior se debe a que con los sistemas de procesamiento de archivos, los datos se guardan en diversos archivos y muchas veces dichos archivos son de diferente formato, ocasionando problemas como la redundancia que trae consigo almacenamiento y dificulta el acceso a los mismos. Dicha redundancia también implica inconsistencias en lo que se tiene almacenado.

Algunas de las ventajas de procesamiento de Base de Datos son:

a) Independencia de datos y tratamiento.
b) Coherencia de resultados
    i) Acciones lógicamente únicas
    ii) Se evita inconsistencia
c) Mejora en la disponibilidad de datos
    i) No hay dueño de datos.
    ii) Ni aplicaciones ni usuarios
d) Cumplimiento de ciertas normas
    i) Restricciones de seguridad
    ii) Accesos.
    iii) Operaciones.
e) Más eficiente gestión de almacenamiento

A continuación veremos brevemente los leguajes de tercera y cuarta generación.

Los lenguajes de tercera generación (3GL; third-generation languages) son los lenguajes propiamente como los conocemos (poseen instrucciones, funciones, sintaxis, semántica); ya trascienden el uso de los términos nemotécnicos. Una instrucción puede indicar 1 o más tareas para la computadora. Pueden dividirse como sigue:

i) Orientados a problemas
ii) Orientados a procedimientos: Los lenguajes orientados al procedimiento requieren que los programadores resuelvan problemas de programación utilizando la lógica de programación tradicional. Se dividen en:
    (1) Lenguajes empresariales. Los 3GL orientados a las empresas se diseñan para ser instrumentos efectivos para desarrollar sistemas de información empresarial. La fuerza de las 3GL empresariales consiste en su capacidad de almacenar, recuperar y manejar datos alfanuméricos.
    (2) Lenguajes científicos. Estos lenguajes del tipo de una fórmula algebraica están diseñados en particular para satisfacer las necesidades de procedimientos repetitivos, la expresión y la solución de ecuaciones matemáticas y demás operaciones pertinentes
iii) Orientados a objetos

Los lenguajes de cuarta generación (4GL - Generadores de aplicaciones que no dependen de una metodología). Son de alto nivel y amigables; los programadores que utilizan 4GL sostienen que experimentan incrementos en la productividad de 200 a 1,000% en comparación con los lenguajes orientados al procedimiento de la tercera. Hay dos tipos de 4GL:

iv) Orientados a la producción: Están diseñados sobre todo para profesionales en la computación.
   (1) Lenguajes orientados al procedimiento
   (2) Lenguajes de la cuarta generación orientados al usuario
v) Orientado al usuario

A continuación enmumeraré algunas características de los sistemas de cuarta generación:

a) Debe tener integrado un manejador de Base de Datos
b) Debe contar con un reporteador basado en objetos
c) Debe haber independencia total entre los datos y las aplicaciones
d) Debe contar con un diccionario o repositorio de datos
e) Debe proporcionar apoyo para un esquema de seguridad por usuarios
f) Debe contar con un lenguaje de rastreo basado en ejemplos
g) Debe hacer hincapié en el modelo de datos y no en la programación del código
h) Debe poseer herramientas para la fácil documentación de análisis, diseño, y modelo de datos.
i) Desventajas: Requieren conocimientos especializados y su operación requiere personal calificado.

CAPÍTULO 2 – PLANTEAMIENTO DEL PROYECTO

I - Objetivo

Mediante los conocimientos adquiridos a través del diplomado decidí generar una Base de Datos para un sistema de ventas; con este trabajo pretendo establecer las bases para en un futuro poder realizar este sistema para su comercialización en pequeñas empresas, en este momento no lo podría realizar porque aunque con el diplomado aprendí como manejar de manera optima la Bases de Datos, necesito capacitarme para decidir en que lenguaje de programación motar el sistema, analizando los costos y beneficios de cada opción.

II - Alcance del Proyecto

Realizar un proyecto para un sistema de ventas en el cual se puedan realizar pedidos y de acuerdo a la información alimentada en el sistema al momento de generar el pedido deberá verificar lo siguiente:

- Deberá de verificar las condiciones de pago del cliente.
- Si el cliente tiene condiciones de pago diferentes a Pago Inmediato, verificar si su línea de crédito es suficiente para generar la entrega, de lo contrario deberá quedar detenido el pedido hasta que el área de crédito lo libere.
- Si el cliente tiene línea de crédito suficiente para generar la entrega, verificar si tiene más de 5 facturas vencidas, de ser así detener su pedido hasta que el área de crédito lo libere.
- Si el pedido no se detiene por crédito, generar la entrega de manera automática.
- Si se generó la entrega de manera automática, descontar el material surtido del inventario y generar la factura y su partida de crédito.

III – Planeación del Proyecto

Objetivo de la Organización: Obtener un sistema con el cual pueda llevar a cabo las ventas de la compañía, controlando los inventarios y los pagos.

Objetivo del Cliente: Que al momento de generar el pedido el sistema genere de manera automática la entrega descontando del inventario el material y genere la factura si el cliente tiene crédito suficiente, de lo contrario sólo se generará el pedido y esperará a que el área de crédito lo libere

Objetivo del Sistema: Generar un sistema en el cual en el pedido se puedan calcular los precios y descuentos a los que tiene derecho el cliente y con el cual se genere la factura de manera automática y se descuenten del inventario los materiales que fueron facturados controlando de forma automática los inventarios de los productos y la línea de crédito asignadas a cada cliente, para permitir determinar si la venta procede de acuerdo a estas condiciones.

CAPÍTULO 3 – DESARROLLO DEL PROYECTO

I - Desarrollo del Sistema

i - Áreas Funcionales

Las áreas funcionales de la empresa que intervienen en el proyecto se representan en la figura 2.1 y son las siguientes:

- Logística: La cual se encargará de dar de alta los materiales y el inventario
- Crédito: Se encargará de dar de alta a los clientes, así como sus límites de crédito y capturar los pagos realizados de las facturas generadas, además de controlar la liberación de los pedidos que sean detenidos por sobrepasar la línea de crédito o por tener más de 5 facturas vencidas.
- Ventas: Es el área encargada de dar de alta, modificar y borrar los pedidos, así como darle mantenimiento a los precios y descuentos.

Área	Sub Área	Operación	Actividad	Documentos Que Genera	Origen	Destino
Ventas	Crédito	Altas	Alta de Clientes Define límite crédito		Crédito	Ventas
Ventas	Logística	Altas	Alta de Productos Captura de Inventario		Logística	Ventas
Ventas	Ventas	Altas	Alta, bajas y modificaciones de Precios y descuentos		Ventas	Ventas
Ventas	Captura	Altas Bajas Modificaciones	Alta, bajas y modificaciones de pedidos	Pedido	Ventas	Ventas
Ventas	Logística	Genera Entrega	Salida de Mcía de inventarios	Entrega	Ventas	Logística
Logística	Crédito	Genera Factura	Generación de Factura a partir de la entrega	Factura	Ventas	Crédito
Crédito	CXC	Cobro de Facturas	Aplicación de pagos a facturas		Crédito	Ventas

Figura 2.1 Áreas Funcionales

ii – Modelo de la Organización

En la figura 2.2 podemos ver el diagrama del modelo de la organización relacionado con el sistema de este proyecto, la herramienta utilizada para este diagrama fue visio.

