UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES PLANTEL CUAUTITLAN

REDES DE COMPUTADORAS

IMPLEMENTACIÓN DE UNA BASE DE DATOS RELACIONAL EN UNA RED DE ÁREA LOCAL:
MODELO CLIENTE - SERVIDOR

T R A B A J O D E S E M I N A R I O
QUE PARA OBTENER EL TITULO DE:
LICENCIADO EN INFORMÁTICA
P R E S E N T A :

CARLOS GUTIÉRREZ FLORES

ASESOR: ING. MOISÉS HERNÁNDEZ DUARTE

CUAUTITLAN IZCALLI ESTADO DE MÉXICO 1997

VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN
UNIDAD DE LA ADMINISTRACION ESCOLAR
DEPARTAMENTO DE EXAMENES PROFESIONALES

U. N. A. M.
FACULTAD DE ESTUDIOS
SUPERIORES-CUAUTITLAN

DEPARTAMENTO DE
EXAMENES PROFESIONALES

DR. JAIME KELLER TORRES
DIRECTOR DE LA FES-CUAUTITLAN
P R E S E N T E.

AT'N: ING. RAFAEL RODRIGUEZ CEBALLOS
Jefe del Departamento de Exámenes
Profesionales de la FES-C.

Con base en el art. 51 del Reglamento de Exámenes Profesionales de la FES-Cuautitlán, nos permitimos comunicar a usted que revisamos el Trabajo de Seminario:

Redes de Computadoras. Implementación de una Base de Datos Relacional en una Red de Area Loca: Modelo Cliente - Servidor

que presenta el pasante: Carlos Gutiérrez Flores.

con número de cuenta: 9010732-2 **para obtener el Título de:**

Licenciado en Informática.

Considerando que dicho trabajo reúne los requisitos necesarios para ser discutido en el EXAMEN PROFESIONAL correspondiente, otorgamos nuestro VISTO BUENO.

A T E N T A M E N T E.
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Edo. de México, a 15 **de** Octubre **de 19** 97.

MODULO:	PROFESOR:	FIRMA:
Modulo 3.	Ing. Moisés Hernández Duarte	
Modulo 3.	M. en I. Gloria Ponce Venegas	
Modulo 4.	Ing. Francisco Chavez Castañeda	

DEP/VOBOSEM

AGRADECIMIENTOS

A DIOS

Por guiarme por un buen camino.
Por darme la oportunidad de prepararme en la vida.
Por todo.
Gracias.

A MIS PADRES

CARLOS E ISABEL
Por darme la vida.
Por su amor.
Por su comprensión.
Por todo el apoyo brindado.
Porque sin escatimar esfuerzo alguno han sacrificado gran parte de su vida en formarme y educarme.
Gracias los adoro.

A MIS HERMANOS

MARCO ANTONIO Y KARLA NOEMI
Por su apoyo.
Por los grandes momentos.
Por su comprensión.
Por se parte de mi vida.

A MIS ABUELOS

Por darme unos padres excelentes e inculcar en ellos que lo primero es la familia.
Por su apoyo.
Por sus experiencias compartidas.
Por su amor.

A MIS TIOS

Por inspirarme confianza.
Por su apoyo.
Por el interes en mi camino por la vida.

A MIS PRIMOS

Por su apoyo.
Por los momentos especiales en familia.

A MI NOVIA

Por su amor.
Por su apoyo.
Por su comprensión y compañia.

A MI ASESOR

ING. MOISÈS HERNÁNDEZ DUARTE
Por el apoyo brindado para la
realización de este trabajo.
Por sus conocimientos.
Gracias.

A UNA GRAN PERSONA

LIC. ROSA VALADEZ OLGUIN
Por las facilidades para poder realizar
este trabajo.
Por su apoyo.
Por sus consejos.
Gracias Rosy.

ÍNDICE

INTRODUCCIÓN ____ *1*

OBJETIVOS ____ *3*

HIPÓTESIS ____ *4*

CAPITULO 1. BASES DE DATOS ____ *5*

1.1 CONCEPTO DE BASES DE DATOS ____ **5**

1.2 IMPORTANCIA DE LAS BASES DE DATOS ____ **6**

1.3 SISTEMA DE GESTIÓN DE BASES DE DATOS ____ **8**

1.4 CARACTERÍSTICAS DE UN SISTEMA DE GESTIÓN DE BASES DE DATOS ____ **8**

1.5 FUNCIONES PRINCIPALES DEL SISTEMA DE GESTIÓN DE BASES DE DATOS ____ **12**

CAPITULO 2. MODELO CLIENTE - SERVIDOR ____ *15*

2.1 CONCEPTO DEL MODELO CLIENTE - SERVIDOR ____ **15**

2.2 BENEFICIOS DEL MODELO CLIENTE - SERVIDOR ____ **20**

2.3 MODELO CLIENTE - SERVIDOR EN DESARROLLO DE SISTEMAS. ____ **22**

CAPITULO 3. IMPLEMENTACIÓN DE BASES DE DATOS RELACIONALES ____ *27*

3.1 METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS ____ **27**

3.2 MODELO RELACIONAL ____ **40**

3.3 IMPLEMENTACIÓN DE BASES DE DATOS RELACIÓNALES EN APLICACIONES CLIENTE - SERVIDOR ____ **46**

CAPITULO 4. LA IMPLEMENTACIÓN DE UNA BASE DE DATOS RELACIONAL EN EL SISTEMA DE OPERACIONES INTERNACIONALES DEL GRUPO FINANCIERO INBURSA ____ *51*

4.1 VISUAL BASIC ____ **51**

4.2 SQL SERVER 52
4.3 GRUPO FINANCIERO INBURSA 53
4.4 SOI 54
4.5 REQUERIMIENTOS PARA EL DESARROLLO 60
4.6 TÓPICOS REFERENTES AL EQUIPO DE DESARROLLO 61
4.7 ORGANIZACIÓN Y ARQUITECTURA DEL PROYECTO 62
4.8 IMPLEMENTACIÓN 63

CONCLUSIONES *79*

BIBLIOGRAFÍA *81*

ANEXOS *83*
TABLAS DE LA BASE DE DATOS 84

INTRODUCCIÓN

Uno de los recursos más importantes para las organizaciones en nuestros dias es la información. En la actualidad en el establecimiento de la economia global, los ejecutivos se dan cuenta que la información es un recurso esencial. Los datos (y la información derivada a partir de ellos) se necesitan para establecer iniciativas competitivas tales como: aumentar la satisfacción de los clientes, facilitar el trabajo de equipo, desarrollar nuevos productos y mercados, y dar respuesta rápida a las diferentes demandas de los clientes. por supuesto, los datos pueden ser útiles solamente si son confiables y se puede disponer de ellos cuando son necesarios. Esta es la razón por la cual, la mayoria de las organizaciones se preocupan por organizar y manejar datos de forma eficiente. Las bases de datos se han convertido en una técnica estándar para estructurar y manejar datos en las organizaciones hoy en dia.

En estos dias la información es algo muy importante para el desarrollo de la humanidad; para obtener un desarrollo es necesario tener información concisa, precisa, actualizada, entre otras caracteristicas. Para llegar a tener una información de este tipo las Bases de Datos hoy en dia son una herramienta poderosa . El buen manejo de los datos se utiliza en todos las áreas en las cuales se aplican sistemas computarizados. Una Base de Datos es aquella en la cual se registran datos (información con significado) los cuales tienen una relación entre ellos y la cual nos ayuda a tener una integridad de los mismos.

Durante décadas y todavia hoy en dia muchas empresas manejan su información a través de grandes aplicaciones monoliticas, donde todo el manejo de la información se encuentra en la lógica de un programa principal, el cual dicta como el usuario debe operar y suministrar su información. Además, cada programa dicta la forma en que el usuario debe navegar entre las pantallas.

Con la aplicación monolitica, cada aplicación contiene una única interface con el usuario diseñada para accesar a la Base de Datos o el Servidor. Esta solución es apropiada cuando los sistemas de información manejan limitados requerimientos, y donde el desarrollo de los sistemas depende de la tecnología disponible en ese momento. A lo anterior se conoce como el Modelo Servidor de Archivos. Pero en la actualidad el 85 % de las organizaciones están moviendo sus sistemas hacia el Modelo Cliente - Servidor para dar solución a sus problemas de antaño permitiéndoles tomar ventaja sobre la competencia, contar con sistemas tanto confiables como flexibles, que van desde una oficina integrada por tres personas hasta corporativos que manejan transacciones dentro de un pais, continentes o todo el mundo ya sea con aplicaciones de proceso de transacción en linea, sin la necesidad de que el usuario final conozca de donde proceden los Datos. Otra ventaja del Modelo Cliente - Servidor es que pueden compartir información con otras aplicaciones como procesadores de texto, hojas de calculo e Internet.

Este trabajo presenta los conceptos requeridos para diseñar, utilizar e implementar Bases de Datos relaciónales y esta dividido en cuatro capitulos en los cuales se explicará el porque de la implementación de Bases de Datos para el manejo eficiente de información,

tomando como ejemplo practico la implementación de una Base de Datos Relacional para el Grupo Financiero Inbursa.

Capitulo 1. En esta parte se presenta información relacionada con las Bases de Datos: Definición de las Bases de Datos, lo que son , para que sirven, caracteristicas, su aplicación, y la importancia de las mismas en el manejo de la información. El Sistema de Gestión de Bases de Datos y sus principales funciones, que hacen , el para que lo hacen y como lo hacen.

Capitulo 2. En este capitulo se explicá el modelo Cliente Servidor, conceptos, caracteristicas, beneficios, el modelo Cliente - Servidor en el desarrollo de sistemas en Red de Área Local con la finalidad de resaltar la importancia de porque en estos tiempos, el modelo Cliente - Servidor es uno de los mejores modelos para manejo de información dentro de las Organizaciones.

Capitulo 3. Aquí se expone la Normalización en las Bases de Datos que es considerada como una característica del buen diseño de Bases de Datos, esto con el fin de que los datos sean precisos. Para el diseño de Bases de Datos se requiere de una metodología, para obtener mayores rendimientos de ellas. Modelo relacional , aquí se hace referencia a un modelo que en este momento es muy eficiente en el diseño de Bases de Datos. La implementación de las Bases de Datos relaciónales en aplicaciones bajo el modelo Cliente - Servidor.

Capitulo 4. En esta parte se presenta un caso práctico que fue realizado en el Grupo Financiero Inbursa en el Sistema de Operaciones Internacionales con el modelo Cliente Servidor , utilizando el lenguaje de programación Visual Basic y el manejador de Bases de Datos relaciónales SQL. En este capitulo se muestran las caracteristicas de cada uno de ellos. Requerimientos para el desarrollo. Representación del equipo de desarrollo. Tópicos referentes al equipo de Desarrollo. La organización y arquitectura del proyecto.

OBJETIVOS

GENERAL.

Identificar los beneficios que ofrece la implementación de una Base de Datos Relacional en una Red de Área Local bajo el modelo Cliente - Servidor.

ESPECÍFICO.

Establecer las normas de la implementación de una Base de Datos Relacional en una Red de Área Local bajo el modelo Cliente - Servidor.

HIPÓTESIS

La implementación de una Base de Datos Relacional en sistemas de computo incrementará la eficiencia del manejo de información y consigo mismo una mayor productividad.

CAPITULO I

BASES DE DATOS

1.1 CONCEPTO DE BASES DE DATOS

Una Base de Datos es un conjunto de datos relacionados entre si, donde datos significa hechos registrados. Por lo regular, una Base de Datos representa un aspecto del muno real, y sirve para fines especificos de uno o más grupos de usuarios. Una colección aleatoria de datos no puede considerarse propiamente una Bases de Datos. Toda Base de Datos se diseña, construye y llena con datos para un propósito especifico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a dichos usuarios.

Una Base de Datos tiene una fuente de la cual se derivan los datos, cierto grado de interacción con los acontecimientos del mundo real y un público que está activamente interesado en el contenido de la Base de Datos.

Las Bases de Datos pueden ser de cualquier tamaño y tener diversos grados de complejidad. La organización y el control son básicos siempre, pero se hacen criticos cuando los volúmenes son mayores. Al tener una enorme cantidad de información debe organizarse y controlarse para que los usuarios puedan buscar, obtener y actualizar los datos cuando sea necesario.

La generación y el mantenimiento de la Base de Datos pueden ser manuales o mecánicos. Un catálogo de tarjetas es un ejemplo de Bases de Datos que se puede crear y mantener manualmente. Las Bases de Datos computarizadas se pueden crear y mantener con un

grupo de programas de aplicación escritos especificamente para esa tarea, o bien mediante un Sistema de Gestión de Bases de Datos (se tratará más adelante).

1.2 IMPORTANCIA DE LAS BASES DE DATOS

Las Bases de Datos, surgidas como respuesta al nuevo planteamiento de los sistemas orientados hacia los datos, para mejorar la calidad de las prestaciones de los sistemas informáticos y aumentar su rendimiento, presentan una multitud de ventajas, referidas a:

LOS DATOS.

- Independencia de éstos respecto a su tratamiento y viceversa.
- Mejor disponibilidad de los mismos.
- Mayor eficiencia en la recolección, codificación y entrada en el sistema.

LOS RESULTADOS.

- Mayor coherencia.
- Mayor valor informativo.
- Mejor y más normalizada documentación de la información.

LOS USUARIOS.

- Acceso más rápido y sencillo de los usuarios finales.
- Más facilidades para compartir los datos por el conjunto de los usuarios.
- Mayor flexibilidad para atender a demandas cambiantes.

Independencia de los datos a los tratamientos y viceversa.

La mutua independencia de datos y su tratamiento lleva a que un cambio a estos últimos no imponga un nuevo diseño lógico y/o fisico de la Base de Datos. Por otra parte, la inclusión de nueva información, desaparición de otra, cambios en la estructura fisica o en los caminos de acceso, etc., no deben obliga · a alterar los programas que se usan para trabajar con los datos.

La flexibilidad que proporciona la independencia de los datos y programas es muy importante para conseguir sin excesivos costes la continua adaptación del sistema de información a la evolución de las organizaciones.

Coherencia de los resultados.

Debido a que la información de la Base de Datos se recoge y almacena una sola vez, en todos los tratamientos se utilizan los mismos datos, por lo que los resultados de todos ellos son coherentes y perfectamente comparables. Además, al no existir la redundancia en los datos, desaparece el problema de que el cambio de un dato obliga a actualizar una serie de datos.

Mejor disponibilidad de los datos para el conjunto de los usuarios.

Cuando se aplica la metodología de Bases de Datos, cada usuario ya no es propietario de los datos, puesto que éstos se comparten entre el conjunto de aplicaciones, existiendo una mejor disponibilidad de los datos para todos los que tienen necesidad de ellos, siempre que estén autorizados para su acceso.

Hay también una mayor transparencia respecto a la información existente, ya que todos los datos que se encuentran en la Base se deben relacionar en un catálogo o diccionario, que puede ser ampliamente difundido y accedido por medios informáticos.

Mayor valor informativo.

Puesto que la Base de Datos es un sistema reflejo del mundo real, donde los distintos elementos están interrelacionados, el valor informativo de su conjunto es superior a la suma del valor informativo de los elementos individuales que lo constituyen.

Normalización de la documentación de la información, la cual esta integrada en los datos.

La documentación de los datos, realizada por el analista o programador, es en general insuficiente y a veces incluso inexistente. Además por lo común , la estandarización brilla por su ausencia. Este problema se atenúa en gran medida en las Bases de Datos, ya que en la misma Base se incluye no solo los datos, sino también la semántica de los mismos.

Mayor eficiencia en la recolección, validación y entrada de los datos al sistema.

Al no existir redundancias, los datos se recogen y validan una sola vez, aumentando así el rendimiento de todo el proceso previo al almacenamiento.

Reducción del espacio de almacenamiento.

La desaparición de las redundancias, así como la aplicación de técnicas de comparación, lleva en los sistemas de Bases de Datos a una menor ocupación de almacenamiento secundario (disco magnético). Se ha de tener presente, sin embargo, que los elementos del sistema ocupan bastante espacio.