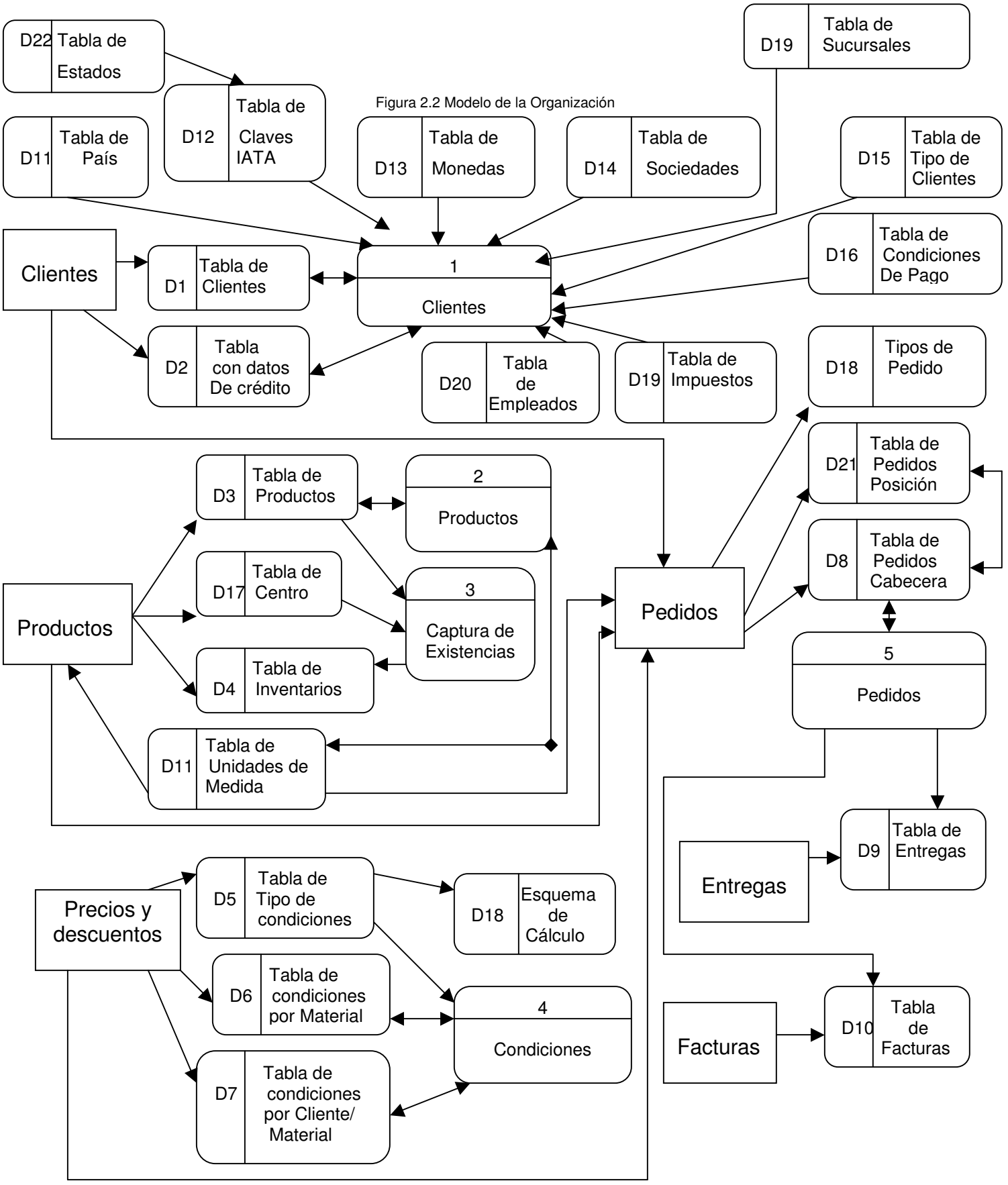
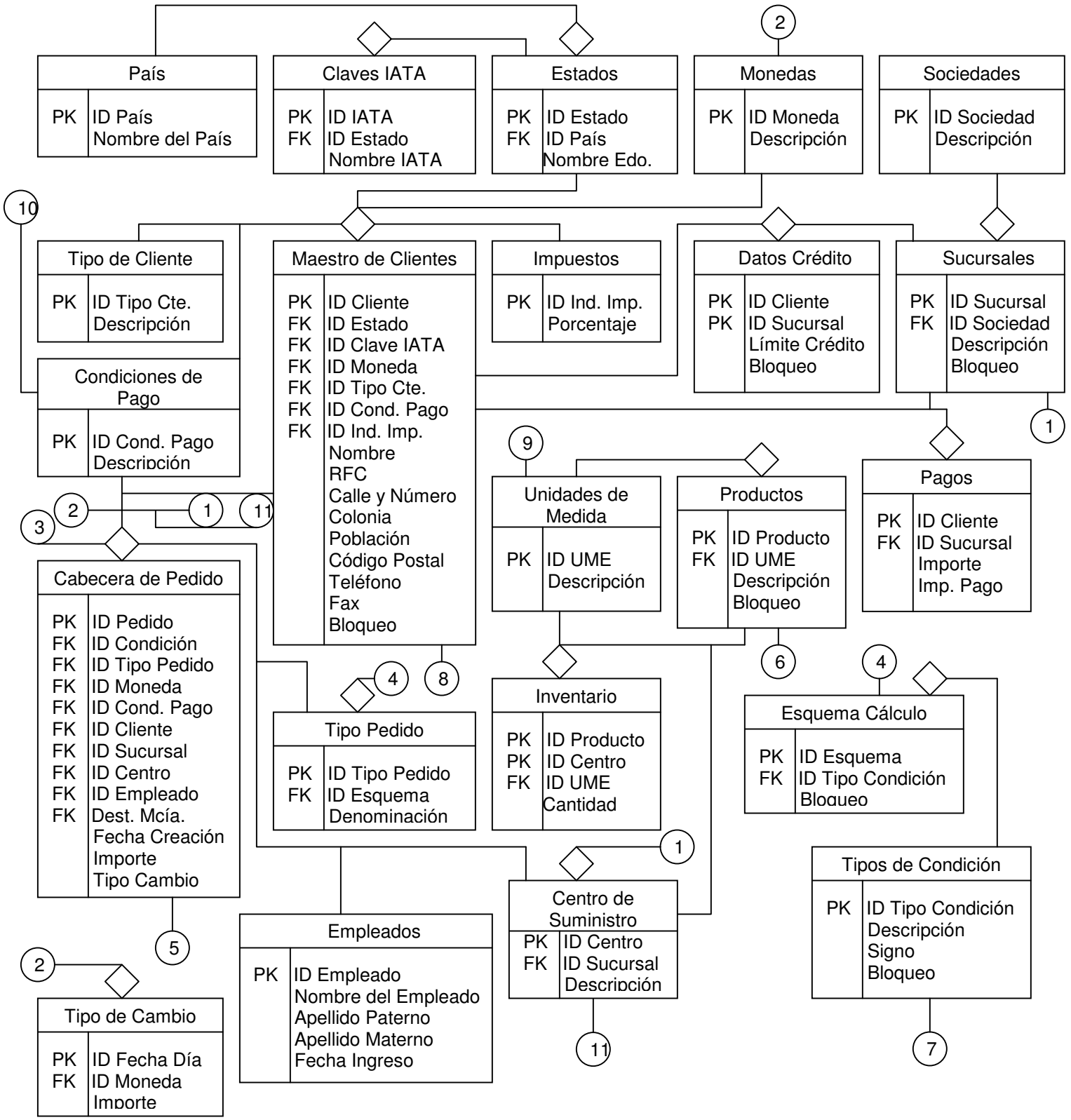


Figura 2.2 Modelo de la Organización

iii – Diagrama Entidad-Relación

A partir del modelo de la organización, podemos obtener el diagrama de entidad relación y los campos que contendrá cada una de ellas (figura 2.3)


País
PK ID País
Nombre del País
Claves IATA
PK ID IATA
FK ID Estado
Nombre IATA
Estados
PK ID Estado
FK ID País
Nombre Edo.
Monedas
PK ID Moneda
Descripción
Sociedades
PK ID Sociedad
Descripción
Tipo de Cliente
PK ID Tipo Cte.
Descripción
Maestro de Clientes
PK ID Cliente
FK ID Estado
FK ID Clave IATA
FK ID Moneda
FK ID Tipo Cte.
FK ID Cond. Pago
FK ID Ind. Imp.
Nombre
RFC
Calle y Número
Colonia
Población
Código Postal
Teléfono
Fax
Bloqueo
Impuestos
PK ID Ind. Imp.
Porcentaje
Datos Crédito
PK ID Cliente
PK ID Sucursal
Límite Crédito
Bloqueo
Sucursales
PK ID Sucursal
FK ID Sociedad
Descripción
Bloqueo
Condiciones de Pago
PK ID Cond. Pago
Descripción
Unidades de Medida
PK ID UME
Descripción
Productos
PK ID Producto
FK ID UME
Descripción
Bloqueo
Pagos
PK ID Cliente
FK ID Sucursal
Importe
Imp. Pago
Cabecera de Pedido
PK ID Pedido
FK ID Condición
FK ID Tipo Pedido
FK ID Moneda
FK ID Cond. Pago
FK ID Cliente
FK ID Sucursal
FK ID Centro
FK ID Empleado
FK Dest. Mcía.
Fecha Creación
Importe
Tipo Cambio
Tipo Pedido
PK ID Tipo Pedido
FK ID Esquema
Denominación
Inventario
PK ID Producto
PK ID Centro
FK ID UME
Cantidad
Esquema Cálculo
PK ID Esquema
FK ID Tipo Condición
Bloqueo
Empleados
PK ID Empleado
Nombre del Empleado
Apellido Paterno
Apellido Materno
Fecha Ingreso
Centro de Suministro
PK ID Centro
FK ID Sucursal
Descripción
Tipos de Condición
PK ID Tipo Condición
Descripción
Signo
Bloqueo
Tipo de Cambio
PK ID Fecha Día
FK ID Moneda
Importe
10
2
1
11
3
9
4
8
6
5
7

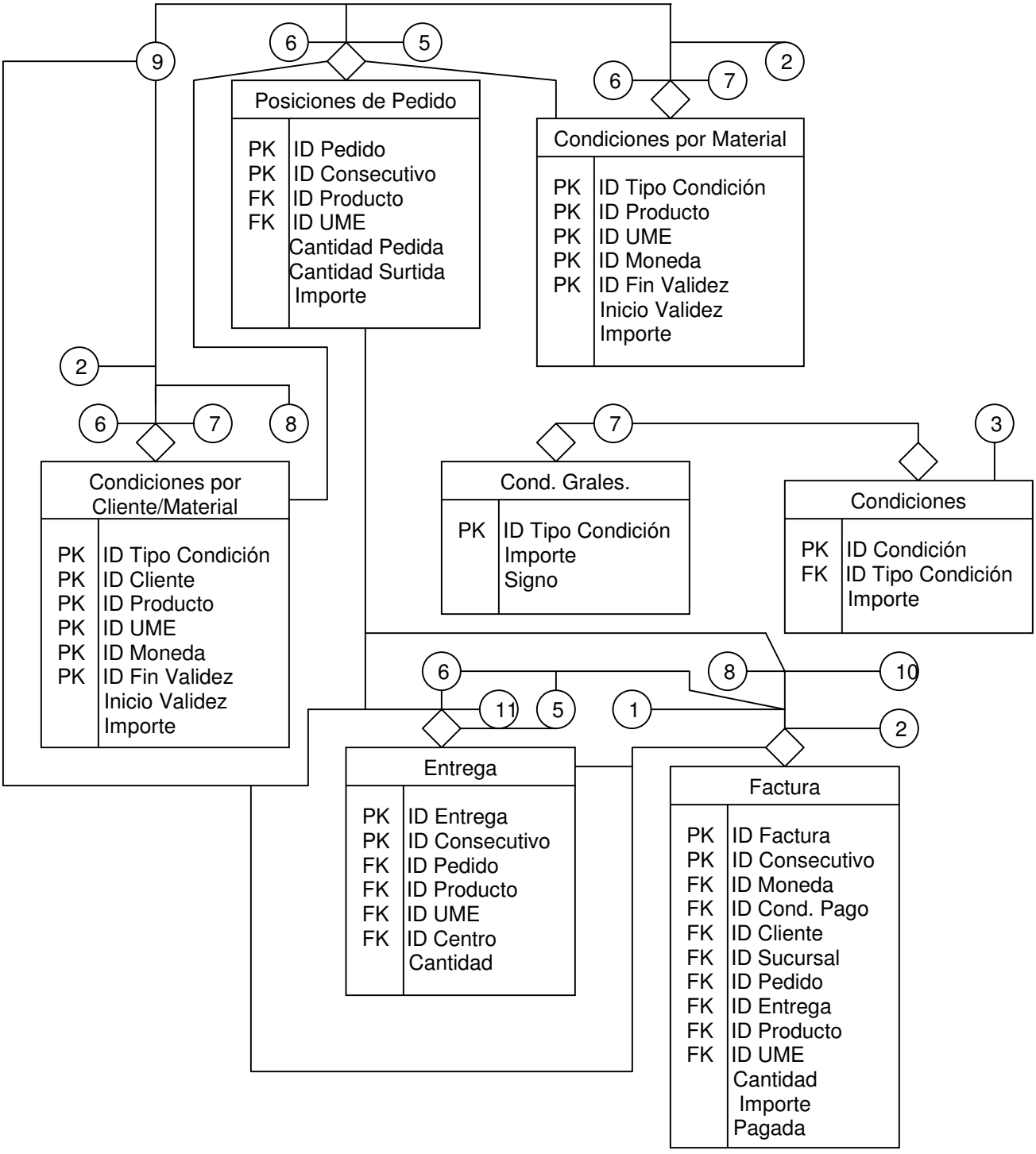


Figura 2.3 Diagrama Entidad-Relación

iv – Diccionario de Datos

A partir del diagrama de Entidad-Relación mostrado en la figura 2.3, podemos obtener el diccionario de datos (Figura 2.4)

Diccionario de Datos

Nombre	Alias	Tipo	Tamaño	No Nulo	Requerido	Entidad / Relación
Identificador de País	Id_Pais	Char	4	X	X	País, Estados
Descripción del ID	Denominacion	Char	40	X	X	Todas
Identificador IATA	Id_IATA	Char	3	X	X	IATA, Clientes
Identificador de Estado	Id_Estado	Char	3	X	X	Estados, IATA, Clientes
Identificador de Moneda	Id_Moneda	Char	3	X	X	Monedas, Clientes, C_Pedido, T_Cambio, Con_Material, Con_Cte_Mat, Factura
Identificador de Sociedad	Id_Sociedad	Smallint	2	X	X	Sociedad, Sucursal
Identificador de Sucursal	Id_Sucursal	Smallint	2	X	X	Sucursal, D_Credito, C_Pedido, Entrega, Factura, Pagos
Identificador de Tipo de Cliente	Id_TCliente	Smallint	2	X	X	T_Cliente, Clientes
Identificador de Condiciones de Pago	Id_CPago	Smallint	2	X	X	C_Pago, Clientes, C_Pedido, Factura
Identificador de Cliente o Destinatario	Id_Cliente	Integer	4	X	X	Clientes, D_Credito,

						Pagos, Con_Cte_Mat, Factura, C_Pedido
Indicador de Impuesto	Id_IndImp	Smallint	2	X	X	Impuestos, Clientes
Nombre	Nombre	Char	35	X	X	Clientes
RFC	RFC	Char	13	X	X	Clientes
Calle y Número	Calleno	Char	40	X	X	Clientes
Colonia	Colonia	Char	35	X	X	Clientes
Población	Poblacion	Char	35	X	X	Clientes
Código Postal	CP	Char	5	X	X	Clientes
Teléfono	Telefono	Char	24	X		Clientes
Fax	Fax	Char	24	X		Clientes
Límite de Crédito ó Importe	Importe	Money	8	X		D_Credito, Condiciones, Factura, Con_Cte_Mat, Con_Material, P_Pedido, C_Pedido, Pagos, Cond_Generales
Bloqueo	Bloqueo	Char	1	X		Clientes, D_Credito, Sucursal, Productos, E_Calculo, T_Condicion
Identificador de Unidad de Medida	Id_UME	Smallint	2	X	X	U_Medida, Productos, Inventario, P_Pedido, Con_Material, Con_Cte_Mat, Entrega, Factura
Identificador de Centro de Suministro	Id_Centro	Smallint	2	X	X	Centro_Sum, C_Pedido, Inventario, Entrega
Identificador de Producto	Id_Producto	Integer	4	X	X	Productos, Inventario, P_Pedido, Con_Material, Con_Cte,Mat,

						Entrega, Factura
Cantidad del Producto o Cantidad Pedida	Cantidad	Float	8	X		Inventario, P_Pedido, Entrega, Factura
Identificador de Tipo deCondición	Id_TCondicion	Smallint	2	X	X	T_Condición, E_Calculo, Con_Material, Con_Cte_Mat, Condiciones, Cond_Generales
Signo	Signo	Char	1	X	X	T_Condicion Cond_Generales
Identificador de Esquema de Cálculo	Id_Esquema	Smallint	2	X	X	E_Calculo, T_Pedido
Identificador de Tipo de Pedido	Id_TPedido	Smallint	2	X	X	T_Pedido, C_Pedido
Identificador de Fecha o Fin de Validez	Id_Fecha	Datetime	8	X	X	Con_Material, Con_Cte_Mat
Inicio de Validez o Fecha de Ingreso o Fecha de Creación	Fecha	Datetime	8	X	X	Con_Material, Con_Cte_Mat, T_Cambio Empleados, C_Pedido
Identificador de Empleado	Id_Empleado	Smallint	2	X	X	Empleados, C_Pedido
Nombre Empleado	NomEmp	Char	20	X	X	Empleados
Apellido Paterno	ApellidoP	Char	20	X	X	Empleados
Apellido Materno	ApellidoM	Char	20	X		Empleados
Identificador de Pedido	Id_Pedido	Integer	4	X	X	C_Pedido, P_Pedido, Entrega, Factura
Importe del Tipo de Cambio	Importe_TC	Money	8	X	X	T_Cambio
Identificador de Posición	Id_Consecutivo	Smallint	2	X	X	P_Pedido, Entrega, Factura
Cantidad Surtida	Cant_Sur	Float	8	X		P_Pedido
Identificador de Entrega	Id_Entrega	Integer	4	X	X	Entrega, Factura
Identificador de Factura	Id_Factura	Integer	4	X	X	Factura
Identificador de	Id_Condicion	Integer	4	X	X	Condiciones,

Número de Condición						C_Pedido
Estatus de Pago de Factura (‘ ‘ = No pagada P = Parcial C = Completa)	Pagada	Char	1	X		Factura
Importe del Pago	Imppago	Money	8	X		Pagos
Porcentaje de Impuesto	PorImp	Numeric	3	X	X	Impuestos

Figura 2.4

v – Procedimientos para Entidad-Relación

A partir de los datos obtenidos anteriormente, podemos obtener los procedimientos que nos serviran para actualizar las tablas (Entidades y relaciones) que serán necesarios para este sistema, estos procedimientos fueron desarrollados en Transact SQL, los cuales aparecen en la figura 2.5

Nombre De la Tabla	Tipo (Entidad/ Relación)	Procedimientos Desarrollados
Pais	Entidad	Inserta País Actualiza País Consulta País
Estados	Relación	Inserta Estados Actualiza Estados Consulta Estados
IATA	Relación	Inserta claves IATA Actualiza claves IATA Consulta claves IATA
Monedas	Entidad	Inserta Monedas Actualiza Monedas Consulta Monedas
Sociedad	Entidad	Inserta Sociedades Actualiza Sociedades Elimina Sociedades Consulta Sociedades
Sucursal	Relación	Inserta Sucursales Actualiza Sucursales Elimina Sucursales Consulta Sucursales Desbloquear Sucursales
T_Cliente	Entidad	Inserta tipos de Clientes Consulta tipos de Clientes
C_Pago	Entidad	Inserta Condiciones de Pago Actualiza Condiciones de Pago Elimina Condiciones de Pago Consulta Condiciones de Pago
Impuestos	Entidad	Inserta Impuestos Actualiza Impuestos Elimina Impuestos Consulta Impuestos
Clientes	Relación	Inserta Clientes Actualiza Clientes Elimina Clientes Consulta Clientes Desbloquear Clientes

Nombre De la Tabla	Tipo (Entidad/ Relación)	Procedimientos Desarrollados
D_Credito	Relación	Inserta Datos de Crédito Actualiza Datos de Crédito Elimina Datos de Crédito Consulta Datos de Crédito
U_Medida	Entidad	Inserta Unidades de Medida Actualiza Unidades de Medida Elimina Unidades de Medida Consulta Unidades de Medida
Centro_Sum	Relación	Inserta Centros de Suministro Actualiza Centros de Suministro Elimina Centros de Suministro Consulta Centros de Suministro
Productos	Relación	Inserta Productos Actualiza Productos Elimina Productos Consulta Productos Desbloquear Productos
Inventario	Relación	Inserta Inventario Consulta Inventario
T_Condicion	Entidad	Inserta Tipos de Condición Actualiza Tipos de Condición Elimina Tipos de Condición Consulta Tipos de Condición Desbloquear Tipos de Condición
E_Calculo	Relación	Inserta Esquemas de Cálculo Elimina Esquemas de Cálculo Consulta Esquemas de Cálculo Desbloquear Esquemas de Cálculo
T_Pedido	Relación	Inserta Tipos de Pedido Consulta Tipos de Pedido
T_Cambio	Relación	Inserta Tipos de Cambio Actualiza Tipos de Cambio Consulta Tipos de Cambio
Con_Material	Relación	Inserta Condiciones por Material Actualiza Condiciones por Material Consulta Condiciones por Material
Con_Cte_Mat	Relación	Inserta Condiciones por Cliente Material Actualiza Condiciones por Cliente Material Consulta Condiciones por Cliente Material
Cond_Generales	Relación	Inserta Condiciones Generales Actualiza Condiciones Generales Elimina Condiciones Generales Consulta Condiciones Generales
Empleados	Entidad	Inserta Empleados Actualiza Empleados Elimina Empleados Consulta Empleados