1.3 SISTEMA DE GESTIÓN DE BASES DE DATOS

Un sistema de gestión de Bases de Datos (SGBD) es un conjunto de programas que permite a los usuarios crear y mantener una Base de Datos. Por tanto, el SGBD es un sistema de software de propósito general que facilita el proceso de definir, construir y manipular Bases de Datos para diversas aplicaciones. Para definir una Base de Datos hay que especificar los tipos de datos, las estructuras y restricciones de los datos que se almacenarán en ella. Construir una Base de Datos es el proceso de guardar los datos mismos en algún medio de almacenamiento controlado por el sistema de gestión de Bases de Datos. En la manipulación de una Base de Datos intervienen funciones como: consultar la Base de Datos para obtener datos específicos, actualizar la Base de Datos para reflejar cambios y generar informes a partir de los datos.

1.4 CARACTERÍSTICAS DE UN SISTEMA DE GESTIÓN DE BASES DE DATOS

Control de redundancia.

En la creación tradicional de programas con procesamiento de archivos, cada grupo de usuarios mantiene sus propios archivos para manejar sus aplicaciones de procesamiento de

datos. Una buena parte de los datos se almacenaría más de una vez: esto es en los archivos de cada grupo de usuarios. Otros grupos de usuarios podría duplicar parte de esos datos, o todos, en sus propios archivos.

A veces, y no pocas, esta redundancia en el almacenamiento de los mismos datos provoca varios problemas. En primer lugar, es necesario realizar una misma actualización lógica. Esto implica una duplicación del trabajo. En segundo lugar, se desperdicia espacio de almacenamiento al guardar los mismos datos en varios lugares, y este problema puede ser grave si las Bases de Datos son grandes. En tercer lugar es posible que los archivos que representan los mismos datos se tornen inconsistentes, quizá porque una actualización se haya aplicado a ciertos archivos pero no a todos.

Con el enfoque de Bases de Datos, las vistas de los diferentes grupos de usuarios se integran durante el diseño de la Base de Datos. Para conservar la consistencia, debe crear un diseño que almacene cada dato lógico, en un solo lugar de la Base de Datos. Ello evita la inconsistencia y ahorra espacio de almacenamiento.

Restricción de los accesos no autorizados.

Cuando muchos usuarios comparten una misma Base de Datos, es probable que no todos tengan autorización para tener acceso a toda la información que contiene. Además, es posible que sólo algunos usuarios tengan permiso para recuperar datos, en tanto que a otros se les permite obtenerlos y actualizarlos. Por tanto, también es preciso controlar el tipo de las operaciones de acceso (obtención o actualización). Por lo regular, a los usuarios o grupos de usuarios se les asigna números de cuenta protegidos con contraseñas, mismos que sirven para tener acceso a la Base de Datos. El Sistema de Gestión de Bases de Datos debe contar con un subsistema de seguridad y autorización que permita al administrador de la Base de Datos crear cuentas y especificar restricciones para ellas. De manera similar, podemos hacer que los usuarios sólo puedan tener acceso a la Base de Datos a través de las transacciones programadas que expresamente fueron creadas para ellos.

Almacenamiento persistente de objetos y estructuras de dato de programas.

Una aplicación reciente de las Bases de Datos consiste en ofrecer almacenamiento persistente para objetos y estructuras de datos de programas. Ésta es una de las principales razones de que se hayan creado los Sistemas de Gestión de Bases de Datos Orientados a Objetos. Es común que los lenguajes de programación cuenten con estructuras de datos complejas, como los tipos de registro en PASCAL o las definiciones de clases en C++. Los valores de las variables de un programa se desechan una vez que éste termina, a menos que el programador explícitamente los almacene en archivos permanentes; para ello, suele requerirse la conversión de esas estructuras complejas a un formato adecuado para su almacenamiento de archivos. Cuando hay que leer otra vez estos datos, el programador debe convertirlos del formato de archivos a la estructura de variables del programa. Los sistemas de Bases de Datos orientados a objetos son compatibles con lenguajes de programación del tipo de C++ y SMALLTALK, y el software del Sistema de Gestión de Bases de Datos realiza automáticamente las conversiones necesarias. Así se puede almacenar permanentemente un objeto complejo de C++ en un Sistema de Gestión de Bases de Datos orientado a objetos. Se dice que los objetos de este tipo son persistentes porque sobreviven cuando termina la ejecución del programa y después se pueden recuperar directamente mediante otro programa en C++.

El almacenamiento persistente de objetos y estructuras de datos de programas es una función importante para los sistemas de Bases de Datos. Los Sistemas de Gestión de Bases de Datos tradicionales a menudo adolecía del llamado problema de incompatibilidad porque las estructuras de datos proporcionadas por el Sistema de Gestión de Bases de Datos eran incompatibles con las del lenguaje de programación. Los sistemas de Bases de Datos orientados a objetos suelen ofrecer compatibilidad de las estructuras de datos con uno o más lenguajes de programación orientada a objetos.

Suministro de múltiples interfaces con los usuarios.

En vista de que muchos tipos de usuarios con diversos niveles de conocimientos técnicos utilizan las Bases de Datos, El sistema de Gestión de Bases de Datos debe ofrecer diferentes interfaces. Entre éstas están los lenguajes de consulta para usuarios esporádicos, las interfaces de lenguaje de programación para programadores de aplicaciones, las formas y códigos de órdenes para los usuarios paramétricos y las interfaces controladas por menús y en lenguaje natural para los usuarios autónomos.

Representación de vínculos complejos entre los datos.

Una Base de Datos puede contener numerosos conjuntos de datos que estén relacionados entre sí de muchas maneras. Es preciso que el Sistema de Gestión de Bases de Datos pueda representar diversos vínculos complejos de los datos y también obtener y actualizar con rapidez y eficiencia datos que estén mutuamente relacionados.

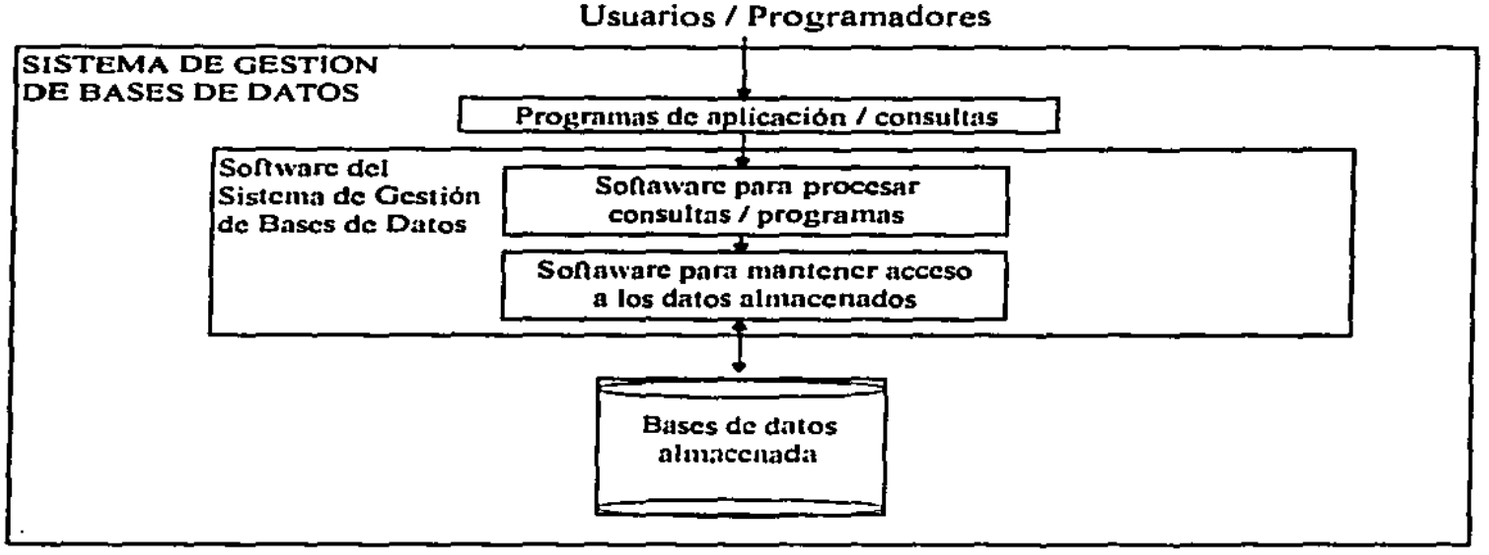
Cumplimiento de las restricciones de integridad.

La mayor parte de las aplicaciones de Bases de Datos tienen ciertas restricciones de integridad que deben cumplir los datos. El Sistema de Gestión de Bases de Datos debe ofrecer recursos para definir tales restricciones y hacer que se cumplan. La forma más simple de restringir la integridad consiste en especificar un tipo de datos para cada elemento de información. Otro tipo de restricción implica especificar que un registro de un archivo debe relacionarse con registros de otros archivos. Otro tipo de restricción especifica que los valores de los elementos de información sean únicos. Es responsabilidad de los diseñadores de la Base de Datos identificar las restricciones de integridad durante el diseño. Algunas restricciones se pueden especificar en el Sistema de Gestión de Bases de Datos, el cual hará automáticamente que se cumplan; otras pueden requerir verificación mediante programas de actualización o en el momento en que se introducen los datos.

Respaldo y recuperación.

Todo Sistema de Gestión de Bases de Datos debe contar con recursos para recuperarse de fallos de hardware o de software. Para ello está el subsistema de respaldo y recuperación del

Sistema de Gestión de Bases de Datos. Por ejemplo si el sistema falla mientras se está ejecutando un complejo programa de actualización, el subsistema de recuperación se encargará de asegurarse de que la Base de Datos se restaure al estado en el que estaba antes de que comenzara la ejecución del programa. Como alternativa, el subsistema de recuperación puede asegurarse de que su efecto completo se registre en la Base de Datos.



Sistema de Gestión de Bases de Datos

1.5 FUNCIONES PRINCIPALES DEL SISTEMA DE GESTIÓN DE BASES DE DATOS

Las funciones principales de un Sistema de Gestión de Bases de Datos son la descripción, manipulación y utilización.

Función de descripción o definición.

Esta función debe permitir al administrador de la Base de Datos especificar los elementos de datos que la integran, su estructura y las relaciones que existen entre ellos, las reglas de integridad, los controles a efectuar antes de autorizar el acceso a la Base de Datos.

Esta función, realizada por el lenguaje de descripción o definición de datos propio de cada Sistema de Gestión de Bases de Datos debe suministrar los medios para definir las tres estructuras de datos (externa, lógica global e interna), especificando las características de los datos a cada uno de estos niveles.

A nivel interno, se ha de indicar el espacio reservado para la Base, la longitud de los campos o elementos de datos.

Para las estructuras externa y lógica global, la función de descripción ha de proporcionar los instrumentos para la definición de las entidades y su identificación, atributos de las mismas, interrelaciones entre ellas, autorizaciones de acceso, restricciones de integridad, etc. Las descripciones de las estructuras lógicas de los usuarios han de estar referidas a la estructura lógica global.

Función de manipulación.

La función de la manipulación permite a los usuarios de la Base, buscar, añadir, suprimir o modificar los datos de la misma, siempre de acuerdo con las especificaciones y las normas de seguridad dictadas por el administrador.

La función de manipulación se llevará a cabo por medio de un lenguaje de manipulación de datos que facilita los instrumentos necesarios para la realización de estas tareas.

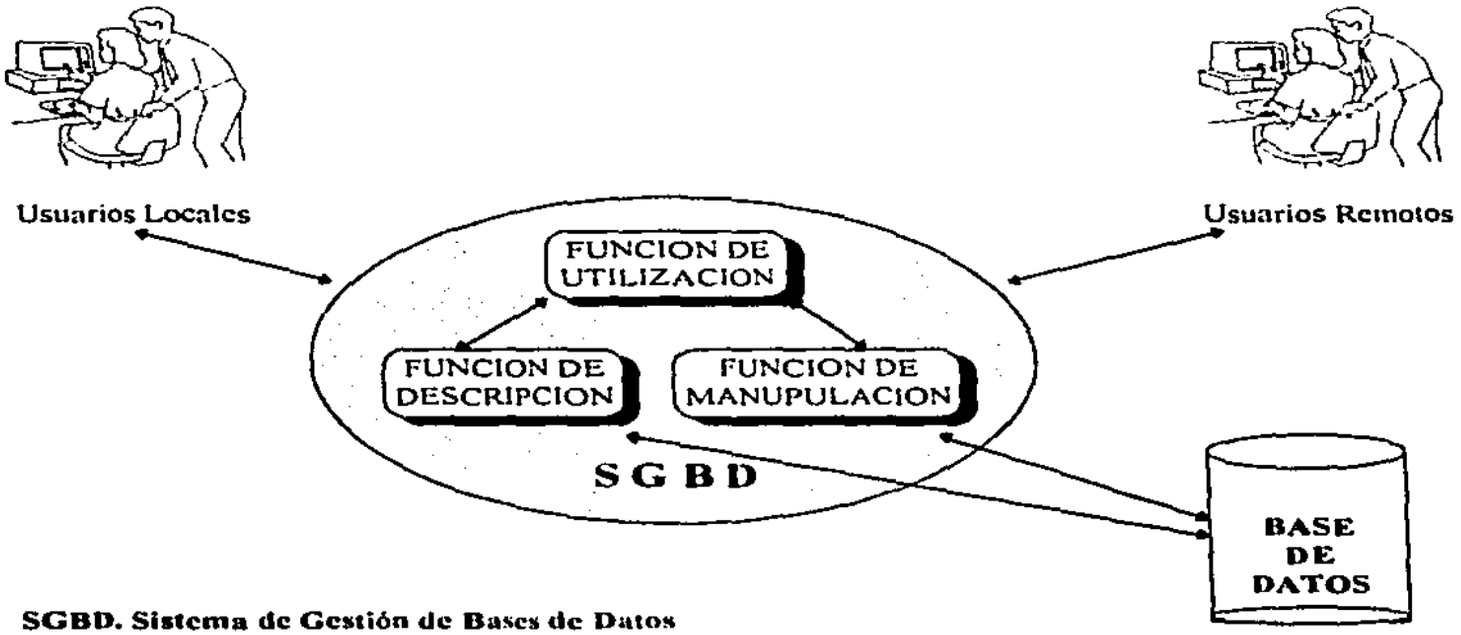
Función de utilización.

La función de utilización reúne todos los interfaces que necesitan los diferentes usuarios para comunicarse con la Base de Datos y proporciona un conjunto de procedimientos para el administrador.

Las exigencias respecto a la forma de utilizar la Base de Datos son muy diferentes, según los tipos de procesos y según los usuarios, siendo preciso que la función de utilización

responda a todas ellas. En especial, esta función debe integrar una serie de instrumentos que faciliten las tareas del administrador.

En la mayoría de los Sistemas de Gestión de Bases de Datos existen funciones de servicio, como cambiar la capacidad de los datos, obtener estadísticas de utilización, cargar archivos, etc., y principalmente las relacionadas con la seguridad física (copias de seguridad, rearranque en caso de caída del sistema, etc.) y de protección frente a accesos no autorizados. Todas ellas se consideran comprendidas en la función de utilización. Ver Figura 1.1



SGBD. Sistema de Gestión de Bases de Datos

Figura 1.1

CAPITULO 2

MODELO CLIENTE - SERVIDOR

2.1 CONCEPTO DEL MODELO CLIENTE - SERVIDOR

Es necesario para realizar el estudio del modelo Cliente - Servidor diferenciarlo de los Servidores de Archivos.

Las aplicaciones durante muchos años, han sido desarrolladas con los tradicionales menejadores de archivos tales como DBASE, Clipper y las primeras versiones de FoxPro, entre otros, cuya capacidad no va más allá del manejo de archivos en forma semirelacional. Como la búsqueda de información entre distintos archivos para obtener datos de una consulta para un informe especifico, y donde muchas de las reglas del negocio son validadas desde el código de la aplicación y no desde el origen de los datos.