Nombre De la Tabla	Tipo (Entidad/ Relación)	Procedimientos Desarrollados
Condiciones	Relación	Inserta Condiciones Consulta Condiciones
C_Pedido	Relación	Inserta Cabecera del Pedido Actualiza Cabecera del Pedido Elimina Cabecera del Pedido Consulta Cabecera del Pedido
P_Pedido	Relación	Inserta Posiciones del Pedido Actualiza Posiciones del Pedido Elimina Posiciones del Pedido Consulta Posiciones del Pedido
Entrega	Relación	Inserta Entrega (a partir del pedido) Actualiza Entrega (a partir del pedido) Elimina Entrega (a partir del pedido) Consulta Entrega (a partir del pedido)
Factura	Relación	Inserta Factura (a partir del Pedido) Actualiza Factura (a partir del Pedido) Elimina Factura (a partir del Pedido) Consulta Factura (a partir del Pedido)
Pagos	Relación	Inserta Pagos Actualiza Pagos Consulta Pagos

Figura 2.5 Procedimientos para entidades y relaciones

vi – Diagramas de Casos de Uso

A partir de la definición del proyecto, podemos obtener los diagramas de los casos de uso, los cuales aparecen en las figuras 2.6 a 2.9

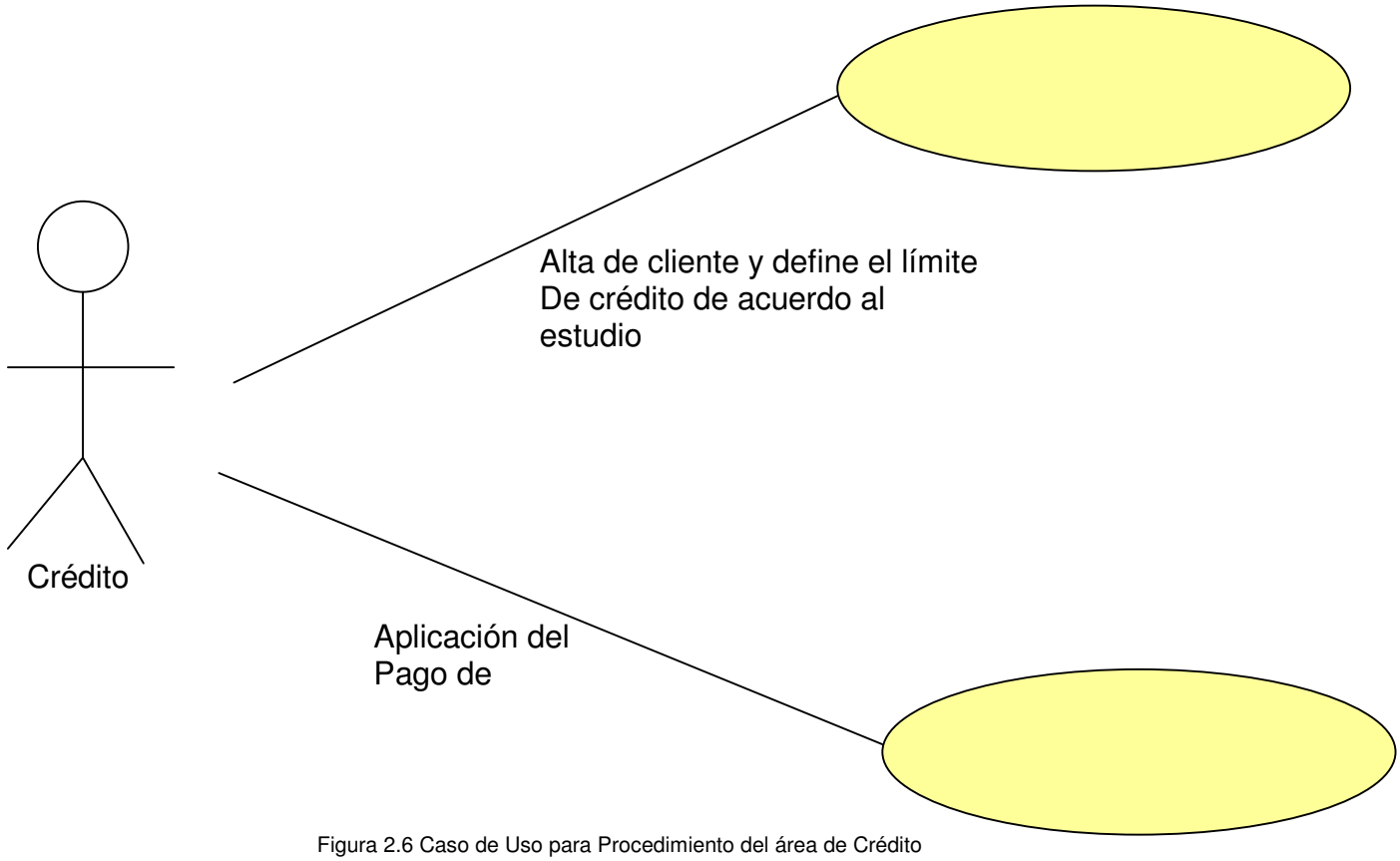


Figura 2.6 Caso de Uso para Procedimiento del área de Crédito

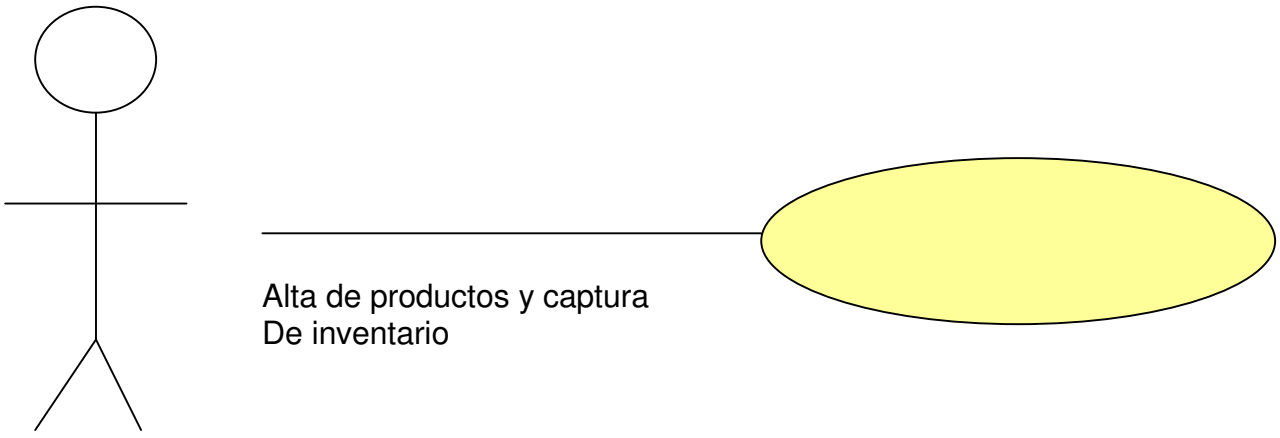


Figura 2.7 Caso de Uso para Procedimiento del área de Logistica

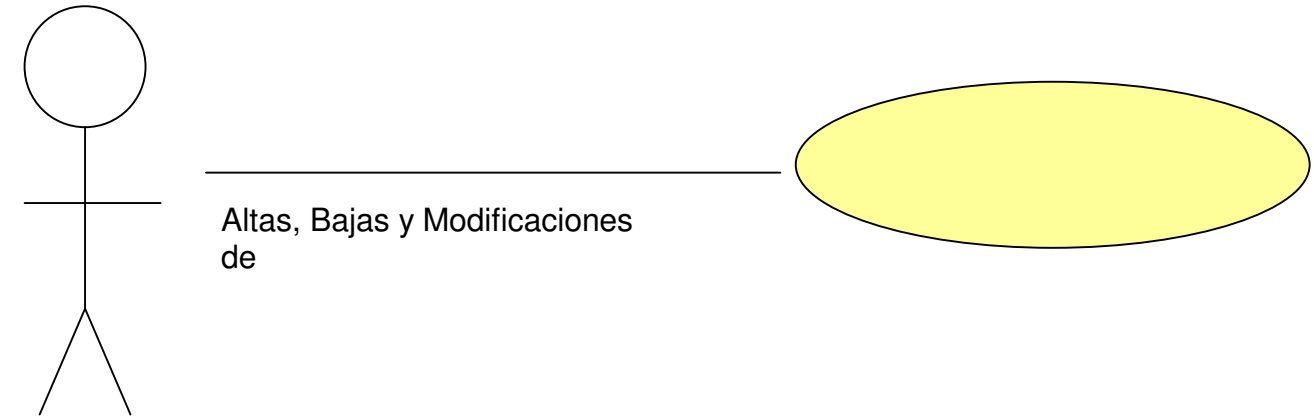


Figura 2.8 Caso de Uso para Procedimiento del área de Administración de ventas

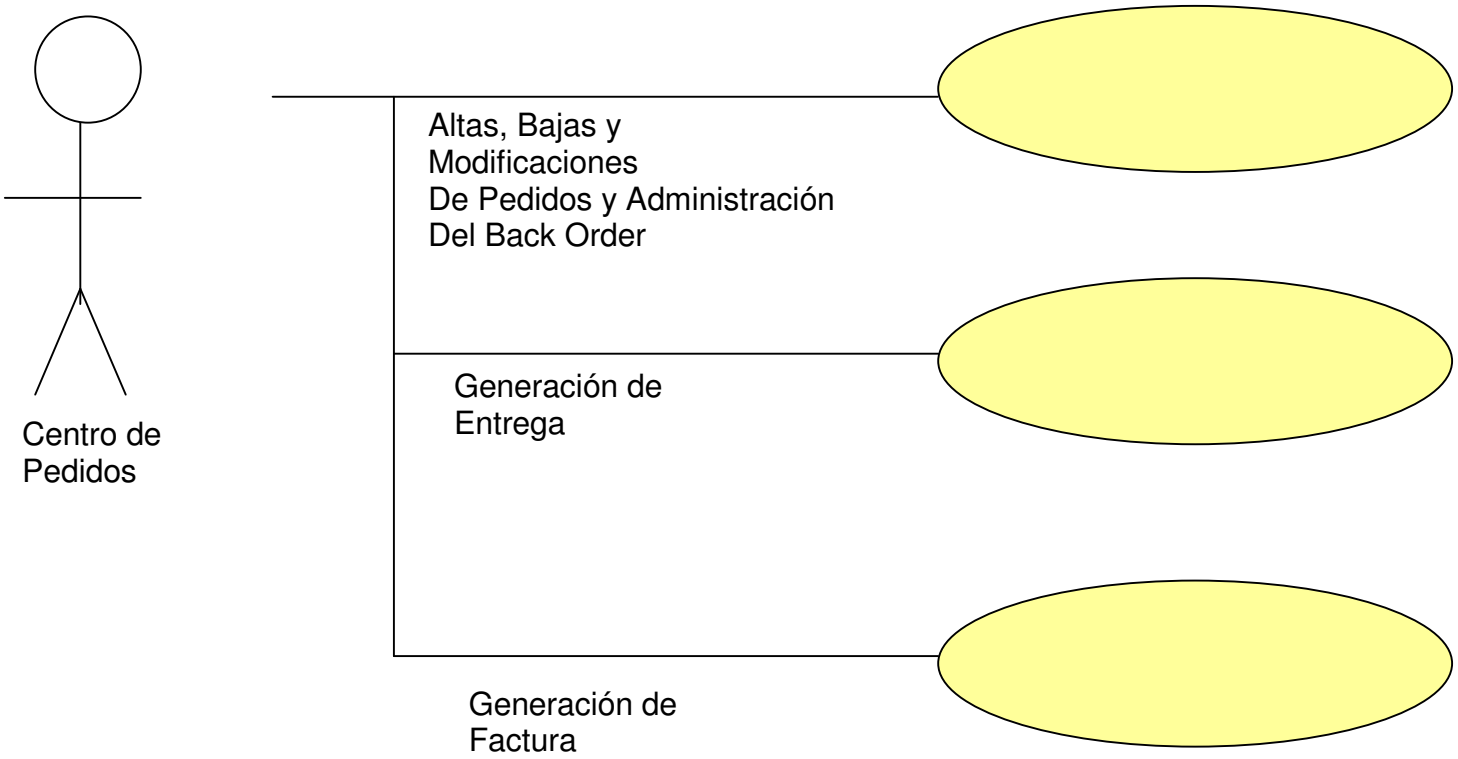


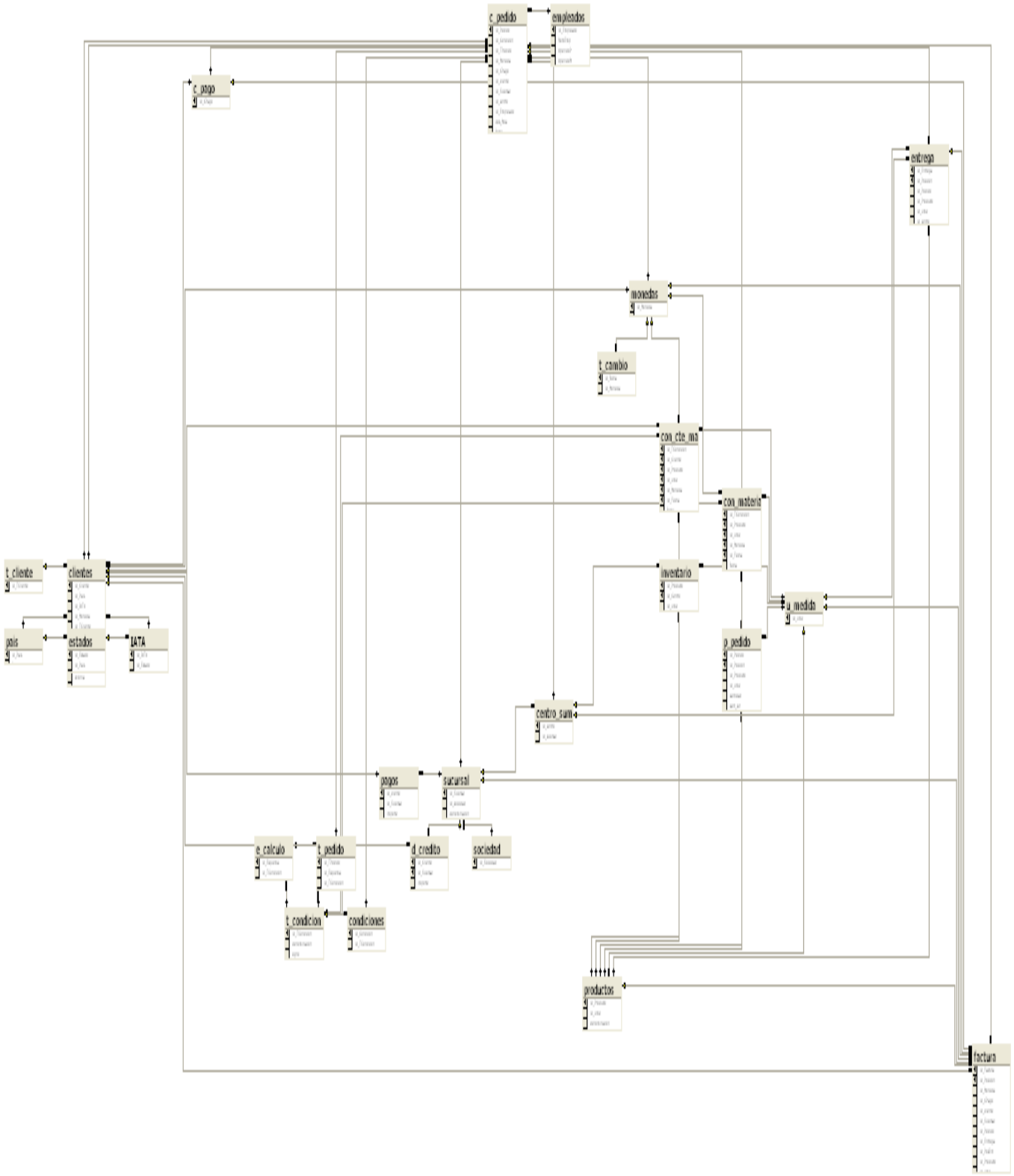
Figura 2.9 Caso de Uso para Procedimiento del área de Ventas

II - Modelado de la Base de Datos

Después de evaluar con qué manejador de base de datos realizaría este proyecto, me decidí por el de Microsoft® SQL Server por requerir menos recursos que el manejador de Base de Datos de Oracle y por lo tanto sería más sencilla su comercialización, ya que los requerimientos de hardware serían menores, aunque si bien es cierto que Oracle es más robusto, elegí SQL porque se trata de un sistema pequeño que funcionaría bien con el manejador de Base de Datos de Microsoft® SQL Server.

A continuación presentaré el modelado de la Base de Datos, el cual consta de las instrucciones en Microsoft® SQL Server para la generación de dicha base de datos y el diagrama de esta (figura 3.1).

Figura 3.1 Diagrama de la Base de Datos



Definición de la Base de Datos de Ejemplo que se utiliza

```
use master
go
```

Crea la Base de Datos

```
CREATE DATABASE AdminVentas ON PRIMARY
(
  NAME = AdminVentas,
        FILENAME = 'C:\AdminVentas.mdf',
        SIZE = 10,
        MAXSIZE = 100,
        FILEGROWTH = 10%
)
GO
```

Creación de Tablas

Se pone como ejemplo la creación de algunas de las tablas utilizadas

Crea tabla de Países

```
create table pais
(
  id_Pais      nvarchar(4)  not null primary key,
  Denominacion nvarchar(40) not null
)
go
```

Crea tabla de sucursales

```
create table sucursal
(
  id_Sucursal  smallint     not null primary key identity,
  id_sociedad  smallint     not null foreign key references sociedad (id_Sociedad),
```

```
 denominacion nvarchar(40) not null,
 bloqueo      nvarchar(1)  not null
)
go
```

Crea tabla de Clientes

```
create table clientes
(
 id_Cliente  int          not null primary key identity,
 id_Pais     nvarchar(4)  not null foreign key references pais(id_pais),
 id_IATA     nvarchar(3)  not null foreign key references IATA(id_IATA),
 id_Moneda   nvarchar(3)  not null foreign key references monedas(id_Moneda),
 id_TCliente smallint     not null foreign key references t_cliente(id_TCliente),
 id_CPago    smallint     not null foreign key references c_pago(id_CPago),
 id_IndImp   smallint     not null foreign key references impuestos(id_IndImp),
 nombre      nvarchar(35) not null,
 rfc         nvarchar(13) not null,
 calleno     nvarchar(40) not null,
 colonia     nvarchar(35) not null,
 poblacion   nvarchar(35) not null,
 cp          nvarchar(5)  not null,
 telefono    nvarchar(24) not null,
 fax         nvarchar(24) not null,
 bloqueo     nvarchar(1)  not null
)
go
```

Crea tabla de Productos

```
create table productos
(
 id_Producto  int          not null primary key identity,
```

```
 id_ume       smallint     not null foreign key references u_medida(id_ume),
 denominacion nvarchar(40) not null,
 bloqueo      nvarchar(1)  not null
)
go
```

Crea Tabla de Cabecera de Pedido

```
create table c_pedido
(
 id_Pedido    int         not null primary key identity,
 id_Condicion int         not null foreign key references condiciones(id_Condicion),
 id_TPedido   smallint    not null foreign key references t_pedido(id_TPedido),
 id_Moneda    nvarchar(3) not null foreign key references monedas(id_Moneda),
 id_CPago     smallint    not null foreign key references c_pago(id_CPago),
 id_cliente   int         not null foreign key references clientes(id_Cliente),
 id_Sucursal  smallint    not null foreign key references sucursal(id_Sucursal),
 id_centro    smallint    not null foreign key references centro_sum(id_centro),
 id_Empleado  smallint    not null foreign key references empleados(id_Empleado),
 des_Mcia     int         not null foreign key references clientes(id_Cliente),
 fecha        datetime    not null,
 importe      money       not null,
 importe_TC   money       not null
)
go
```