Estas aplicaciones que empezaron en PC tradicionales tuvieron que ser llevadas a más usuarios que pudieran accesar la misma información, lo que originó tres grandes problemas: primero, que la velocidad de la aplicación se viera afectada debido a que tendria que ser accesada por más usuarios al mismo tiempo, lo que obligó a emigrar la aplicación a un ambiente multiusuario mediante la instalación de la Base de Datos (o archivos semirelacionados) en una Red de Área Local dándoles acceso a los usuarios para que pudieran cargar la aplicación fuera del Servidor y compartir los mismos archivos. Lo anterior lleva a un segundo problema, donde la seguridad de los datos se pone en peligro debido al aumento de la concurrencia donde el control de la misma se complica cuando dos o más usuarios intentan accesar al mismo registro a la vez. Si la transacción es de solo lectura, el problema se soluciona dando derechos de solo lectura a ambos usuarios sobre el archivo. Cuando un usuario intenta consultar y el otro actualizar, muchos manejadores de archivos otorgan accesos de lectura/escritura sobre el

archivo para el primer usuario que entre y de solo lectura para los demás usuarios y otros utilizan incluso el bloqueo a nivel registro.

Pero independientemente de como se manejan las transacciones, existe un tercer problema donde el uso de un Servidor de Archivos tradicional aumenta el trafico de la red. Cuando un usuario desea accesar a la Base de Datos el Servidor de Archivos envia el código de la aplicación a la estación de trabajo donde realmente se produce el procesamiento, que puede ser desde una simple petición hasta el archivo complejo junto con sus indices relacionados. Al aumentar más usuarios a una aplicación aumenta el trafico de la red disminuyendo asi el rendimiento de la misma. No es por causa del Servidor de Archivos este problema, porque éste realmente realiza sólo la tares de disco duro remoto, ni tampoco por la estación de trabajo ya que si el archivo es demasiado grande para la capacidad de la PC, la aplicación se degradará más ahi que en otra estación con mucho mayor capacidad.

Lo anterior es bastante preocupante, pero es peor si pensamos que los formatos de archivos de éstos manejadores son compatibles entre si, y que pueden ser accesados por otras aplicaciones donde un usuario puede saltarse la seguridad creada por la aplicación y entrar desde un programa externo para realizar modificaciones al archivo y asi corromperlo.

Una solución a los problemas antes mencionados es la utilización de la arquitectura Cliente - Servidor.

El termino Cliente - Servidor fue utilizado por primera vez en 1987, por la compañía Forrester Research, Inc., con la publicación de in informe titulado "The New Client/Server Paradigm". tanto Microsoft y SYBASE fueron los primeros en adoptar este término para su Servidor de SQL. En el Capitulo 4 se ahondará en la herramienta SQL.

El modelo **Cliente - Servidor** es un modelo en donde el procesamiento y la funcionalidad de un sistema o aplicación está distribuido entre dos elementos. El primero llamado Cliente (mejor conocido como Front-End) y el segundo llamado Servidor (mejor

conocido como Back-End). El hecho de mencionar un Servidor de datos no significa que esta arquitectura no pueda enfocarse a otras aplicaciones. Las aplicaciones de Base de Datos son las más comunes.

En éste modelo el usuario final hace uso de los dos elementos el Cliente y el Servidor. Pero solamente el interactúa con el Cliente (Front-End) donde el usuario solicita información del Servidor de Bases de Datos. Este recibe la petición, la procesa y envía de vuelta los resultados al Cliente para ser desplegados.

Cabe mencionar, que tanto el diseño lógico como físico de la aplicación se encuentra separados entre el Cliente y el Servidor, lo que permite que la carga de trabajo se coordine entre ambos elementos y donde cada uno es responsable de utilizar y administrar sus recursos de manera eficiente para completar exitosamente cada una de sus tareas.

El papel del Cliente.

La única parte de un sistema Cliente - Servidor con la cual el usuario final tiene contacto, es con la parte del Cliente. El cual es una aplicación Front-End que se encuentra en la computadora del usuario y sobre la cual éste accesa a la información de la Base de Datos. Hoy en día, la mayoría de los sistemas Cliente - Servidor cuentan con aplicaciones gráficas mejor conocidas como GUI (Graphical User Interface) que ofrecen la ventaja de enviar una petición a la Base de Datos, validar la información, realizar cálculos sobre la misma y darle una presentación adecuada para ofrecer una interface amigable con el usuario. Lo anterior, es posible gracias a las diversas herramientas que se encuentran en el mercado, las cuales contienen controles estándares tales como un listbox, command button, checkbox entre otros, que le permiten navegar al usuario entre una y otra aplicación sin sufrir grandes cambios. Entre las herramientas más conocidas podemos mencionar a PowerBuilder, Visual Basic y Delphi. En el Capitulo 4 se ahondará en la herramienta Visual Basic.

El papel del Servidor.

El Servidor o Back-End, no consiste más que en un manejador de Bases de Datos relaciónales (RDBMS) que corre en una computadora total y absolutamente independiente al Cliente, en la cual se almacena la información de la aplicación. Debido a que los RDBMS cuentan con numerosas funciones de mantenimiento que aseguran la integridad de la información, el cumplimiento de la reglas del negocio mediante el manejo adecuado de bloqueos, transacciones, respaldos y monitoreo queda garantizada. Permitiendo la facilidad de que simultáneamente múltiples usuarios accesen a la información desde su aplicación Front-End para poder leer, actualizar, insertar y borrar datos sin que el rendimiento de la aplicación se vea perjudicada.

Aunque no suele ser una opción demasiada ventajosa, la Base de Datos puede colocarse en el Servidor de archivos de la Red de Área Local, con el riesgo de que al crecer la Base de Datos y el número de usuarios, el trabajo del Servidor de archivos y el RDBMS puedan degradarse y a la larga se coloquen cada uno en maquinas separadas. Por ello, lo optimo es colocar el RDBMS en su propio sistema dedicado desde un principio.

El papel de la Red.

A pesar de que el modelo solamente menciona al Cliente y al Servidor como elementos de un sistema, es necesario hablar de que existe un tercer elemento que es vital para el funcionamiento de la misma este es la Red. Ésta juega un papel determinante en el rendimiento de los sistemas Cliente - Servidor, porque ella controla que tan rápido viajan las peticiones del Cliente hacia el Servidor y la rapidez en la que son devueltas. De ahí, la gran importancia de un buen diseño y planeación de un sistema Cliente - Servidor que sea capaz de minimizar el tráfico de la red.

Ventajas del modelo Cliente - Servidor sobre Servidores de Archivos.

Las ventajas del modelo Cliente - Servidor sobre las aplicaciones basadas en Servidores de archivos son enormes. Primero, porque el rendimiento de la aplicación de Base de Datos ya no depende del Cliente, éste solo tiene la tarea de generar la consulta o actualización de los

datos y enviársela al servido de Base de Datos, el cual procesa la petición y envía de vuelta los datos que cumplen con el criterio de consulta.

El modelo Cliente - Servidor es más confiable en la seguridad de los datos, ya que los usuarios no tienen acceso directo sobre los archivos de datos para realizar consultas o actualizaciones. De hecho, el usuario no accesa a los archivos de datos, todo el acceso lo proporciona el RDBMS permite otorgar y revocar privilegios sobre los tipos de acciones que deseen realizar sobre el registro o incluso sobre determinadas columnas del mismo.

Otra de las ventajas es que el Cliente es total y absolutamente independiente del Servidor de Base de Datos. Con las Bases de datos tradicionales, la aplicación tenía que escribirse en el lenguaje de programación del manejador de archivos.

En la figura 2.1 se muestra como trabaja la arquitectura Cliente - Servidor, nótese la gran diferencia con el Modelo Servidor de Archivos. El tráfico de la red disminuye y tanto el Cliente como el Servidor administran sus recursos.

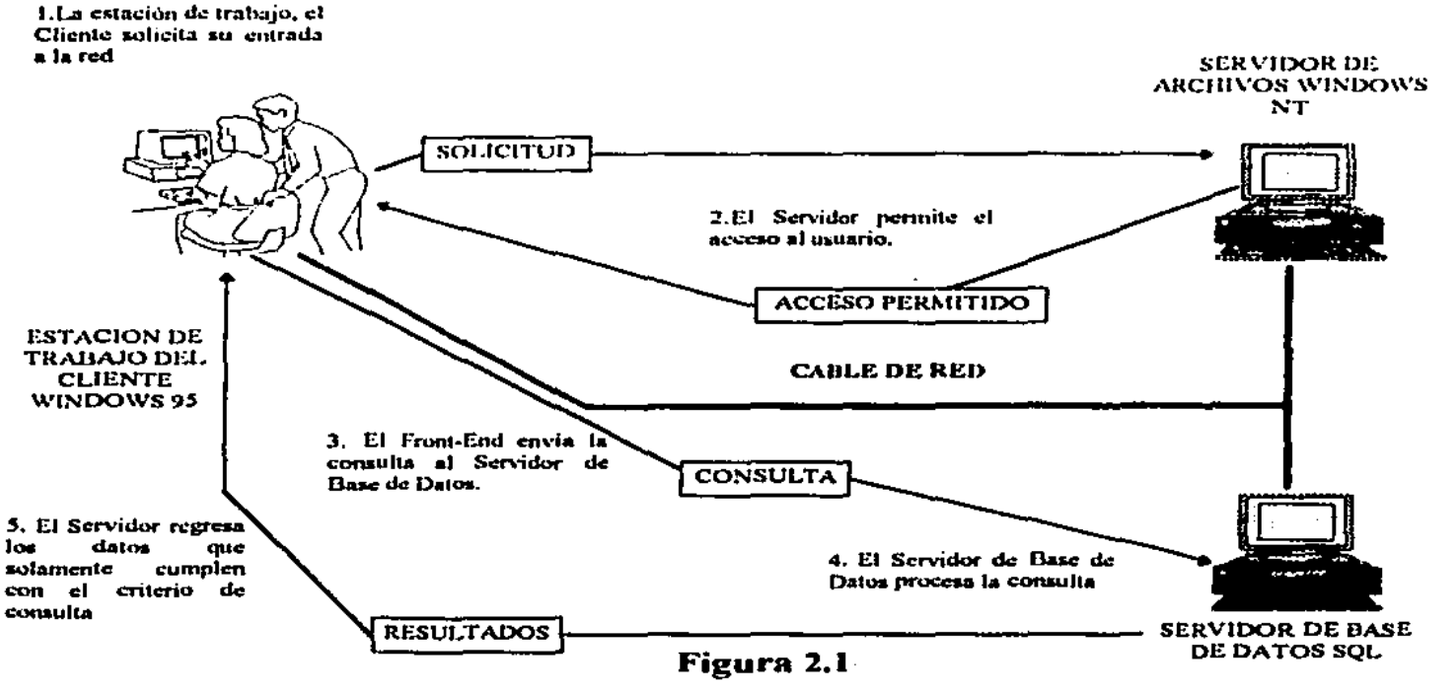


Figura 2.1

El modelo Cliente - Servidor divide las tareas de la aplicación. El Back-End administra la información. El Front-End presenta la interface al usuario disminuyendo el trafico de la red.

2.2 BENEFICIOS DEL MODELO CLIENTE - SERVIDOR

Una aplicación desarrollada bajo el modelo Cliente - Servidor ofrece muchas ventajas en comparación con las aplicaciones tradicionales, principalmente porque permite que el corporativo junto con los empleados obtengan grandes beneficios que se ven reflejados en el incremento de la productividad. El personal, tiene acceso fácil a la información que le permite tomar decisiones correctas, facilidad de comunicación, reducción del tiempo invertido en las actividades administrativas y menos esfuerzo y costo en la realización de las tareas.

Acceso a la información.

Un buen diseño de este modelo permite que el usuario final accese de manera fácil a la información para poder realizar su trabajo adecuadamente. Por ejemplo el usuario a través de un simple click de mouse en la aplicación Front-End, puede obtener un conjunto de datos sin la necesidad de saber si estos se encuentran en distintas Bases de Datos. Además, la funcionalidad es proporcionada por un solo sistema y no varios, donde el usuario debe conocer cada uno e ir viajando entre ellos para obtener el mismo resultado que se obtiene en una aplicación Cliente - Servidor con tan solo un click.

Incremento de la productividad.

Un sistema Cliente - Servidor incrementa la productividad del usuario mediante el uso de herramientas que permiten completar las tareas de una manera rápida y fácil. Por ejemplo el usuario final a través de su aplicación Front-End puede hacer publico algún reporte de la Base de Datos. Ello, gracias a que SQL permite colocar una petición como una página WEB para la intranet de una compañía, evitando que el usuario tenga que ir a fotocopiar el reporte para cada una de las distintas áreas a las que va dirigido.

Automatización de procesos.

Con Cliente - Servidor muchos de los procesos manuales que se realizaban, actualmente se pueden automatizar con un margen mínimo de errores. Por ejemplo , la persona encargada del respaldo de una Base de Datos puede automatizar este proceso mediante el uso de una calendarización de tareas en el RDBMS. Ejecutándose un respaldo de la Base de Datos cada hora.

Facilidad de generar reportes.

Debido a que los sistemas Cliente - Servidor son almacenados en Bases de Datos relaciónales, la información puede ser manejada fácilmente para poder obtener una gran variedad de reportes utilizando sentencias SQL. Por ejemplo, el Data Direct Explorer de Intersolve es una herramienta entre muchas que permite a usuarios finales realizar reportes rápidos y sencillos mediante el uso de búsquedas cruzadas sin la necesidad de aprender a programar en SQL.

Desarrollo rápido de la aplicación.

Actualmente el mercado ofrece gran cantidad de herramientas que permiten la programación orientada a eventos y el desarrollo de aplicaciones modulares. Ello permite a los programadores diseñar y desarrollar aplicaciones en forma rápida, debido a la reutilización del código de programas que ya existen para poder generar nuevos sistemas. Incluso las actuales aplicaciones Cliente - Servidor pueden ser fácilmente modificadas como consecuencia de cambios en reglas del negocio. Además, un cambio en el Cliente no forzosamente significa un cambio en el Servidor o viceversa.

Reducción de costos y mantenimiento.

Los sistemas Cliente - Servidor han reducido considerablemente los costos al sustituir los sistemas Mainframe. Lo que a permitido que las empresas se vean beneficiadas en tener su información en línea, ahorrar tiempo y dinero en la distribución y almacenamiento del papeleo, así como reducir el número de empleados involucrados.

Respuesta rápida a cambios del mercado.

El mercado se encuentra en constante cambio en todos sus ámbitos que, van desde la foma de administrar un pequeño negocio hasta los corporativos que manejan grandes y criticos volúmenes de información. Lo que obliga a cada empresa ser más competitiva para poder sobrevivir, mismo que se logra teniendo sistemas confiables, seguros y rápidos que puedan adaptarse a las nuevas reglas del negocio.

En los años 80's los sistemas fueron desarrollados en computadoras centrales donde todo.el procesamiento de la Red de Área Local era soportado por el mainframe. Cuando se realizaba un cambio en la empresa ya sea por nuevas políticas o lanzamiento de un nuevo producto representaba un gran reto para la gente de sistemas ya que tenian que buscar en corto tiempo miles de lineas de código y realizar los cambios necesarios a la aplicación, lo que requería más tiempo, costos y personal.

Los sistemas Cliente - Servidor permiten responder rápidamente a los cambios del mercado ya que el trabajo es repartido entre los elementos que forman ente modelo.

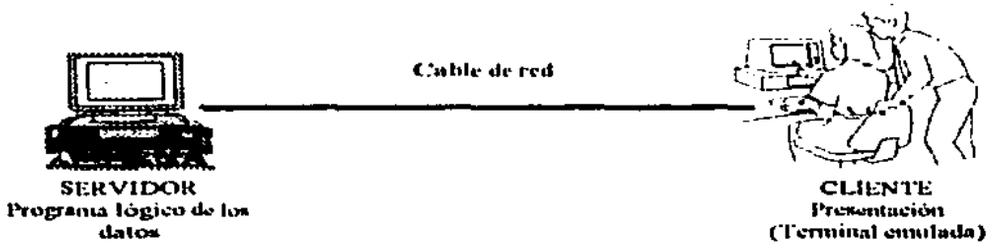
2.3 MODELO CLIENTE - SERVIDOR EN DESARROLLO DE SISTEMAS.