Crea tabla de Posiciones de Pedido

```
create table p_pedido
(
 id_Pedido   int      not null foreign key references c_pedido(id_Pedido),
 id_Posicion smallint not null,
 id_Producto int      not null foreign key references productos(id_Producto),
```

```
 id_ume      smallint not null foreign key references u_medida(id_ume),
 cantidad    float    not null,
 cant_sur    float    not null,
 importe     money    not null,
 constraint Ped_Pos PRIMARY KEY (id_Pedido, id_Posicion)
)
go
```

Crea Tabla de Cabecera de Entregas

```
Create table c_entrega
(
id_Entrega  int      not null identity primary key,
des_Mcia    int      not null foreign key references clientes(id_Cliente),
)
go
```

Crea Tabla de Entregas

```
create table entrega
(
 id_Entrega  int      not null foreign key references c_entrega(id_Entrega),
 id_Posicion smallint not null,
 id_Pedido   int      not null foreign key references c_pedido(id_Pedido),
 id_Producto int      not null foreign key references productos(id_Producto),
 id_ume      smallint not null foreign key references u_medida(id_ume),
 id_centro   smallint not null foreign key references centro_sum(id_centro),
 cantidad    float    not null,
 constraint Ent_Pos PRIMARY KEY (id_Entrega, id_Posicion)
)
go
```

Crea Tabla de Cabecera de Facturas

```
Create table c_factura
(
id_Factura  int      not null identity primary key,
id_cliente   int        not null foreign key references clientes(id_Cliente),
)
go
```

Crea Tabla de Facturas

```
create table factura
(
 id_Factura  int      not null  foreign key references c_factura(id_Factura),
 id_Posicion smallint not null,
 id_Moneda   nvarchar(3)  not null foreign key references monedas(id_Moneda),
 id_CPago    smallint not null foreign key references c_pago(id_CPago),
 id_cliente  int      not null foreign key references clientes(id_Cliente),
 id_Sucursal smallint not null foreign key references sucursal(id_Sucursal),
 id_Pedido   int      not null foreign key references c_pedido(id_Pedido),
 id_Entrega  int      not null,
 id_PosEnt   smallint not null,
 id_Producto int      not null foreign key references productos(id_Producto),
 id_ume      smallint not null foreign key references u_medida(id_ume),
 cantidad    float    not null,
 importe     money    not null,
 pagada      nvarchar(1) not null,
 constraint Fac_Pos PRIMARY KEY (id_Factura, id_Posicion),
 constraint FK_Ent_Pos foreign key (id_Entrega, id_PosEnt) references
entrega(id_Entrega, id_Posicion)
)
go
```

A continuación presentaré las instrucciones en Microsoft® SQL Server para la generación de los diferentes procedimientos que se definieron en la figura 2.5 que serían necesarios.

PROCEDIMIENTOS

A continuación incluiré parte del código que será necesario para la implementación del sistema de ventas.

Lo primero que necesitaríamos es llamar a la Base de Datos para que los programas no marquen errores, porque si no hacemos el llamado a la base de datos los programas no encontrarán las entidades y relaciones (tablas) que fueron creadas anteriormente y por lo tanto no será posible ejecutarlos, todo esta desarrollado sobre el usuario administrador.

```
use AdminVentas
go
```

Crea procedimiento para insertar registros en País

```
drop procedure sp_inserta_pais
go
create procedure sp_inserta_pais
@id_pais      nvarchar(4),
@denominacion nvarchar(40)
as
  declare @mensaje nvarchar(50)
  if exists (select * from pais where id_pais = @id_pais)
    begin
      set @mensaje = 'El País ya existe'
      print @mensaje
    end
  else
    begin
      insert into pais values (@id_pais, @denominacion)
       set @mensaje = 'País Insertado Exitosamente'
       print @mensaje
    end
go
```

Crea procedimiento para actualizar registros en País

```
drop procedure sp_actualiza_pais
go
create procedure sp_actualiza_pais
@id_pais      nvarchar(4),
```

```
@denominacion nvarchar(40)
as
  declare @mensaje nvarchar(50)
  if not exists (select * from pais where id_pais = @id_pais)
    begin
      set @mensaje = 'El País no existe'
      print @mensaje
    end
  else
    begin
      update pais set denominacion = @denominacion
      from pais where id_pais = @id_pais
      set @mensaje = 'País Actualizado Exitosamente'
      print @mensaje
    end
go
```

Crea procedimiento para insertar registros en Clientes

```
drop procedure sp_inserta_clientes
go
create procedure sp_inserta_clientes
@id_Pais     nvarchar(4),
@id_IATA     nvarchar(3),
@id_Moneda   nvarchar(3),
@id_TCliente smallint,
@id_CPago    smallint,
@nombre      nvarchar(35),
@rfc         nvarchar(13),
@calleno     nvarchar(40),
@colonia     nvarchar(35),
@poblacion   nvarchar(35),
@cp          nvarchar(5),
@telefono    nvarchar(24),
@fax         nvarchar(24)
as
  declare @mensaje nvarchar(50)
  declare @id_cliente int
  if exists (select * from clientes where rfc = @rfc)
    begin
      set @mensaje = 'El cliente ya existe'
      print @mensaje
    end
  else
    if not exists (select * from pais where id_pais = @id_pais)
      begin
        set @mensaje = 'País no existe'
```

```
      print @mensaje
    end
  else
    if not exists (select * from iata where id_iata = @id_iata)
      begin
        set @mensaje = 'Clave IATA no existe'
        print @mensaje
      end
    else
      if not exists (select * from monedas where id_moneda = @id_moneda)
        begin
          set @mensaje = 'Moneda no existe'
          print @mensaje
        end
      else
        if not exists (select * from t_cliente where id_tcliente = @id_tcliente)
          begin
            set @mensaje = 'Tipo de Cliente no existe'
            print @mensaje
          end
        else
          if not exists (select * from c_pago where id_cpago = @id_cpago)
            begin
              set @mensaje = 'Condicón de Pago no existe'
              print @mensaje
            end
          else
            begin
              insert into clientes values (@id_Pais, @id_IATA, @id_Moneda,
@id_TCliente, @id_CPago,
                    @nombre, @rfc, @calleno, @colonia, @poblacion, @cp,
@telefono, @fax, ' ')
              set @mensaje = 'Cliente Insertado Exitosamente'
              print @mensaje
              select @id_cliente = id_cliente from clientes where rfc = @rfc
              print @id_cliente
            end
go
```

Crea procedimiento para Desbloquear Clientes

```
drop procedure sp_desbloquear_clientes
go
create procedure sp_desbloquear_clientes
@id_cliente int
as
  declare @mensaje nvarchar(50)
```

```
  if exists (select * from clientes where id_cliente = @id_cliente
                                and bloqueo = 'X')
    begin
      update clientes set bloqueo = ' '
      from clientes where id_cliente = @id_cliente
      set @mensaje = 'Cliente Desbloqueado Exitosamente'
      print @mensaje
    end
  else
    begin
      set @mensaje = 'Cliente no Existe o no esta Bloqueado'
      print @mensaje
    end
go
```

Crea procedimiento para insertar Productos

```
drop procedure sp_inserta_productos
go
create procedure sp_inserta_productos
@id_ume       smallint,
@denominacion nvarchar(40)
as
  declare @mensaje  nvarchar(50)
  declare @id_producto int
  if exists (select * from productos where denominacion = @denominacion)
    begin
      set @mensaje = 'Producto ya Existe'
      print @mensaje
    end
  else
    begin
      if not exists (select * from u_medida where id_ume = @id_ume)
        begin
          set @mensaje = 'Unidad de Medida no Existe'
          print @mensaje
        end
      else
        begin
          insert into productos values (@id_ume, @denominacion, ' ')
          set @mensaje = 'Producto insetado Exitosamente'
          print @mensaje
          select @id_producto = id_producto from productos where denominacion =
@denominacion
          print @id_producto
        end
```

```
   end
go
```

Crea procedimiento para insertar Inventario

```
drop procedure sp_inserta_inventario
go
create procedure sp_inserta_inventario
@id_producto int,
@id_centro   smallint,
@id_ume      smallint,
@cantidad    float
as
  declare @mensaje  nvarchar(50)
  if not exists (select * from productos where id_producto = @id_producto)
    begin
      set @mensaje = 'Producto No Existe'
      print @mensaje
    end
  else
    begin
      if not exists (select * from u_medida where id_ume = @id_ume)
        begin
          set @mensaje = 'Unidad de Medida no Existe'
          print @mensaje
        end
      else
        if not exists (select * from centro_sum where id_centro = @id_centro)
          begin
            set @mensaje = 'Centro no Existe'
            print @mensaje
          end
        else
          if @cantidad <= 0
            begin
              set @mensaje = 'Cantida debe ser mayor que cero'
              print @mensaje
            end
          else
            begin
              insert into inventario values (@id_producto, @id_centro, @id_ume,
@cantidad)
              set @mensaje = 'Inventario insetado Exitosamente'
              print @mensaje
            end
   end
go
```

Crea procedimiento para insertar Tipo de Pedido

```
drop procedure sp_inserta_t_pedido
go
create procedure sp_inserta_t_pedido
@id_esquema    nvarchar(5),
@denominacion  nvarchar(40)
as
  declare @mensaje    nvarchar(50)
  declare @id_tpedido smallint
  if exists (select * from t_pedido where denominacion = @denominacion)
    begin
      set @mensaje = 'Tipo de Pedido ya Existe'
      print @mensaje
    end
  else
    if not exists (select * from e_calculo where id_esquema = @id_esquema)
      begin
        set @mensaje = 'Esquema de Cálculo no Existe'
        print @mensaje
      end
    else
      begin
        insert into t_pedido values (@id_esquema, @denominacion)
        set @mensaje = 'Tipo de Pedido insetado Exitosamente'
        print @mensaje
        select @id_tpedido = id_tpedido from t_pedido where id_esquema =
@id_esquema
                                        and denominacion = @denominacion
        print @id_tpedido
      end
go
```

Crea procedimiento para insertar Tipo de Cambio

```
drop procedure sp_inserta_t_cambio
go
create procedure sp_inserta_t_cambio
@id_fecha  datetime,
@id_Moneda nvarchar(3),
@importe   money
as
  declare @mensaje nvarchar(50)
  if exists (select * from t_cambio where id_fecha = @id_fecha)
    begin
      set @mensaje = 'Ya Existe Tipo de Cambio para esta Fecha'
      print @mensaje
```

```
    end
  else
    if not exists (select * from monedas where id_moneda = @id_moneda)
      begin
        set @mensaje = 'Moneda no Existe'
        print @mensaje
      end
    else
      if @importe <= 0
        begin
          set @mensaje = 'Importe debe ser Mayor que Cero'
          print @mensaje
        end
      else
        begin
          insert into t_cambio values (@id_fecha, @id_moneda, @importe)
          set @mensaje = 'Tipo de Cambio insetado Exitosamente'
          print @mensaje
        end
go
```

Crea procedimiento para insertar Condición por Material

```
drop procedure sp_inserta_con_mat
go
create procedure sp_inserta_con_mat
@id_tcondicion smallint,
@id_producto   int,
@id_ume        smallint,
@id_moneda     nvarchar(3),
@id_fecha      datetime,
@fecha         datetime,
@importe       money
as
  declare @mensaje nvarchar(50)
  if not exists (select * from t_condicion where id_tcondicion = @id_tcondicion)
    begin
      set @mensaje = 'No Existe Tipo de Condición'
      print @mensaje
    end
  else
    if not exists (select * from productos where id_producto = @id_producto)
      begin
        set @mensaje = 'Producto no Existe'
        print @mensaje
      end
    else
```

```
    if not exists (select * from u_medida where id_ume = @id_ume)
      begin
        set @mensaje = 'Unidad de Medida no Existe'
        print @mensaje
      end
    else
      if not exists (select * from monedas where id_moneda = @id_moneda)
        begin
          set @mensaje = 'Moneda no Existe'
          print @mensaje
        end
      else
        if exists (select * from con_material where id_tcondicion = @id_tcondicion
                                    and id_producto = @id_producto
                                    and id_ume = @id_ume
                                    and id_moneda = @id_moneda
                                    and id_fecha = @id_fecha)
          begin
            set @mensaje = 'Condición ya Existe'
            print @mensaje
          end
        else
          if @importe <= 0
            begin
              set @mensaje = 'Importe debe ser Mayor que Cero'
              print @mensaje
            end
          else
            begin
              insert into con_material values (@id_tcondicion, @id_producto,
@id_ume, @id_moneda, @id_fecha, @fecha, @importe)
              set @mensaje = 'Condición por Material insetado Exitosamente'
              print @mensaje
            end
go
```

Crea procedimiento para insertar Empleados

```
drop procedure sp_inserta_empleado
go
create procedure sp_inserta_empleado
@nomEmp    nvarchar(20),
@ApellidoP nvarchar(20),
@ApellidoM nvarchar(20)
as
  declare @mensaje     nvarchar(50)
  declare @id_empleado smallint
```

```
 if exists (select * from empleados where nomemp = @nomemp
                              and apellidop = @apellidop
                              and apellidom = @apellidom)
   begin
     set @mensaje = 'Este Empleado ya Existe'
     print @mensaje
   end
 else
   begin
     insert into empleados values (@nomemp, @apellidop, @apellidom, getdate())
     set @mensaje = 'Empleado insetado Exitosamente'
     print @mensaje
     select @id_empleado = id_empleado from empleados where nomemp =
@nomemp
                                        and apellidop = @apellidop
                                        and apellidom = @apellidom
     print @id_empleado
   end
go
```

Crea procedimiento para insertar Cabecera de Pedido

```
drop procedure sp_inserta_cpedido
go
create procedure sp_inserta_cpedido
@id_tpedido  smallint,
@id_cliente  int,
@id_sucursal smallint,
@id_centro   smallint,
@id_empleado smallint,
@des_mcia    int
as
 declare @mensaje       nvarchar(50)
 declare @id_moneda     nvarchar(3)
 declare @id_cpago      smallint
 declare @importe_tc    money
 declare @importe       money
 declare @id_esquema    nvarchar(5)
 declare @id_tcondicion smallint
 declare @id_condicion  int
 declare @id_pedido     int
 if not exists (select * from clientes where id_cliente = @id_cliente)
   begin
     set @mensaje = 'Cliente no Existe'
     print @mensaje
   end
 else
```

```
   if not exists (select * from sucursal where id_sucursal = @id_sucursal)
     begin
       set @mensaje = 'Sucursal no Existe'
       print @mensaje
     end
   else
     if exists (select * from d_credito where id_cliente = @id_cliente
                                  and id_sucursal = @id_sucursal
                                  and bloqueo = 'X' )
       begin
         set @mensaje = 'Cliente Bloqueado por Crédito'
         print @mensaje
       end
     else
       if not exists (select * from centro_sum where id_centro = @id_centro)
         begin
           set @mensaje = 'Centro de Suministro no Existe'
           print @mensaje
         end
       else
         if not exists (select * from empleados where id_empleado = @id_empleado)
           begin
             set @mensaje = 'Empleado no Existe'
             print @mensaje
           end
         else
           if not exists (select * from clientes where id_cliente = @des_mcia)
             begin
               set @mensaje = 'Destinatario de Mercancía no Existe'
               print @mensaje
             end
           else
             begin
               select @id_moneda = id_moneda, @id_cpago = id_cpago from clientes
                     where id_cliente = @id_cliente
               set @importe_tc = 1
               select @importe_tc = importe from t_cambio where id_moneda =
@id_moneda
                     order by id_fecha
               select @id_esquema = id_esquema from t_pedido where id_tpedido =
@id_tpedido
               select @id_tcondicion = id_tcondicion from e_calculo where id_esquema
= @id_esquema
                                                     and bloqueo = ' '
               insert into condiciones values (@id_tcondicion,99999999)
               select @id_condicion = id_condicion from condiciones
                     where id_tcondicion = @id_tcondicion
```

```
                and importe = 99999999
            insert into c_pedido values (@id_condicion, @id_TPedido,
@id_Moneda, @id_CPago, @id_cliente, @id_Sucursal, @id_centro,
@id_Empleado, @des_mcia, getdate(), 0, @importe_tc)
            select @id_pedido = id_pedido from c_pedido where id_condicion =
@id_condicion
                                        and id_TPedido = @id_TPedido
                                        and id_Moneda = @id_Moneda
                                        and id_CPago = @id_CPago
                                        and id_cliente = @id_cliente
                                        and id_Sucursal = @id_Sucursal
                                        and id_centro = @id_centro
                                        and id_Empleado = @id_Empleado
                                        and des_mcia = @des_mcia
                                        and importe = 0
                                        and importe_tc = @importe_tc
            print @id_pedido
          end
 go
```