Los sistemas Cliente - Servidor son clasificados basandose en su construcción. La diferencia entre estos radica en el diseño y la lógica de la aplicación que se utilizaron para implantar un sistema Cliente - Servidor. Entre mayor sea la participación del Servidor en la lógica de la aplicación menor será la participación del Cliente y viceversa. Estos modelos no son mutuamente excluyentes lo que significa que un sistema construido en un modelo puede funcionar en otro debido a que el contexto es el mismo, donde el Cliente solicita una petición y el Servidor responde a ella. El modelo que se elija dependerá de las necesidades y recursos de la compañía.

Presentación Distribuida.

La presentación distribuida significa que tanto el Cliente como el Servidor se encarga de dar formato a los datos que el usuario fina visualiza. El Servidor envia los datos semiformateados al Cliente el cual se encarga de darles el toque final antes de presentarlos al usuario.

El beneficio que ofrece este modelo es que permite migrar los sistemas basados en mainframe a sistemas Cliente - Servidor en tan solo unos cuantos pasos, debido a que el Cliente puede comportarse como una terminal de mainframe que recibe los datos del Servidor y los copia hacia la aplicación. A este tipo de copia se le conoce como Screen Scraping, donde se esconden las pantallas recibidas por el Servidor y son presentadas bajo la interface de la PC como Windows u OS/2. Ejemplo en la Figura 2.2



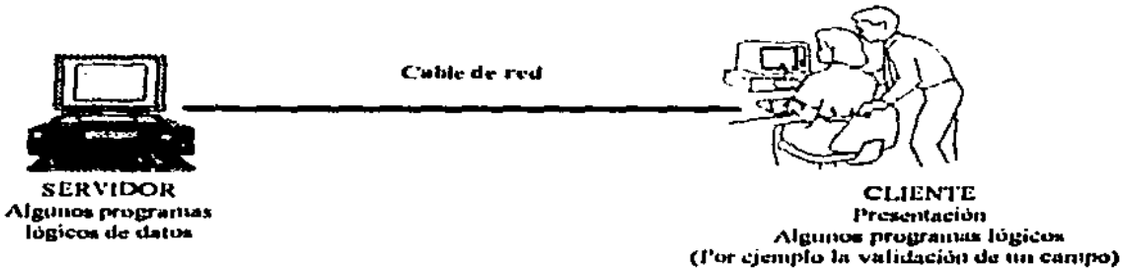
MODELO DE PRESENTACION DISTRIBUIDA

Figura 2.2

Presentación Remota.

En algunos casos es necesario trasladar alguna parte lógica del Servidor al Cliente, a lo que se le conoce como presentación remota. Debido a que en este modelo el Servidor envia los datos crudos al Cliente, la mayoria del manejo lógico de la aplicación reside en éste, y que puede ser desde la validación de un dato hasta cálculos complejos y formateo de los mismos para poderlos presentar al usuario final. Ello no significa que el Servidor pierda por completo la lógica de la aplicación ya que el puede manejar la integridad de los datos a través de

procedimientos que se disparan al momento de realizar una operación con la Base de datos al modificarla. Ejemplo Figura 2.3

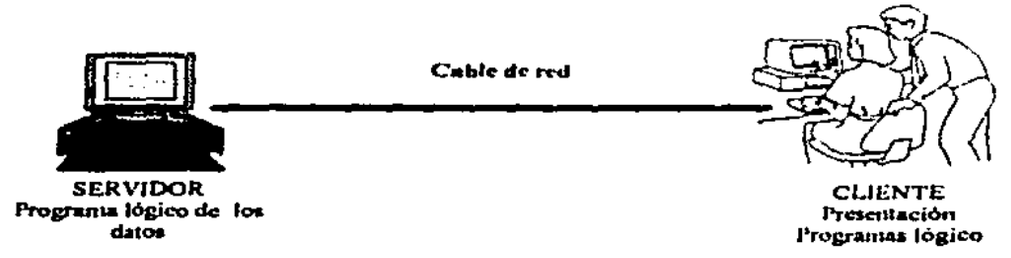


MODELO DE PRESENTACION REMOTA

Figura 2.3

Distribución Lógica.

En este modelo, los sistemas Cliente - Servidor dividen la lógica de la aplicación entre el Cliente y el Servidor. El Cliente mediante el uso de aplicaciones gráficas mejor conocidas como GUI (Grafical User Interface), controla el flujo de la aplicación mientras que las reglas del negocio y de la Base de Datos son controladas por el Servidor. La comunicación entre el Cliente y el Servidor se realiza a través del uso de llamadas a procedimientos remotos. Ejemplo en la Figura 2.4

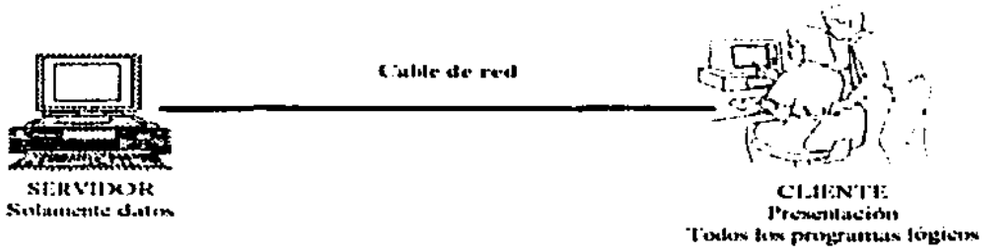


MODELO DE DISTRIBUCION LOGICA

Figura 2.4

Datos Remotos.

En este modelo el Cliente se encarga de manejar toda la aplicación lógica y su presentación al usuario final, mientras que el Servidor solamente proporciona los datos. generalmente en este modelo el Cliente utiliza ODBC (Open DataBase Connectivity), para accesar a la Base de datos almacenados en el Servidor. Hoy en dia este modelo es el más utilizado para el desarrollo de sistema Cliente - Servidor. Ejemplo en la Figura 2.5

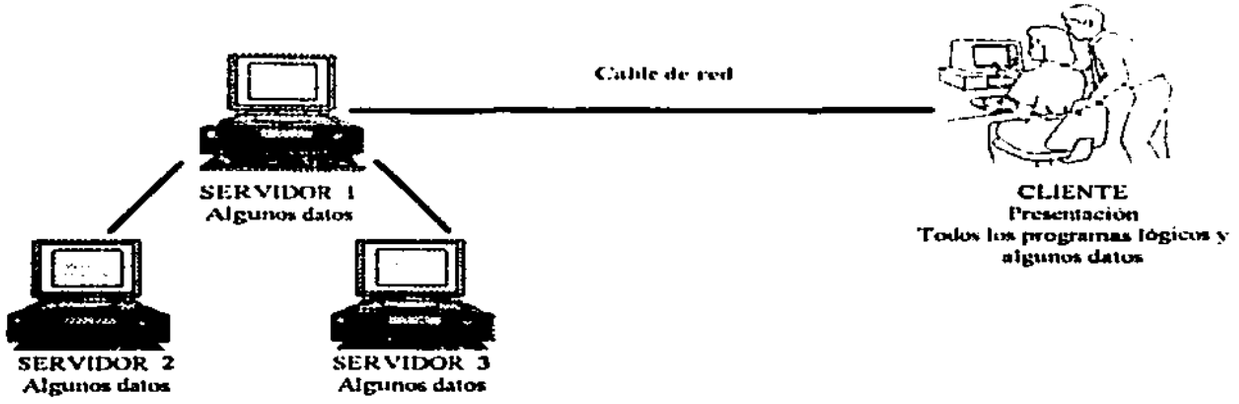


MODELO DE DATOS REMOTOS

Figura 2.5

Datos Distribuidos.

En este caso el almacenamiento de los datos esta distribuido entre varios Servidores de Bases de Datos. El manejo de los datos se realiza a través de un Cliente y el almacenamiento entre múltiples servidores. Aqui se requiere un mayor cuidado en el manejo de los datos para asegurar la consistencia, seguridad e integridad de la información. Generalmente se utiliza en la replicación, donde la información se distribuye hacia otros Servidores. Ver ejemplo en la figura Figura 2.6



MODELO DE DATOS DISTRIBUIDOS

Figura 2.6

Hardaware Necesario para el desarrollo de sistemas en Cliente - Servidor.

En los sistemas Cliente - Servidor existen tres factores que son determinantes para que un sistema construido bajo el modelo Cliente - Servidor funcione adecuadamente.

Velocidad del disco. Entre más rápido sea un disco, el software del Servidor podrá accesar a los datos mucho más rápido. Por lo tanto, un disco lento implica que el rendimiento general del sistema se vea degradado en lecturas y escrituras al disco. Lo recomendable para es punto es adquirir discos duros mayores a 1GB y de arquitectura RAID (Redundant Arrays of Inexpensive Disks).

Procesador. La mayoría del procesamiento de la aplicación se realiza en el Servidor de Bases de Datos, sin tomar en cuenta cuando varios usuarios intentan realizar consultas o modificaciones sobre los datos al mismo tiempo. Cuando mayor sea la velocidad y la capacidad del procesador, mejor será el rendimiento global del sistema. Lo recomendable para los sistemas Cliente - Servidor es que el Servidor de Bases de Datos tenga como mínimo procesador Pentium a 100 Mhz.

Memoria RAM. El último factor importante y que consiste en la cantidad de memoria RAM que dispondrá el Servidor para levantar las Bases de Datos. Generalmente los servidores de Bases de Datos requieren menos memoria que la requerida por le sistema operativo, no obstante, aunque se presentan excepciones.

CAPITULO 3

IMPLEMENTACIÓN DE BASES DE DATOS RELACIONALES

3.1 METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS

La concepción de una Base de Datos y en especial la de una Base de Datos Relacional, sigue siendo una tarea larga y costosa.

Estas dificultades inherentes al diseño de una Base de datos han de afrontarse con procedimientos ordenados y metódicos. En distintas áreas de la ingeniería de software se han realizado importantes esfuerzos para encontrar las metodologías más adecuadas; esto se debe al gran impacto que una metodología tiene en el desarrollo de un producto software, ya sea en lo que se refiere a los costos y plazos de entrega del mismo, como a la calidad y mantenimiento del producto.

Un buen diseño de Bases de Datos es la clave de una buena ingeniería de software. Un sistema de software bien diseñado es fácil de aplicar y mantener, además de ser comprensible y fiable. Los sistemas mal diseñados, aunque puedan funcionar, serán costosos de mantener, difíciles de probar y poco fiables.

Una metodología de diseño de Bases de Datos es un conjunto de modelos y herramientas que nos permiten pasar de una etapa a la siguiente en el proceso de diseño.

En la determinación de las fases de la metodología se define una jerarquía de niveles de abstracción que resulte apropiada, en el sentido de ser lo suficientemente amplia para que a cada nivel le correspondan decisiones de diseño bien definidas.

El problema del diseño de Bases de Datos puede expresarse como diseñar la estructura lógica y física de una o más Bases de Datos para atender las necesidades de información de los usuarios en una organización para un conjunto definido de aplicaciones.

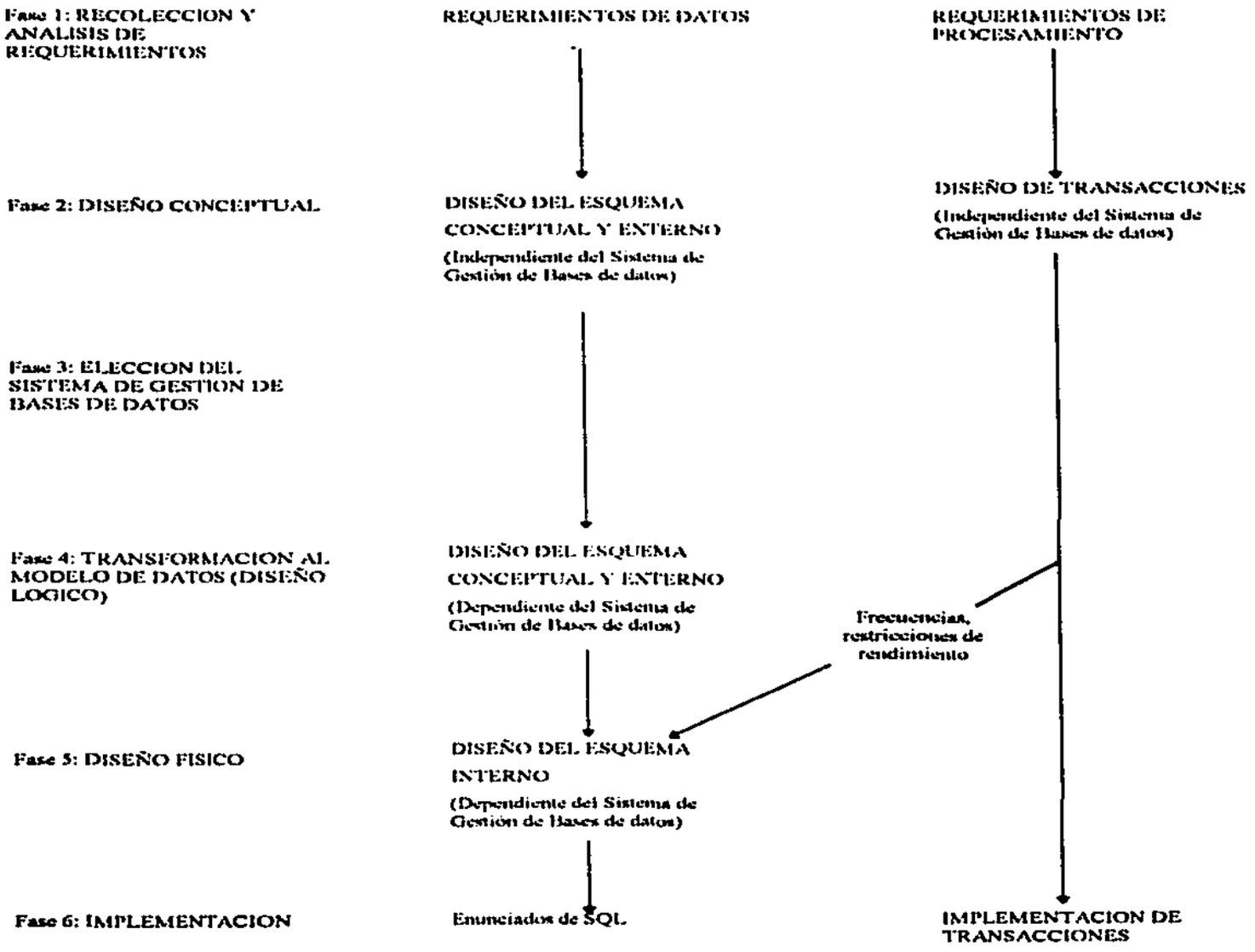
Las metas de un diseño de Bases de Datos son múltiples: satisfacer los requerimientos de contenido de información de los usuarios y aplicaciones especificados; proveer una estructuración de la información natural y fácil de entender, y apoyar los requerimientos de procesamiento y cualesquier otros objetivos de rendimiento, como el tiempo de respuesta, el tiempo de procesamiento y el espacio de almacenamiento. Es muy difícil lograr y medir estas metas. El problema se agrava porque el proceso del diseño de Bases de Datos a menudo comienza con requerimientos muy informales y muy mal definidos. En contraste, el resultado de la actividad de diseño en un esquema de Bases de Datos rígidamente definido que no se podrá modificar fácilmente una vez implementada.

Para el diseño de Bases de Datos existen seis fases principales:

1. Recolección y análisis de requerimientos.
2. Diseño conceptual.
3. Elección de un Sistema de Gestión de Bases de Datos.
4. Transformación al modelo de datos (diseño lógico).
5. Diseño físico.
6. Implementación del sistema de Bases de Datos.

El proceso de diseño consta de dos actividades paralelas como se muestra en la figura 3.1. La primera actividad implica el diseño del contenido de datos y estructura de la Base de Datos; la segunda atañe el diseño del procesamiento de la Base de Datos y de las aplicaciones de software. Estas dos actividades están íntimamente entrelazadas. Por ejemplo, podemos identificar los elementos de información que se almacenarán en la Base de Datos analizando las aplicaciones de esta última. Además, la fase de diseño físico de la Base de Datos, durante la

cual se eligen las estructuras de almacenamiento y los caminos de acceso de los archivos de la Bases de Datos, depende de las aplicaciones que van a utilizar estos archivos.



Proceso de diseño en dos actividades paralelas

Figura 3.1

Fase 1. Recolección y análisis de requerimientos.

Antes de poder diseñar eficazmente una Base de Datos, se debe conocer las expectativas de los usuarios y los usos que se piensa dar a la Base de Datos con el mayor detalle posible. El proceso de identificar y analizar los usos propuestos se denomina recolección y análisis de requerimientos. Para especificar los requerimientos, primero se debe identificar las demás partes del sistema de información que van a interactuar con el sistema de Base de Datos. Entre ellas están los usuarios y las aplicaciones nuevos y ya existentes. Una vez realizado lo anterior se reunirán y analizarán los requerimientos de dichos usuarios y aplicaciones. Por lo regular, esta fase incluye las siguientes actividades:

· *Identificación de las principales áreas de aplicación y grupos de usuarios que utilizarán la Base de Datos.* Se eligen individuos clave dentro de cada grupo como participantes destacados en los pasos subsecuentes de recolección y especificación de requerimientos.

· *Estudio y análisis de la documentación existente relativa a las aplicaciones.* Se repasa otra documentación; manuales de politicas, formas, informes y diagramas de organización para determinar si influye de alguna manera sobre el proceso de recolección y especificación de requerimientos.

· *Estudio del entorno de operación actual y de los planes de aprovechamiento de la información.* Esto incluye el análisis de los tipos de transacciones y de sus frecuencias, asi como del flujo de información dentro del sistema. Se especifican los datos de entrada y salida de las transacciones.

· *Recolección de respuestas escritas a grupos de preguntas hechas a los posibles usuarios de la Base de Datos.* Estas preguntas se refieren a las prioridades de los usuarios y a la importancia que dan a las diversas aplicaciones. Posiblemente se entrevisten individuos clave que ayudarán a estimar el valor de la información y a establecer las prioridades.

Los métodos anteriores para recabar los requerimientos producen especificaciones de requerimientos mal estructuradas y en su mayor parte informales, las cuales se convierten después a una forma más estructurada mediante una de las técnicas de especificación de requerimientos más formales. Estas incluyen los diagramas HIPO (entrada / procesamiento / salida jerárquicos), la técnica de análisis y diseño estructurado, los diagramas de flujo entre otros. Todos estos son métodos diagramaticos para organizar y presentar los requerimientos de procesamiento de la información.

Fase 2. Diseño conceptual de la Base de Datos.

La segunda fase del diseño de Base de Datos implica dos actividades paralelas. la primera, el diseño del esquema conceptual, examina los requerimientos de datos resultantes de la fase 1 y produce un esquema de Base de Datos conceptual. La segunda actividad, el diseño de transacciones, examina las aplicaciones de Base de Datos analizadas en la fase 1 y produce especificaciones de alto nivel para estas transacciones.

Diseño del esquema conceptual. El esquema conceptual que resulta de esta fase suele estar contenido en un modelo de datos de alto nivel independiente del Sistema de Gestión de Bases de Datos, de modo que no puede usarse directamente para implementar la Base de Datos. la importancia de un esquema conceptual independiente del Sistema de Gestión de Bases de Datos no puede sobre estimarse, por las siguientes razones:

- La meta del diseño del esquema conceptual es un entendimiento completo de la estructura, el significado, los vínculos y las restricciones de la Base de Datos.

- El esquema conceptual es muy valioso como una descripción estable del contenido de la Base de Datos.

- Un buen entendimiento del esquema conceptual es decisivo para los usuarios de la Base de Datos y los diseñadores de aplicaciones.

- La descripción diagramatica del esquema conceptual puede servir como un excelente vehículo de comunicación entre los usuarios, diseñadores y analistas de la Base de Datos.

En esta fase del diseño, es importante usar un modelo de datos conceptual que tenga las siguientes caracteristicas:

- Expresividad: El modelo de datos debe ser lo bastante expresivo para distinguir los diferentes tipos de datos, vinculos y restricciones.

- Sencillez: El modelo debe ser lo bastante simple para que la generalidad de los usuarios no especialistas comprendan y usen sus conceptos.

- Minimalidad: El modelo debe tener un número pequeño de conceptos básicos cuyo significado sea distinto y no se traslape.

- Representación diagramatica: El modelo debe contar con una representación diagramatica para mostrar un esquema conceptual que sea fácil de interpretar.

- Formalidad: Un esquema conceptual expresado en el modelo de datos debe representar una especificación formal, sin ambigüedad, de los datos. Por tanto, los conceptos del modelo deben definirse con exactitud y sin que haya posibilidad de confusión

Para diseñar un esquema conceptual debemos identificar los componentes básicos del esquema: los tipos de entidades, los tipos de vínculos y sus atributos.

Hay dos enfoques para diseñar el esquema conceptual. En el primero, llamado enfoque centralizado de diseño del esquema, los requerimientos de las diferentes aplicaciones y grupos de usuario de la fase 1 se combinarán en un solo conjunto de requerimientos antes de iniciarse el diseño del esquema. En el segundo enfoque, el que es llamado enfoque de integración de

vistas, no se combinan los requerimientos; más bien, se diseña un esquema (o vista) para cada grupo de usuarios o aplicación con Base sólo en sus requerimientos.

La diferencia principal entre los dos enfoques radica en la manera y las circunstancias en que las diferentes vistas o requerimientos de los múltiples usuarios y aplicaciones se concilian y combinan. En el enfoque centralizado, la reconciliación la efectúa manualmente el personal de la Administración de la Base de Datos antes de diseñar algún esquema, y se aplica directamente a los requerimientos recabados en la fase 1. En el enfoque de integración de vistas, cada grupo de usuarios o aplicación diseñada y realiza realmente su propio esquema conceptual a partir de sus requerimientos. Luego la Administración de Base de Datos aplica un proceso de integración a estos esquemas (vistas) para formar el esquema global integrado.

Dado un conjunto de requerimientos, sea para un solo usuario o para una comunidad de usuarios grande, debemos crear un esquema conceptual que satisfaga dichos requerimientos. Hay varias estrategias para diseñar tales esquemas, la mayoría de ellas seguirán un enfoque incremental; es decir parten de ciertas construcciones de esquema derivadas de los requerimientos y luego modifican, refinan o desarrollan de manera incremental dichas construcciones. Alguna de estas estrategias son las siguientes:

- Estrategia descendente: Se parte de un esquema que contiene abstracciones de alto nivel y luego se aplican refinaciones descendentes sucesivas.

- Estrategia ascendente: Se parte de un esquema que contiene abstracciones básicas, y luego se combinan o se les añaden otras abstracciones. Por ejemplo se comienza con los atributos y son agrupados en tipos de entidades y vinculos. Conforme avanza el diseño, podríamos añadir nuevos vinculos entre tipos de entidades.

- Estrategia de adentro hacia afuera: Este es un caso especial de estrategia ascendente, en el que la atención se encuentra en un conjunto central de conceptos que son los más

evidentes. A continuación el modelo se extiende hacia afuera al considerar conceptos nuevos en las cercanías de los ya existentes.

· Estrategia mixta: En vez de seguir una estrategia especifica durante todo el diseño, los requerimientos se olvidarán según una estrategia descendente, y se diseñará una parte del esquema para cada partición de acuerdo con una estrategia ascendente. por último, se combinan las diferentes partes del esquema.

En el caso de Bases de Datos grandes que se espera tendrán muchos usuarios y aplicaciones, puede usarse el enfoque de integración de vistas, diseñando esquemas individuales y luego cambiándolos. Ya que es posible tener vistas individuales relativamente pequeñas, el diseño de los esquemas se simplifica.

Diseño de transacciones. El propósito del diseño de transacciones en paralelo con el diseño del esquema conceptual es diseñar las características de las transacciones conocidas como de la Base de Datos con independencia del Sistema de Gestión de Base de Datos. Cuando se está diseñando un sistema de Base de Datos, los diseñadores están concientes de muchas aplicaciones conocidas (o transacciones) que se ejecutaran en la Base de Datos una vez que se implemente. Una parte importante del diseño de Base de Datos es especificar las características funcionales de estas transacciones en una etapa temprana del diseño. Esto garantiza que el esquema de la Base de Datos incluirá toda la información requerida por dichas transacciones. Además, conocer la importancia relativa de las diversas transacciones y la frecuencia con que se espera invocarlas desempeña un papel crucial en el diseño fisico de la Base de Datos. Por lo regular, sólo se conocen algunas de las transacciones de la Base de Datos en el momento del diseño; una vez implementando el sistema se identificará e implementará continuamente nuevas transacciones. Sin embargo, es común que las transacciones más importantes se conozcan antes de la implementación del sistema, y deberán especificarse en una etapa temprana.

Una técnica común para especificar transacciones en un nivel conceptual es identificar su comportamiento de entrada / salida y funcional. Al especificar los datos de entrada, los datos de salida y el flujo de control interno, los diseñadores pueden especificar una transacción en una forma conceptual independiente del sistema. Por lo regular las transacciones pueden agruparse en tres categorías: transacciones de obtención, transacciones de actualización y transacciones mixtas. Las transacciones de obtención sirven para obtener datos para exhibirlos en pantalla o producir un informe. Las transacciones de actualización sirven para introducir datos nuevos o modificar datos que ya existen en la Base de Datos. Las transacciones mixtas se usan en aplicaciones más complejas que obtienen y actualizan datos.

Fase 3. Elección del Sistema de Gestión de Base de Datos.

La elección del Sistema de Gestión de Base de Datos depende de varios factores, algunos de ellos técnicos, otros económicos y otros mas relativos a las políticas de la organización. Los factores técnicos tienen que ver con la idoneidad del Sistema de Gestión de Base de Datos para la tarea en cuestión. Los que se deben considerar aqui son el tipo de Sistema de Gestión de Base de Datos(relacional, de red, jerárquico, orientado a objetos o de otra Base), las estructuras de almacenamiento y caminos de acceso que maneja el Sistema de Gestión de Base de Datos, las interfaces de usuario y programador disponibles, los tipos de lenguajes de consulta, etc. Los factores económicos y de organización que influyen en la elección del Sistema de Gestión de Base de Datos. Al escoger un Sistema de Gestión de Base de Datos, debemos considerar los siguientes costos:

· Costo de adquisición del software. Este es el gasto inicial que se hace al comprar el software, en el que se incluyen las opciones de lenguajes, diferentes interfaces como formas y pantallas, opciones de recuperación y respaldo, métodos de acceso especiales y documentación. Debe seleccionarse la versión correcta de Sistema del Gestión de Base de Datos para el sistema operativo especifico que se tiene.

· Costo de mantenimiento. Este es el costo recurrente por concepto del servicio de mantenimiento del proveedor y de la actualización regular de la versión del Sistema de Gestión de Base de Datos.

· Costo de adquisición de hardware. Tal vez se requiera más equipo, como memoria adicional, terminales, unidades de disco de almacenamiento especializado para el Sistema de Gestión de Base de Datos.

· Costo de creación y conversión de la Base de Datos. Éste es el costo de crear el sistema de Base de Datos desde cero o bien de convertir un sistema existente al nuevo software de Sistema de Gestión de Bases de Datos.

· Costo de personal. Cuando una organización adquiere por primera vez un software de Sistema de Gestión de Bases de Datos a menudo emprende un proceso de organización de su departamento de procesamiento de datos.

· Costo de capacitación. Como el Sistema de Gestión de Bases de Datos suelen ser sistemas complejos, casi siempre es preciso capacitar al personal para su uso y programación.

· Costo de operación. El costo de la operación continua del sistema de Base de Datos no suele incluirse en la evaluación de las alternativas porque es independiente del Sistema de Gestión de Bases de Datos que se seleccione.

.No es tan fácil medir y cuantificar los beneficios de adquirir un Sistema de Gestión de Bases de Datos. Un Sistema de Gestión de Bases de Datos tiene varias ventajas intangibles respecto a los sistemas de archivos tradicionales, como serían la facilidad de uso, la mayor disponibilidad de los datos y un acceso más rápido a la información. Entre los beneficios más tangibles están la reducción en el costo de crear aplicaciones, la menor redundancia de los datos y mejor control y seguridad

Fase 4. Transformación al modelo de datos (diseño lógico de la Base de Datos).

La siguiente fase del diseño de la Base de Datos consiste en crear un esquema conceptual y esquemas externos en el modelo de datos del Sistema de Gestión de Bases de Datos elegido. Esto se logra transformando los esquemas conceptual y externos producidos del modelo de datos conceptual al modelo de datos del Sistema de Gestión de Bases de Datos. La transformación puede establecerse en dos etapas:

Transformación independiente del sistema. En este paso, la transformación al modelo de datos del Sistema de Gestión de Bases de Datos no considera las características específicas o casos especiales que se aplican a la forma como el Sistema de Gestión de Bases de Datos implementa el modelo de datos.

Adaptación de los esquemas a un Sistema de Gestión de Bases de Datos. Los diferentes Sistemas de Gestión de Bases de Datos implementan un modelo de datos con características y restricciones de modelado específicas. Tal vez sea preciso ajustar los esquemas obtenidos en la etapa anterior para adaptarlos a las características de implementación específicas de un modelo de datos en el Sistema de Gestión de Bases de Datos seleccionado.

El resultado de esta fase debe construir en enunciados escritos en el lenguaje del Sistema de Gestión de Bases de Datos elegido que especifiquen los esquemas a nivel conceptual y externo del sistema de Base de Datos. Sin embargo, si los enunciados contienen algunos parámetros de diseño físico, la especificación completa de los enunciados deberá esperar hasta que se haya concluido la fase de diseño físico.

Fase 5. Diseño físico de la Base de Datos.

El diseño físico de la Base de Datos es el proceso de elegir estructuras de almacenamiento y caminos de acceso específicos para que los archivos de la Base de Datos tengan un buen rendimiento con las diversas aplicaciones de la Base de Datos. Cada Sistema de Gestión de Bases de Datos ofrece varias opciones de organización de archivos y caminos de acceso, entre ellas diversos tipos de indización, agrupamiento de registros relacionados en

bloques de disco, enlace de registros relacionados mediante apuntadores y varios tipos de técnicas de dispersión. Una vez seleccionado el Sistema de Gestión de Bases de Datos especifico, el proceso de diseño fisico se reduce a elegir las estructuras más apropiadas para los archivos de la Base de Datos entre las opciones que ofrece ese Sistema de Gestión de Bases de Datos. A menudo se utilizan los siguientes criterios para guiar la elección de las opciones de diseño fisico:

Tiempo de respuesta. Este es el tiempo que transcurre entre la introducción de una transacción de Base de Datos para ser ejecutada y la obtención de una respuesta. Un aspecto que influye mucho sobre el tiempo de respuesta y que esta bajo el control del Sistema de Gestión de Bases de Datos es el tiempo de acceso a la Base de Datos para obtener los elementos de información a los que hace referencia la transacción. El tiempo de respuesta también depende de factores que no puede controlar el Sistema de Gestión de Bases de Datos, como son la carga del sistema, la planificación de tareas del sistema operativo o los retrasos de comunicación.

Aprovechamiento del espacio. Esto se refiere a la cantidad de espacio de almacenamiento que ocupan los archivos de la Base de Datos y sus estructuras de acceso.

Productividad de las transacciones. Este es el número promedio de transacciones que el sistema de Base de Datos puede procesar por minuto; es un parámetro critico de los sistemas de transacciones como los que se usan para las reservaciones en lineas aéreas o el servicio en los bancos. La productividad de transacciones debe medirse en las condiciones pico del sistema.

Por lo general se especifican límites promedio y del peor de los casos para los parámetros anteriores como parte de los requerimientos de rendimiento del sistema. Se utilizan técnicas analiticas o experimentales, como la creación de prototipos y la simulación, para estimar los valores promedio y del peor de los casos suponiendo diferentes decisiones del diseño fisico, a fin de determinar si satisfacen los requerimientos de rendimiento especificados.

El rendimiento depende del tamaño y del número de registros que contienen los archivos; por ello, debemos estimar estos parámetros para cada archivo. Por añadidura, debemos estimar los patrones de actualización y obtención de datos para el archivo colectivamente a partir de todas las transacciones. Es recomendable construir caminos de acceso primarios e indices secundarios para los atributos con que se seleccionan los registros. También deben tenerse en cuenta durante la fase de diseño fisico las estimaciones del crecimiento de los archivos, sea en el tamaño de los registros debido a la adición de nuevos atributos o en el número de registros.