Crea procedimiento para insertar Posición de Pedido

```
drop procedure sp_inserta_ppedido
go
create procedure sp_inserta_ppedido
@id_pedido   int,
@id_producto int,
@cantidad    float
as
 declare @mensaje       nvarchar(50)
 declare @id_ume        smallint
 declare @importe       money
 declare @importe1      money
 declare @id_cliente    int
 declare @id_condicion  int
 declare @id_moneda     nvarchar(3)
 declare @id_tcondicion smallint
 declare @id_esquema    nvarchar(5)
 declare @id_tpedido    smallint
 declare @id_indimp     smallint
 declare @id_posicion   smallint
 declare @imp_credito   money
 declare @id_Sucursal   smallint
 declare @imp_deuda     money
 declare @imp_pago      money
 declare @id_centro     smallint
 declare @cant_inv      float
```

```
 if not exists (select * from c_pedido where id_pedido = @id_pedido)
  begin
   set @mensaje = 'Cabecera de Pedido no Existe'
   print @mensaje
  end
 else
  if not exists (select * from productos where id_producto = @id_producto)
   begin
    set @mensaje = 'Producto no Existe'
    print @mensaje
   end
  else
   if @cantidad <= 0
    begin
     set @mensaje = 'Cantidad debe ser mayor que cero'
     print @mensaje
    end
   else
    select @id_ume = id_ume from productos where id_producto = @id_producto
    select @id_condicion = id_condicion, @id_cliente = id_cliente, @id_moneda
= id_moneda,
          @id_tpedido = id_tpedido, @id_sucursal = id_sucursal, @id_centro =
id_centro
          from c_pedido where id_pedido = @id_pedido
    set @importe = 0
    set @importe1 = 0
    set @imp_credito = 0
    set @imp_deuda = 0
    set @imp_pago = 0
    set @id_posicion = 0
    select @imp_credito = importe from d_credito where id_cliente = @id_cliente
                                    and id_sucursal = @id_sucursal
    select @imp_pago = impago, @imp_deuda = importe from pagos where
id_cliente = @id_cliente
                                          and id_sucursal = @id_sucursal
    set @imp_credito = @imp_credito - ( @imp_deuda - @imp_pago )
    select @id_esquema = id_esquema from t_pedido where id_tpedido =
@id_tpedido
    select @id_tcondicion = id_tcondicion from e_calculo where id_esquema =
@id_esquema
    if exists (select * from con_cte_mat where id_tcondicion = @id_tcondicion
                                  and id_cliente = @id_cliente
                                  and id_producto = @id_producto
                                  and id_ume = @id_ume
                                  and id_moneda = @id_moneda
                                  and id_fecha <= getdate())
```

```
        begin
          select @importe1 = importe from con_cte_mat where id_tcondicion =
@id_tcondicion
                                        and id_cliente = @id_cliente
                                        and id_producto = @id_producto
                                        and id_ume = @id_ume
                                        and id_moneda = @id_moneda
                                        and id_fecha <= getdate()
          set @importe = @importe + @importe1
          set @importe = @importe * @cantidad
          select @importe1 = importe from c_pedido where id_pedido = @id_pedido
          set @importe1 = @importe1 + @importe
          select @id_posicion = id_posicion from p_pedido where id_pedido =
@id_pedido
          set @id_posicion = @id_posicion + 1
          insert into p_pedido values (@id_pedido, @id_posicion, @id_producto,
@id_ume, @cantidad, 0, @importe)
          update c_pedido set importe = @importe1 from c_pedido where id_pedido
= @id_pedido
          update condiciones set importe = @importe1 from condiciones where
id_condicion = @id_condicion
          if @imp_credito > @importe1
            begin
              set @imp_deuda = @imp_deuda + @importe1
              if exists (select * from pagos where id_cliente = @id_cliente
                                    and id_sucursal = @id_sucursal)
                begin
                  update pagos set importe = @imp_deuda from pagos where id_cliente
= @id_cliente
                                                    and id_sucursal = @id_sucursal
                end
              else
                begin
                  insert into pagos values (@id_cliente, @id_sucursal, @imp_deuda, 0)
                end
              select @cant_inv = cantidad from inventario where id_producto =
@id_producto
                                            and id_centro = @id_centro
                                            and id_ume = @id_ume
            end
            if @cant_inv > @cantidad
              begin
                insert into entrega values (@id_posicion, @id_pedido, @id_producto,
@id_ume, @id_centro, @cantidad)
                set @cant_inv = @cant_inv - @cantidad
                update inventario set cantidad = @cant_inv from inventario where
id_producto = @id_producto
```

```
                                                    and id_centro = @id_centro
                                                    and id_ume = @id_ume
            end
           else
            if @cant_inv > 0
              begin
                insert into entrega values (@id_posicion, @id_pedido, @id_producto,
@id_ume, @id_centro, @cant_inv)
                set @cant_inv = 0
                update inventario set cantidad = @cant_inv from inventario where
id_producto = @id_producto
                                                      and id_centro = @id_centro
                                                      and id_ume = @id_ume
              end
           end
       else
        if exists (select * from con_material where id_tcondicion = @id_tcondicion
                                    and id_producto = @id_producto
                                    and id_ume = @id_ume
                                    and id_moneda = @id_moneda
                                    and id_fecha <= getdate())
          begin
            select @importe1 = importe from con_material where id_tcondicion =
@id_tcondicion
                                        and id_producto = @id_producto
                                        and id_ume = @id_ume
                                        and id_moneda = @id_moneda
                                        and id_fecha <= getdate()
           set @importe = @importe + @importe1
           set @importe = @importe * @cantidad
           select @importe1 = importe from c_pedido where id_pedido =
@id_pedido
           set @importe1 = @importe1 + @importe
           select @id_posicion = id_posicion from p_pedido where id_pedido =
@id_pedido
           set @id_posicion = @id_posicion + 1
           insert into p_pedido values (@id_pedido, @id_posicion, @id_producto,
@id_ume, @cantidad, 0, @importe)
           update c_pedido set importe = @importe1 from c_pedido where id_pedido
= @id_pedido
           update condiciones set importe = @importe1 from condiciones where
id_condicion = @id_condicion
           if @imp_credito > @importe1
             begin
               set @imp_deuda = @imp_deuda + @importe1
               if exists (select * from pagos where id_cliente = @id_cliente
                                    and id_sucursal = @id_sucursal)
```

```
                begin
                  update pagos set importe = @imp_deuda from pagos where
id_cliente = @id_cliente
                                                    and id_sucursal = @id_sucursal
                end
               else
                begin
                  insert into pagos values (@id_cliente, @id_sucursal, @imp_deuda,
0)
                end
               select @cant_inv = cantidad from inventario where id_producto =
@id_producto
                                              and id_centro = @id_centro
                                              and id_ume = @id_ume
             end
             if @cant_inv > @cantidad
              begin
                insert into entrega values (@id_posicion, @id_pedido, @id_producto,
@id_ume, @id_centro, @cantidad)
                set @cant_inv = @cant_inv - @cantidad
                update inventario set cantidad = @cant_inv from inventario where
id_producto = @id_producto
                                                    and id_centro = @id_centro
                                                    and id_ume = @id_ume
              end
             else
              if @cant_inv > 0
                begin
                  insert into entrega values (@id_posicion, @id_pedido,
@id_producto, @id_ume, @id_centro, @cant_inv)
                  set @cant_inv = 0
                  update inventario set cantidad = @cant_inv from inventario where
id_producto = @id_producto
                                                    and id_centro = @id_centro
                                                    and id_ume = @id_ume
                end
          end
         else
          begin
            set @mensaje = 'No Existe Precio para el producto'
            print @mensaje
          end
go
```

CONCLUSIONES

Podemos concluir que es necesario conocer varios de los modelos de Bases de Datos disponibles para que podamos analizarlos y hacer una buena selección de cuál es el modelo de Bases de Datos que mejor se adapta a nuestras necesidades y esto nos permitirá reducir el tiempo de análisis, desarrollo e implementación del proyecto.

Aunque es cierto que con el análisis y diseño de la Base de Datos tenemos una gran parte del proyecto resuelto, un 30% depende de entender y aplicar de manera correcta las Reglas del Negocio, pues si estas no están lo suficientemente claras o bien definidas tendremos problemas, pues será necesario redefinir nuestra Base de Datos cada vez que se tenga que aplicar una corrección en las Reglas del Negocio, por lo tanto, es primordial sentarse con el cliente y definir de manera clara y precisa todas las Reglas del Negocio que serán implementadas en nuestro sistema. Para esto es necesario que estén involucradas personas de todas las áreas que estarán presentes en nuestro sistema, pues son ellos los que conocen mejor que nadie las necesidades de su área y los lineamientos a seguir para cada proceso que vaya a ser automatizado, ellos le darán al diseñador de la Base de Datos todos los elementos que son necesarios considerar para realizar de manera correcta el diseño de la Base de Datos y ellos serán los responsables de hacer las pruebas al sistema antes de su implementación.

Con la utilización de un buen análisis y diseño de la Base de Datos tenemos aproximadamente el 50% del sistema elaborado, pues una vez que se organiza y diseña de manera correcta la Base de Datos de acuerdo a las Reglas del Negocio, sólo resta definir en que lenguaje de programación se realizará la interfase de usuario.

En este caso utilicé SQL Server porque requiere menos recursos y este es un sistema pequeño, pero en el análisis se determina que manejador de Base de Datos se debe utilizar dependiendo de los recursos tanto económicos como de hardware con los que se cuenta.

Una vez que se cuenta con las definiciones de las Reglas del Negocio, el modelo de Base de Datos a utilizar y el diseño de esta, se puede trabajar sin necesidad de estar redefiniendo el proyecto o nuestras tablas, pues estas quedaron definidas desde el diseño de la Base de Datos, lo cual ayuda a que se trabaje de manera eficiente y rápida desde el principio y tenemos un ahorro en tiempo de desarrollo y esto se refleja en el costo del proyecto, el cual será menor.

Debido a lo expuesto anteriormente, el diseño de mi proyecto deberá ser modificado para versiones posteriores, puesto que es muy general y basado únicamente en las necesidades básicas que consideré debe tener un pequeño sistema de ventas, sin embargo, una vez que lo elabore y empiece con la pruebas con algunas micro empresas obtendré una visión más amplia de las necesidades de los clientes y de esta manera podré implementar las mejoras necesarias para las siguientes versiones.

Sin embargo debido al conocimiento adquirido en el diseño de Base de Datos considero que tengo un buen avance y que las modificaciones no me llevarán a redefinir toda la Base de Datos, seguramente tendré que agregar algunas tablas y validaciones, pero en general lo que tengo como base me va a seguir funcionando, a pesar de las modificaciones que vaya sufriendo el diseño.

Por ello es que es fundamental el conocimiento y determinación del esquema que se desea utilizar, así como el alcance del proyecto porque de lo contrario sería empezar de cero y volver a diseñar todo, lo cual sería una perdida de tiempo y recursos.

BIBLIOGRAFÍA

Stallings,William
Sistemas Operativos
Ed. Prentice Hall

Piattini Mario, Adoración de Miguel, Marcos Esperanza.
DISEÑO DE BASES DE DATOS RELACIONALES.
Ed. Alfaomega

Jackson, G.A.
Introducción al diseño de Bases de Datos Relacionales.
Ed. Anaya

Yourdon, E.
Análisis Estructurado Moderno
Ed. Pretince Hall

Martín, James
Análisis y Diseño Orientado a Objetos
Ed. Pretince Hall

Thomas M. Connolly, Carolyn E. Begg
Sistemas de Bases de Datos: Un enfoque práctico para diseño, implementación y gestión
Ed. Pearson Addison

Silberschatz, Korth, Sudarshan
Fundamentos de Bases de Datos
Ed. McGraw Hill

Rafael Barzanallana
Apuntes. Sistemas de Bases de Datos.
Universidad de Murcia.

http://www.ur.mx/ur/faciya/carreras/cursos/sis/mod-dat1/graph.HTM
www.yudy.8m.com/Sistemasmanejador.htm
berzal.freeservers.com/freeware/dbms/spanish.html
http://www.lafacu.com/apuntes/informatica/base_datos/default.htm#Introducción
http://www.dbinternet.com.ar/metodo.htm
http://www.uas.mx/cursoswebct/Progsist/material.htm
http://www.programacionfacil.com/basic/cuatro4.htm
http://www.yudy.8m.com/Sistemasmanejador.htm
http://elizabethpeguero.8m.com/Eliza.htm
http://arraquis.dif.um.es/~rafa/bd1.htm
http://es.wikipedia.org