El resultado de la fase de diseño fisico es una determinación inicial de las estructuras de almacenamiento y los caminos de acceso para los archivos de la Base de Datos. Casi siempre es necesario afinar el diseño con base en su rendimiento observado después de la implementación del sistema. La mayor parte de los sistemas cuenta con una utileria de supervisión que reúne datos estadisticos de rendimiento, los cuales se conservan en el catálogo del sistema o en el diccionario de datos para su análisis posterior. Estos datos incluyen el número de invocaciones de transacciones o consultas predefinidas, actividades de entrada / salida de cada archivo, conteo de páginas de archivos o registros indice, y frecuencia de utilización de indices.

Fase 6. Implementación del sistema de Base de Datos.

Una vez completado los diseños lógico y fisico, se puede implementar la Base de Datos. Se compilan los enunciados escritos en el lenguaje del Sistema de Gestión de Bases de Datos seleccionado, y con ellos se crean los esquemas de la Base de Datos y sus archivos. En seguida ya puede cargarse la Base de Datos. Si los datos tienen que convertirse de un sistema computarizado antiguo, tal vez se requieran rutinas de conversión para modificar el formato de los datos y asi se puede almacenar en la nueva Base de Datos.

En esta etapa, los programadores de aplicaciones deben implementar las transacciones de la Base de Datos. Se examinan las especificaciones conceptuales de las transacciones, y se escribe y se prueba el código de programa correspondiente con ordenes de lenguaje de manipulación de datos incorporadas. Una vez que las transacciones estén listas y los datos se

hayan almacenado en la Base de Datos, la fase de diseño e implementación habrá terminado y se iniciará la fase de operación del sistema.

3.2 MODELO RELACIONAL

Una caracteristica fundamental de Bases de Datos es que proporciona cierto nivel de abstracción de los datos al ocultar detalles de almacenamiento que la mayoría de los usuarios no necesitan conocer. Los modelos de datos son el principal instrumento para ofrecer dicha abstracción.

Un modelo de datos es un conjunto de conceptos que pueden servir para describir la estructura de una Base de Datos.

Categorias de los modelos de datos. Los modelos de datos de alto nivel o conceptuales disponen de conceptos muy cercanos al modo como la generalidad de los usuarios percibe los datos, en tanto que en los modelos de bajo nivel o fisicos proporcionan conceptos que describen los detalles de cómo se almacenan los datos en el computador.

Los modelos de datos de alto nivel utilizan conceptos como entidades, atributos y vínculos. Una entidad representa un objeto o concepto del mundo real. Un atributo representa alguna propiedad de interés que da una descripción más amplia de una entidad. Un vinculo describe una interacción entre dos o más entidades.

Los modelos de datos de representación o de implementación son los más utilizados en los Sistemas de Gestión de Bases de Datos comerciales actuales, y entre ellos se cuentan los tres modelos más comunes: el **relacional**, el de **red** y el **jerárquico**.

Los modelos de datos fisicos describen cómo se almacenan los datos en el computador, al representar información como los formatos y ordenamientos de los registros y los caminos de

acceso. Un camino de acceso es una estructura que hace eficiente la búsqueda de registros específicos de la Base de Datos.

Para el estudio del modelo relacional es necesario introducirse en el modelo entidad - relación.

Modelo entidad - relación.

El objeto básico que se representa en el modelo entidad - relación, es la entidad: una cosa del mundo real con existencia independiente. Una entidad puede ser un objeto con existencia física una cierta persona, un automóvil, una casa o un empleado o un objeto con existencia conceptual, como una compañía, un puesto de trabajo o un curso universitario. Cada entidad tiene propiedades específicas, llamadas atributos, que la describen. Por ejemplo, una entidad empleado puede describirse por su nombre, su edad, su dirección, su salario y su puesto. Una entidad particular tendrá un valor para cada uno de sus atributos; los valores de los atributos que describen a cada entidad constituyen una parte decisiva de los datos almacenados en la Base de Datos.

En el modelo entidad relación se manejan varios tipos distintos de atributos. simples o compuestos; monovaluados o multivaluados, y almacenados o derivados.

Los atributos compuestos se pueden dividir en componentes más pequeños, que representan atributos más básicos con su propio significado independiente. Los atributos no divisibles se denominan atributos simples o atómicos.

En su mayoría, los atributos tienen un solo valor para una entidad en particular, y reciben el calificativo de monovaluados. Hay casos en los que un atributo puede tener un conjunto de valores para la misma entidad los atributos de este tipo se denominan multivaluados y pueden tener límites inferior y superior del número de valores para una entidad individual. En algunos casos la entidades pueden no tener valores cuando sucede esto se les llama valores nulos.

Atributos clave de un tipo de entidades. Una restricción importante de las entidades de un tipo es la restricción de clave o de unicidad de los atributos. Los tipos de entidades casi siempre tienen un atributo cuyo valor es distinto para cada entidad individual. Por ejemplo en los clientes de un banco el número de cuenta es único, no pueden tener un mismo número de cuenta dos o más clientes.

NUMERO DE CUENTA	CLIENTE
8548971132922091	200
4912950000054317	217
5290170458177390	125

Modelo relacional.

El modelo relacional de los datos fue introducido por Codd (1970). Se basa en una estructura de datos simple y uniforme (la relación) y tiene fundamentos teóricos sólidos. El modelo relacional se está estableciendo firmemente en el mundo de las aplicaciones de Bases de Datos.

El modelo relacional representa la Base de Datos como una colección de relaciones. En términos informales, cada relación semeja una tabla o, hasta cierto punto, un archivo simple.

Una relación es una tabla de valores, cada fila de la tabla representa una colección de valores de datos relacionados entre sí. Dichos valores se pueden interpretar como hechos que describen una entidad o un vinculo entre entidades del mundo real. El nombre de la tabla y los nombres de las columnas ayudan a interpretar el significado de los valores que están en cada fila de la tabla.

En la terminología del modelo relacional, una fila se denomina tupla, una cabecera de columna es un atributo y la tabla es una relación. El tipo de datos que describe los tipos de valores que pueden aparecer en cada columna se llama dominio.

Un dominio es un conjunto de valores atómicos. Por atómico queremos decir que cada valor del dominio es indivisible en lo tocante al modelo relacional. Un método común de especificación de los dominios consiste en especificar un tipo de datos al cual pertenecen los valores que constituyen el dominio.

Una clave candidata de una relación es un conjunto no vacio de atributos que identifican univoca y minimamente cada tupla. Por la propia definición de relación, siempre hay, al menos, una clave candidata, ya que al ser una relación un conjunto, no existe dos tuplas repetidas y, por tanto, el conjunto de todos los atributos identificará univocamente a las tuplas; si no se cumpliera la condición de minimalidad se eliminarán aquellos atributos que los impidiesen Una relación puede tener más de una clave candidata.

Clave primaria. es aquella clave candidata que el usuario escogerá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la relación.

Claves alternativas. son aquellas claves candidatas que no han sido escogidas como clave primaria.

Estructura del modelo relacional.

La relación es el elemento básico del modelo relacional, y se puede representar como una tabla. En ella podemos distinguir un conjunto de columnas, denominadas atributos, que representan propiedades de la misma y que están caracterizadas por un nombre, y un conjunto de filas llamadas tuplas, que son las ocurrencias de la relación. El número de filas de una relación se denomina cardinalidad, mientras que el número de columnas es el grado. Existen también dominios de donde los atributos toman sus valores.

Nombre

ATRIBUTO 1	ATRIBUTO 2	.....	ATRIBUTO n	
XXXXXXXXX	XXXXXXXXX	.....	XXXXXXXXX	tupla 1
XXXXXXXXX	XXXXXXXXX	.....	XXXXXXXXX	tupla 2
XXXXXXXXX	XXXXXXXXX	.....	XXXXXXXXX	.
XXXXXXXXX	XXXXXXXXX	.....	XXXXXXXXX	.
XXXXXXXXX	XXXXXXXXX	.....	XXXXXXXXX	tupla n

Representación del modelo relacional en una tabla
Figura 3.2

Una relación se puede representar en forma de tabla, aunque tiene una serie de elementos característicos que la distinguen de la tabla:

- No puede haber filas duplicadas, es decir, todas la tuplas tienen que ser distintas.
- El orden de las filas es irrelevante.
- La tabla es plana, es decir, en el cruce de una fila y una columna sólo puede haber un valor (no se admiten atributos multivaluados).

Una relación siempre tiene un nombre, y en ella es posible distinguir una cabecera que define la estructura de una tabla; es decir, sus atributos con los dominios subyacentes, y un cuerpo, extensión, que está formado por un conjunto de tuplas que varían en el tiempo.

Esta representación de la relación como una tabla ha sido el origen de que los productos relaciónales y los usuarios utilicen habitualmente el nombre de la tabla para denominar la relaciones y, como consecuencia de ello, se llame filas a las tuplas y columnas a los atributos.

Restricciones.

En el modelo relacional, existen restricciones es decir, estructuras u ocurrencias no permitidas, siendo preciso distinguir entre restricciones inherentes y restricciones de usuario.

Los datos almacenados en la Base han de adaptarse a las estructuras impuestas por el modelo y han de cumplir las restricciones de usuario para construir una ocurrencia válida del esquema.

En el modelo relacional cabe mencionar como restricciones inherentes, la integridad de entidad.

- No hay dos tuplas iguales.
- El orden de las tuplas no es significativo.
- El orden de los atributos no es significativo.
- Cada atributo sólo puede tomar un único valor del dominio, no admitiéndose por tato los grupos repetitivos.
- La integridad de la entidad establece que ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo.

Se considera la restricción del usuario, dentro del contexto relacional, como un predicado definido sobre un conjunto de atributos, tuplas o dominios, que debe ser verificado por los correspondientes objetos para que éstos constituyan una ocurrencia válida del esquema.

Dentro de las restricciones de usuario destaca la restricción de integridad referencial que se expresa de la siguiente manera: Si una relación R2 (relación que referencia) tiene un descriptor que es la clave primaria de otra relación R1 (relación referenciada), todo valor de dicho descriptor debe concordar con un valor de la clave primaria de R1 o ser nulo. El descriptor es por tanto una clave ajena de la otra relación R2.

Además de definir las claves ajenas, se debe determinar las consecuencias que pueden tener ciertas operaciones realizadas sobre tuplas de la relación referenciada; pudiéndose distinguir. Borrado y Modificación.

Operación restringida: esto es, el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada sólo se permite si no existen tuplas con dicha clave en la relación que contiene la clave ajena.

Operación con transmisión en cascada: esto es, el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada lleva consigo el borrado o modificación en cascada de las tuplas de la relación que contiene la clave ajena.

Operación con puesta de nulos: esto es, el borrado o la modificación de tuplas de la relación que contiene la clave primaria referenciada lleva consigo poner a nulos los valores de las claves ajenas de la relación que referencia.

Operación con puesta a valor por defecto: esto es, el borrado o la modificación de tuplas de la relación que contiene la clave primaria referencia la lleva consigo poner el valor por defecto a la clave ajena de la relación que referencia; valor por defecto que habria sido definido al crear la tabla correspondiente.

Operación que desencadena un procedimiento de usuario: en este caso el borrado o la modificación de tuplas de la tabla referenciada pone en marcha un procedimiento definido por el usuario.

3.3 IMPLEMENTACIÓN DE BASES DE DATOS RELACIÓNALES EN APLICACIONES CLIENTE - SERVIDOR

En una organización grande, el sistema de Base de Datos suele ser parte de un sistema de información mucho mayor con el que ella controla sus recursos de información. Un sistema de información incluye todos los recursos dentro de la organización que participan en la recolección, administración, uso y diseminación de la información. En un entorno computarizado, estos recursos incluirán los datos mismos, el Sistema de Gestión de Base de Datos, el hardware del computador y los medios de almacenamiento, el personal que usa y

manejа los datos, el software de aplicación que tiene acceso a los datos y los actualiza, y los programadores que crean estas aplicaciones. Así pues, el sistema de Base de Datos es sólo una parte de un sistema de información mucho mayor dentro de la organización.

A menudo se llama macro ciclo de vida al ciclo de vida del sistema de información, y micro ciclo de vida al ciclo de vida del sistema de Base de Datos. Por lo regular, el macro ciclo de vida incluye las siguientes fases:

Análisis de factibilidad. Éste se ocupa de analizar los posibilidades de analizar áreas de aplicación, realizar estudios preliminares de costo - beneficios y establecer prioridades entre las aplicaciones.

Recolección y análisis de requerimientos. Se obtienen requerimientos detallados interactuando con los posibles usuarios para identificar sus problemas y necesidades específicos.

Diseño. Esta fase tiene dos aspectos el diseño del sistema de Base de Datos y el diseño de los sistemas de aplicación que usan y procesan la Base de Datos.

Implementación. El sistema de información de implementa, la Base de Datos se carga y las transacciones de la Base de Datos se implementan y prueban.

Validación y prueba de aceptación. Se valida la aceptabilidad del sistema en cuanto a la satisfacción de los requerimientos de los usuarios y a los criterios de rendimiento. El sistema se prueba contra los criterios de rendimiento y las especificaciones de comportamiento.

Operación. Ésta puede ir precedida por la conversión de los usuarios de un sistema anterior así como por la capacitación de los usuarios. La fase operativa comienza cuando todas las funciones del sistema están disponibles y han sido validadas.

Las actividades del ciclo de vida del sistema de aplicación de Base de Datos son las siguientes:

Definición del sistema. Se definen el alcance del sistema de Base de Datos, sus usuarios y sus aplicaciones.

Diseño. Al final de esta fase, estará listo un diseño lógico y físico completo del sistema de Base de Datos en el Sistema de Gestión de Base de Datos elegido.

Implementación. esto comprende el proceso de escribir las definiciones conceptual, externa e interna de la Base de Datos, crear archivos de Base de Datos vacios e implementar las aplicaciones de software.

Carga o conversión de los datos. la Base de Datos se alimenta ya sea cargando los datos directamente o convirtiendo archivos ya existentes al formato del sistema de Base de Datos.

Conversión de aplicaciones. Cualesquier aplicaciones de software que se usaban en un sistema anterior se convierten al nuevo sistema.

Prueba y validación. Se prueba y valida el nuevo sistema.

Operación. El sistema de Base de Datos y sus aplicaciones se ponen en operación.

Supervisión y mantenimiento. Durante la fase de operación, el sistema se vigila y mantiene constantemente. Puede haber crecimiento y expansión tanto en el contenido de datos como en las aplicaciones de software. Es posible que de vez en cuando se requieran modificaciones y reorganizaciones importantes.

Entre las actividades relacionadas con el ciclo de vida del sistema de aplicación para la Base de Datos están las siguientes fases:

El modelo Cliente - Servidor se creó para manejar los nuevos entornos de computo en los que un gran número de computadores personales, estaciones de trabajo, servidores de archivos, impresoras y otros equipos que están conectados a través de una red. La idea es definir servidores especializados con funciones especificas. Por ejemplo, es posible conectar como clientes varias estaciones de trabajo o cumputadores personales sin disco a una máquina servidor de archivos que mantenga los archivos de los usuarios de los clientes

El modelo Cliente - Servidor se está incorporado cada vez más en los paquetes de Sistema de Gestión de Base de Datos comerciales. la idea consiste en dividir el software del Sistema de Gestión de Base de Datos en dos niveles Cliente y Servidor para reducir su complejidad. Algunos sitios pueden ejecutar exclusivamente el software del Cliente mientras que otros sitios podrían ser máquinas servidoras dedicadas que ejecutan solo el software de Servidor.

Todavia no se ha establecido una forma precisa de dividir la funcionalidad del Sistema de Gestión de Base de Datos entre el Cliente y el Servidor, y se han propuesto diferentes enfoques. Una posibilidad consiste en incluir la funcionalidad de un Sistema de Gestión de Base de Datos centralizado en el nivel de Servidor. Varios productos de Sistemas de Gestión de Base de Datos relaciónales han adoptado este enfoque, en el que se proporciona un Servidor SQL a los Clientes. Cada Cliente debe entonces formular las consultas SQL apropiadas y proveer la interfaz con el usuario y las funciones de interfaz con los lenguajes de programación. Puesto que SQL es una norma relacional, diversos servidores SQL diferentes, posiblemente provenientes de distintos proveedores, pueden aceptar ordenes SQL. El Cliente también puede hacer referencia a un diccionario de datos que tenga información sobre la distribución de los datos entre los diferentes servidores SQL, asi como módulos para descomponer una consulta global en varias consultas locales que se pueden ejecutar en los diferentes sitios. La interacción entre el Cliente y el Servidor podría efectuarse como sigue durante el procesamiento de una consulta SQL:

1. El Cliente analiza sintácticamente la consulta del usuario y la descompone en varias consultas de un sitio independiente. Cada consulta de sitio se envia al sitio Servidor apropiado.

2. Cada Servidor procesa la consulta local y envia la relación resultante al sitio del Cliente.

3. El sitio Cliente combina los resultados de las subconsultas para producir el resultado de la consulta original.

En este enfoque, el Servidor SQL recibe también el nombre de procesador de Bases de Datos o máquina dorsal. El usuario puede especificar la interacción entre el Cliente y el Servidor en el nivel de Cliente o a través de un modulo Cliente especializado. Por ejemplo, el usuario podria saber qué datos están almacenados en cada Servidor, descomponer manualmente una consulta en subconsultas de sitios y presentar subconsultas individuales a los diferentes sitios. Las tablas resultantes podrian combinarse explícitamente mediante otra consulta del usuario en el nivel de Cliente. La alternativa es hacer que el modulo Cliente lleve a cabo estas acciones automáticamente.

CAPITULO 4

LA IMPLEMENTACIÓN DE UNA BASE DE DATOS RELACIONAL EN EL SISTEMA DE OPERACIONES INTERNACIONALES DEL GRUPO FINANCIERO INBURSA

4.1 VISUAL BASIC

Visual Basic es un lenguaje de programación que sirve para la creación de aplicaciones en ambiente Windows. Este lenguaje de programación es una herramienta para crear aplicaciones gráficas con caracteristicas de objetos, menús, ventanas, soporte del ratón (Mouse) e iconos; todos estos elementos pueden hallarse en la mayoria de las aplicaciones Windows.

En el mundo actual de los negocios de competencia creciente, complejidad y demandas, los desarrolladores están constantemente buscando mejores herramientas y técnicas con el fin de proporcionar soluciones Cliente - Servidor de forma rápida y eficiente para ambientes de computo distribuido. Con ambientes de desarrollo gráfico para aplicaciones "front - end" (cliente) tales como Visual Basic, con esta herramienta ahora es muy fácil crear aplicaciones Cliente - Servidor.

Por medio del lenguaje Visual Basic es posible realizar una interfaz entre el lenguaje y un manejador de Bases de Datos (en este caso SQL Server). Existen varias herramientas para realizar estas interfaces por ejemplo DB-Library y ODBC (Open Data Base Connectivity). DB-Library provee una conexión directa con el manejador de Bases de Datos por medio de bibliotecas (procedimientos), mientras que ODBC provee interfaces a manejadores de sesiones que a su vez llevan a cabo la conexión con el manejador de Base de Datos.

En el presente caso práctico se utiliza la interfaz por medio de ODBC.

4.2 SQL SERVER

SQL Server es un sistema manejador de Bases de Datos Relacional inteligente, basado en la arquitectura Cliente - Servidor, que soporta aplicaciones de procesamiento de transacciones en linea y de soporte en la toma de decisiones.

La seguridad y garantia del procesamiento de transacciones en SQL Server lo hacen ideal para las más exigentes aplicaciones de misión critica. Las capacidades avanzadas de SQL Server tales como: Triggers, Procedimientos Almacenados, Reglas y Omisiones, y su tecnologia de llamado a procedimiento remoto hacen de SQL Server la herramienta ideal para el desarrollo de aplicaciones dorsales (back-ends) en estaciones de trabajo, minicomputadoras y mainframes. Su arquitectura diseñada para el alto rendimiento, un solo proceso, multi-tarea, asegura un rendimiento consistente y estable que se mantiene aún cuando el número de usuarios aumenta.

Muchas organizaciones están descubriéndose asi mismas como dependientes del manejo de la información para las más fundamentales operaciones del negocio asi como la toma de decisiones estratégicas. Procesamiento de pedidos, manejo de inventarios, contabilidad y otro sin número de aplicaciones son vitales para el desempeño sano de la vida de la organización, por lo que es incuestionable la seguridad, disponibilidad y veracidad de la información.

SQL Server ofrece un soporte sin paralelo de procesamiento de transacciones demandado por las muy complejas aplicaciones de hoy en dia y los ambientes de negocios:

Administración remota de los recursos del sistema.

Un completo y robusto manejo de transacciones con capacidad de rollforward y roll-back que permite la recuperación automática de transacciones al reinicio del sistema.

Capacidad para generación automática de copias de seguridad.

La ejecución de procedimientos almacenados en forma remota y procesamiento de transacciones en dos fases permiten la distribución de la información en servidores geográficamente dispersos.

Un robusto soporte de tolerancia a fallas.

El reforzamiento centralizado de la integridad de los datos minimiza el mantenimiento, programación y reduce los costos asegurando que los datos son correctos.

Interoperabilidad dinámica con la red local.

Una arquitectura de un solo proceso multitareas asegura un desempeño del sistema eficiente para un gran número de usuarios con poco consumo de recursos.

Interfaces de programación remota.

SQL Server ha ganado repetidamente los premios como el mejor manejador de Bases de Datos en arquitectura Cliente - Servidor para Redes Locales, en adición a otros premios de la industria. El SQL Server es la opción del manejo de Bases de Datos Relaciónales más flexible y poderosa del mercado.

4.3 GRUPO FINANCIERO INBURSA

Institución que consta de Banca Múltiple, Compañia Aseguradora, Casa de Bolsa y Empresas arrendadoras y de factoraje. Algunas funciones principales son las de otorgar créditos a personas físicas y morales cobrando un interés por el crédito, realizar transacciones de recursos, cambio de moneda extranjera, compra venta de remesas, envio de giros, entre otras.

Para el registro y manejo de la información de las operaciones antes mencionadas se tuvo la necesidad de realizar un sistema computacional en el cual se implementa una Base de Datos. La cual es tema del presente trabajo.

4.4 SOI

Sistema de Operaciones Internacionales.

En el Grupo Financiero Inbursa existe un sistema de información llamado Sistema de Operaciones Internacionales, el cual tiene la finalidad de tener un control de la información relacionada con operaciones de contado y crédito que realiza el Grupo Financiero (Figura 4.1).

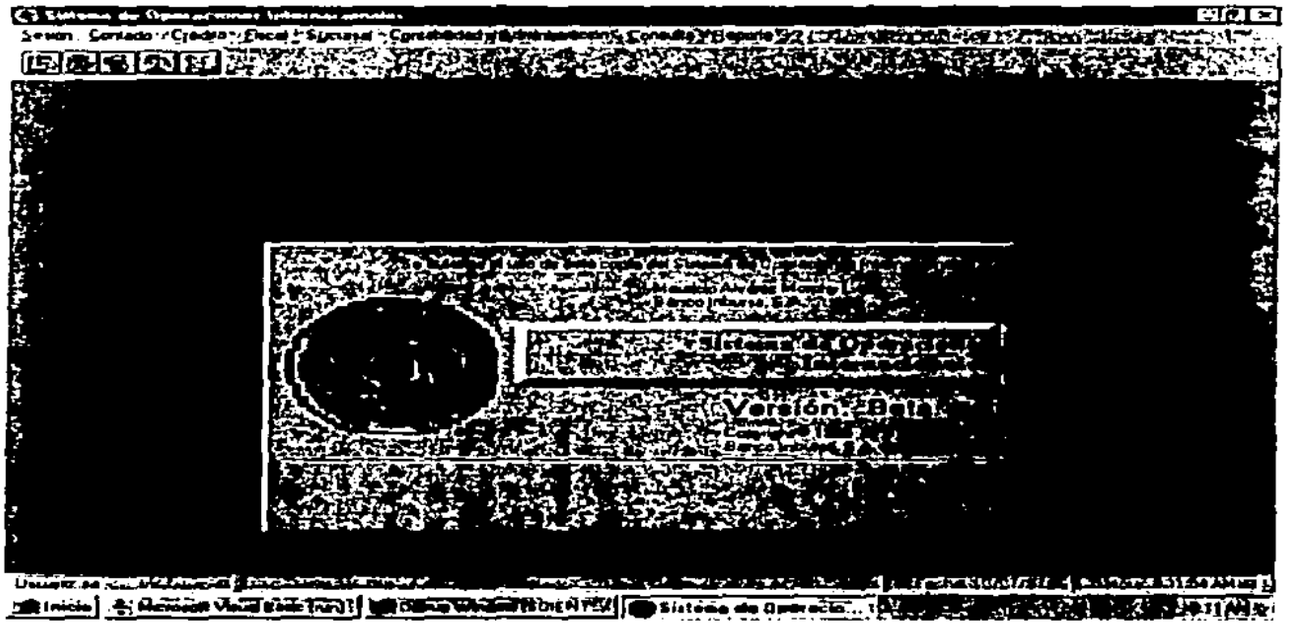


Figura 4.1 Pantalla principal del Sistema de Operaciones Internacionales

El sistema consta de cuatro módulos principalmente:

- Operaciones de Contado.
- Operaciones de Crédito.
- Contabilidad.
- Administración.

El módulo de Operaciones de Contado.

Este módulo es el encargado de realizar manipulaciones de ordenes de compra-venta de los productos que se manejan en el Grupo Financiero esta manipulación se refiere a altas, modificaciones, consultas y bajas, en operaciones de contado (Figura 4.2).

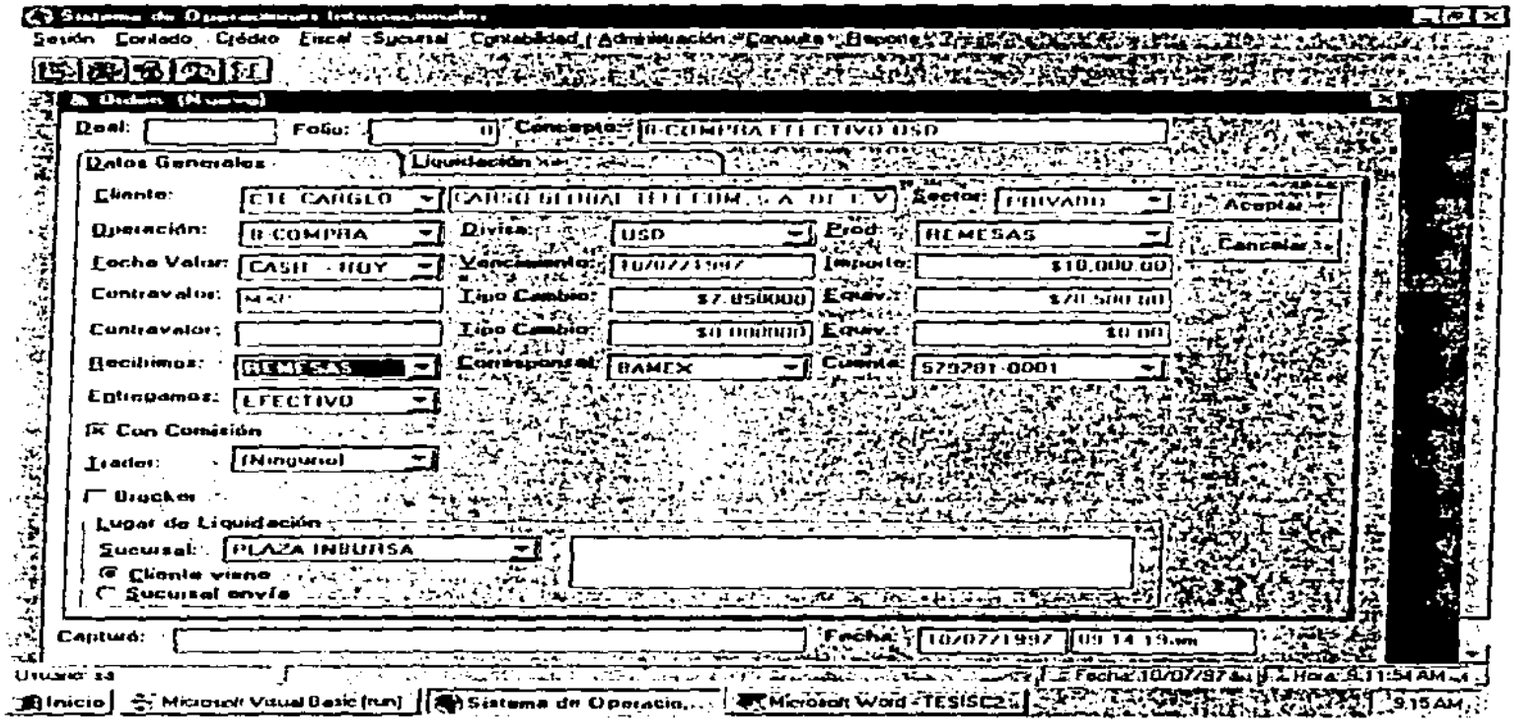


Figura 4.2 Pantalla donde se realiza la orden de una compra de remesas

Los productos que se manejan en este módulo de operaciones de contado son:

Cheques de viajero. De los cuales se realizan instrucciones de transferencia, inventario de los cheques de viajero, solicitud para los mismos y reporte de seguimiento de cheques de viajero. Los cheques de viajero son documentos que representan dinero los cuales son adquiridos por personas ya sea físicas o morales en cierta parte del mundo y pueden ser cambiados en el lugar donde va ha viajar.

Giros. Sobre los giros de dinero se realizan compensación de giros que es cuando se realiza la venta de giros, descompensación cuando son cobrados esos giros, verificación de si se encuentran sin haberse realizado el cobro de los giros (llamado stop payment) en México y en el extranjero. En estos tiempos es muy común la realización de giros; esto es cuando una persona manda recursos (dinero) a otra en un lugar distinto al de él.

Remesas. Las remesas son un conjunto de documentos que alguna persona las lleva a vender al banco para que este realice el cobro de los mismos. En el sistema de operaciones internacionales se realiza el proceso de información de compensación de remesas, descompensación, consultas y modificaciones de las mismas.

Transferencias. La transferencia de fondos de una cuenta a otra dentro y fuera del pais, transferencia de valores de un banco a otro en todo el mundo. En las transferencias se realizan operaciones de confirmación, instrucciones, consultas, altas y modificaciones.

Ejemplo de las operaciones de contado en la figura 4.3

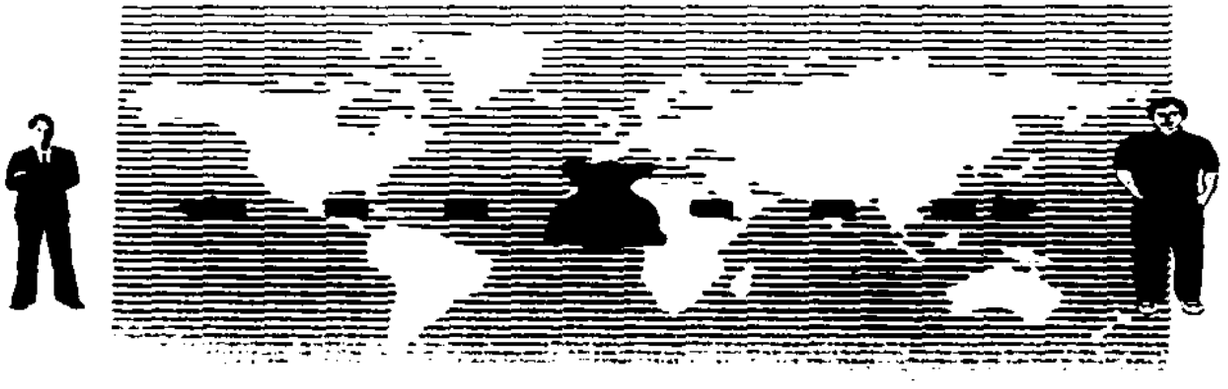


Figura 4.3 Giro de valores

El módulo de Operaciones de Crédito.

En este módulo se realizan los registros de las operaciones que realiza el grupo financiero para poder realizar prestamos a clientes del mismo, y registro de donde obtiene los

recursos para poder prestarlo (Figura 4.4), en este módulo básicamente se realizan dos operaciones básicas que son :

Captaciones. Esta operación es la de obtener recursos con intereses bajos para prestarlos con intereses altos y así obtener ganancias.

Colocaciones. Aquí se registran las operaciones para poder realizar el préstamo de los recursos.

En ambas operaciones se registran plazos de pago de capital e interés, registro de comisiones forma de pago, etc. (Figura 4.5).

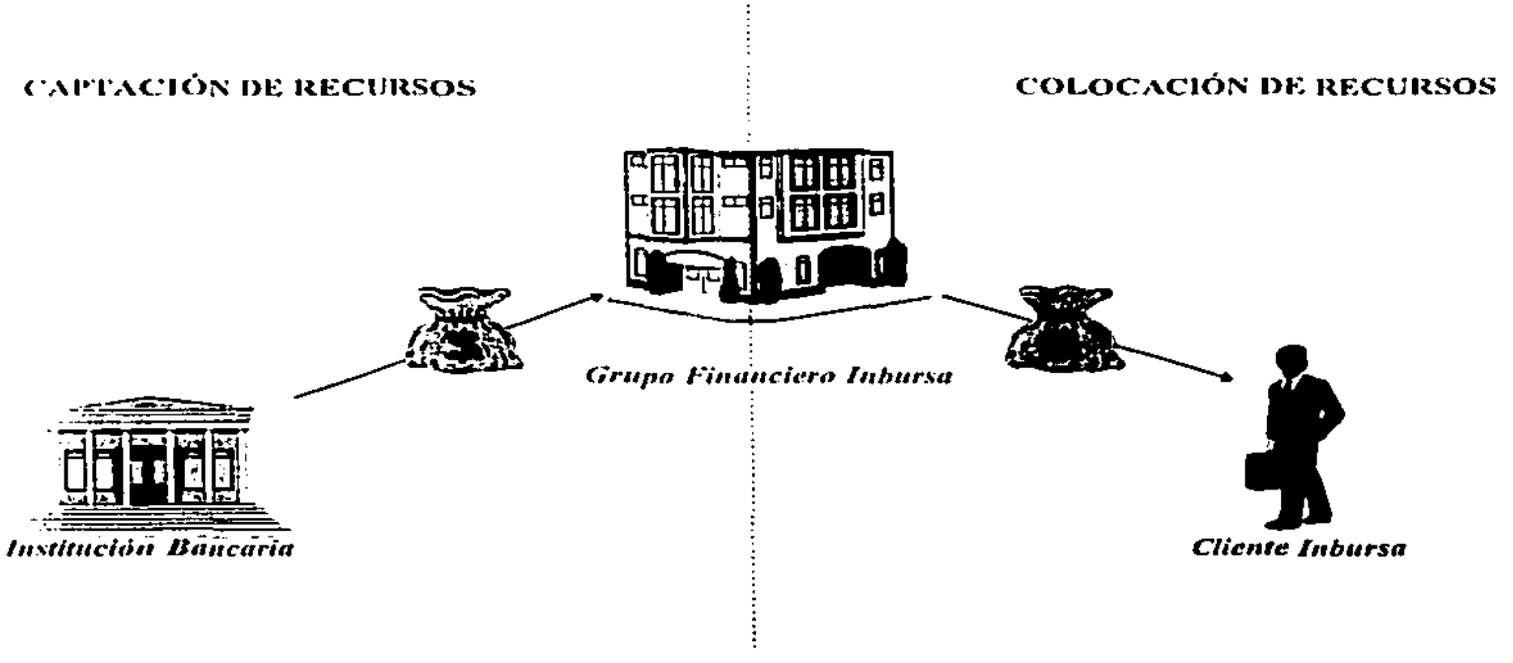


Figura 4.4 Operaciones de Crédito

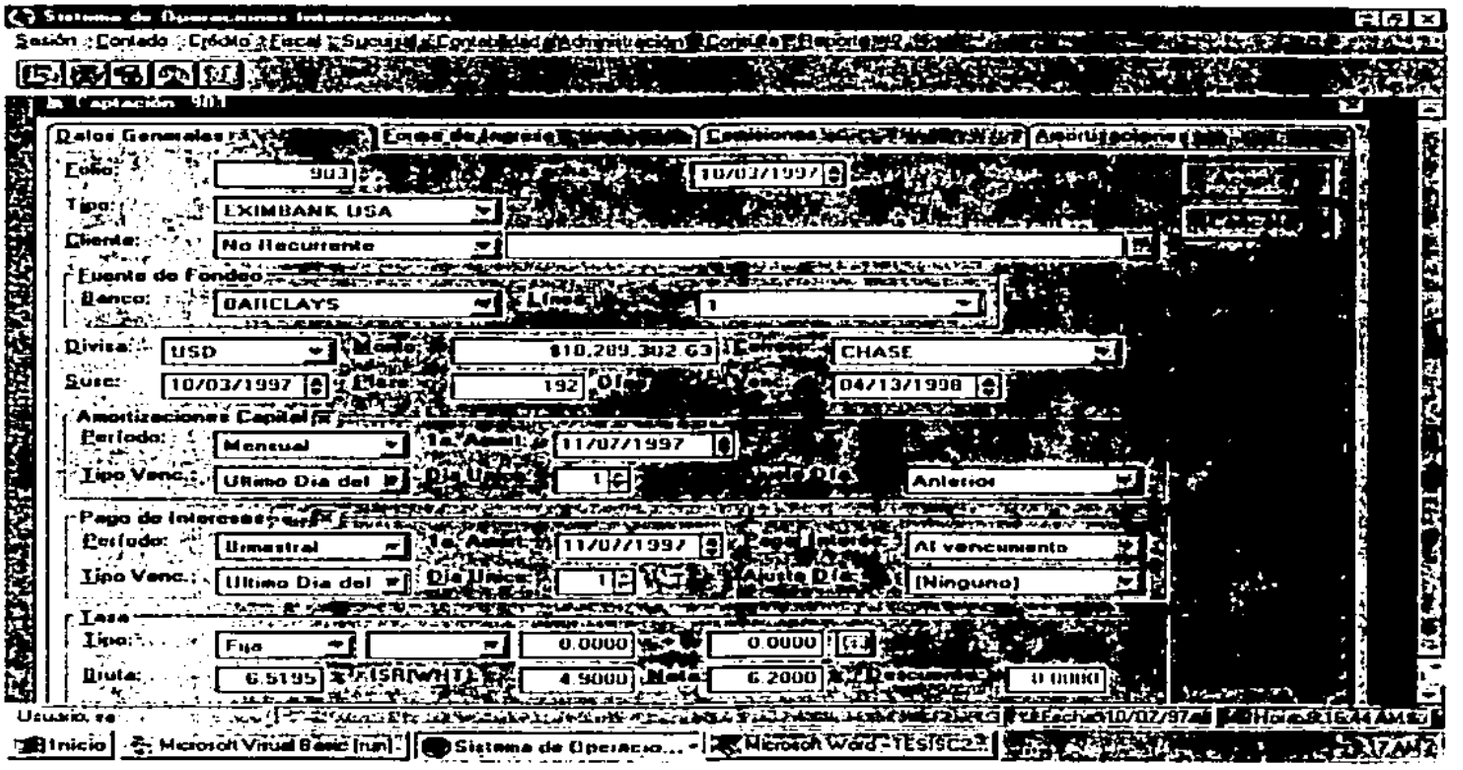


Figura 4.5 Pantalla para realizar una captación de recursos

El módulo de Contabilidad.

En este módulo se realiza el registro de cuentas contables las cuales se van a afectar en las operaciones de contado y de crédito (Figura 4.6), consulta de movimientos diarios por cuenta, presenta las cuentas que fueron afectadas con las operaciones realizadas; registro de relaciones de cliente - cuenta, los clientes de Inbursa que son recurrentes tienen alguna cuenta; relación de evento - cuenta, en la compra o venta de productos se afectan ciertas cuentas; y relación de producto - cuenta, las cuentas que se deben afectar en relación a los productos vendidos o comprados. El módulo de contabilidad es dependiente de los módulos de contado y de crédito, ya que la operaciones de estos generan movimientos que alimentan al módulo de contabilidad.

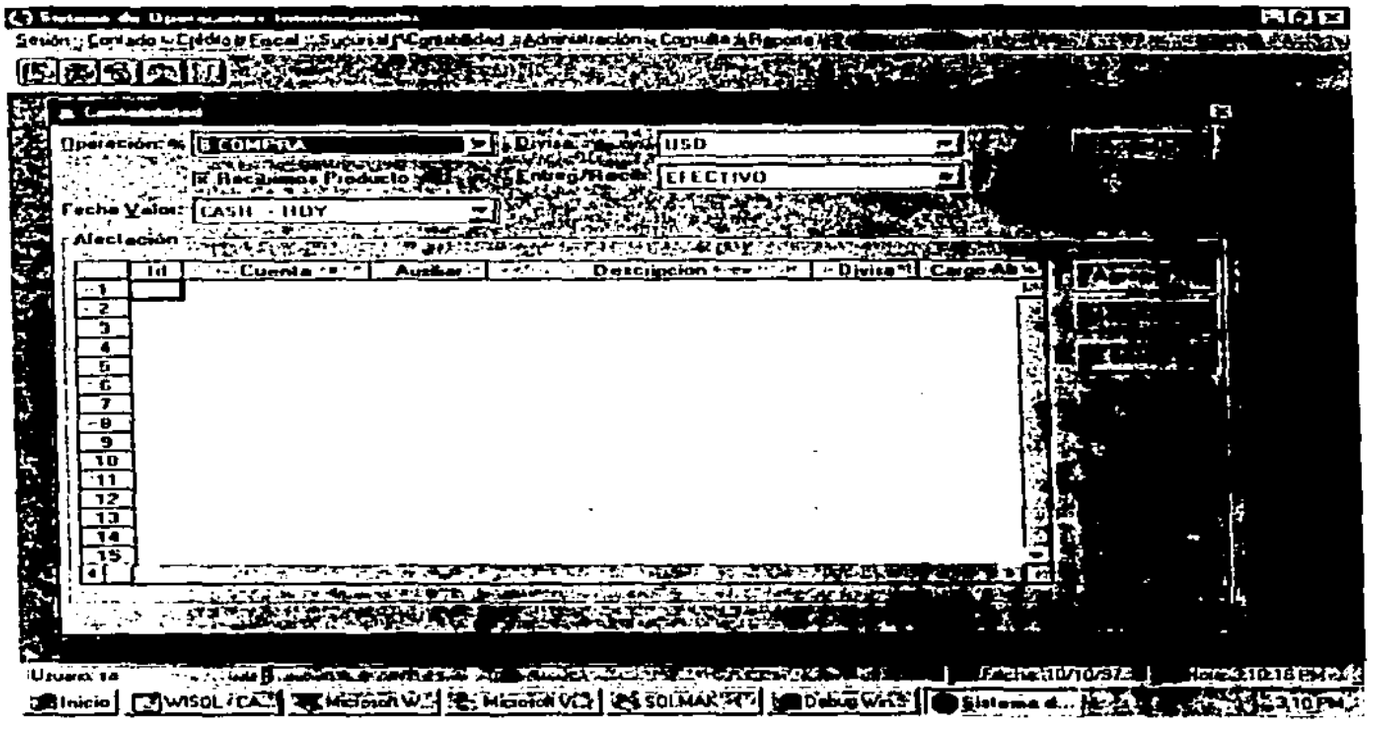


Figura 4.6 Pantalla de relación de cuentas con operaciones de crédito y contado

El módulo de Administración.

Este módulo sirve para realizar la administración correcta del sistema, se maneja un calendario de días laborables, catálogos de clientes, de cuentas, de productos, de sucursales, cierre contable diario, posición al inicio y al final del día en cuanto a contabilidad, la seguridad (Figura 4.7) y tipo de cambio del día en diferentes monedas manejadas en el mundo. Este módulo tiene dos funciones la primera es la administración del proyecto, y la segunda que es el manejo de catálogos para la funcionalidad del resto de los módulos.

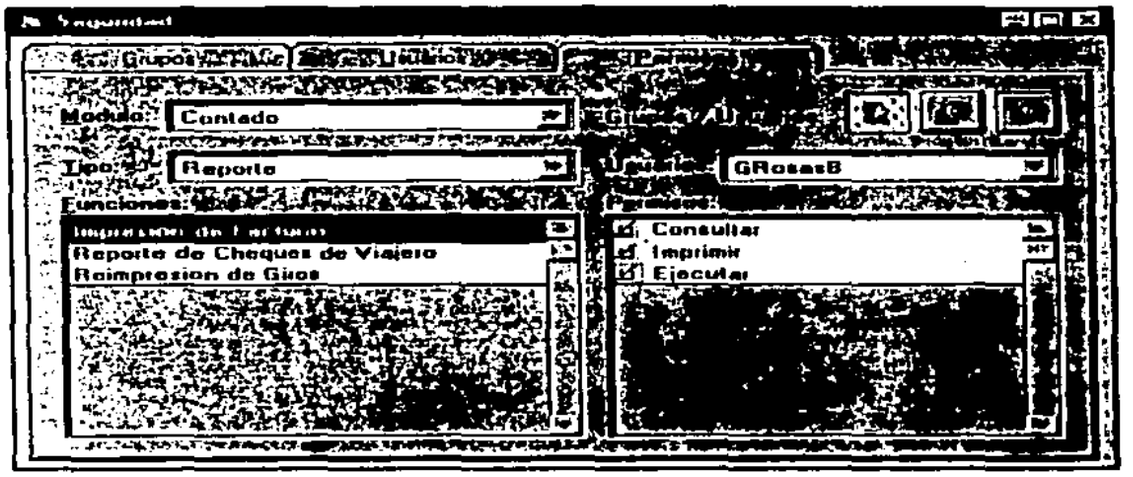


Figura 4.7 Pantalla para la seguridad del sistema

4.5 REQUERIMIENTOS PARA EL DESARROLLO

El concepto de manejar a un grupo de desarrollo envuelve requerimientos físicos y de documentación, así como una serie de pasos a seguir en la organización y planeación del proyecto para que sea llevado a su terminación de manera exitosa.

Las necesidades físicas que deben ser satisfechas antes del arranque del proyecto con el propósito de llevar a cabo un desarrollo efectivo por el grupo encargado son las siguientes:

- La red de trabajo.
- Estrategia del control de versiones.
- Estándares de interface y funcionalidad.

La red de trabajo. Si no se cuenta con una conexión en red para los desarrolladores del grupo, el tiempo invertido en copiar el código entre las computadoras y la revisión coordinada de esta actividad devorará una significativa parte del tiempo de duración del proyecto. Para la implementación de la Base de Datos Relacional en el grupo Financiero Inbursa se utiliza una Red de Área Local.

Control de Versiones. A lo largo del desarrollo de una aplicación, es conveniente realizar la generación periódica de versiones tanto de la aplicación como de la Base de Datos con el objetivo de permitir a los demás departamentos o áreas involucradas en el proyecto, dar seguimiento a la evolución del sistema.

Estándares. La presencia de estándares tienen ciertas ventajas:

Al implementar una Base de Datos el diseñador puede familiarizarse rápidamente con la operación de las partes integrales de la aplicación. Una vez familiarizado puede predecir el comportamiento de nuevos módulos añadidos a una aplicación. Un conjunto de aplicaciones de una misma organización con interfaces homogéneas adquieren una personalidad muy funcional

Requerimientos de software. Se requiere para el desarrollo del Sistema de Operaciones Internacionales. Se requiere de un manejador de Bases de Datos en este caso SQL Server, El lenguaje de programación Visual Basic, Controlador de versiones Visual Source Safe y Diseñador de Bases de Datos Relaciónales "Erwin".

Requerimientos de hardware. Se utilizó un servidor de Bases de Datos, terminales con procesador pentium, 16 MB de memoria RAM conectadas en red.

4.6 TÓPICOS REFERENTES AL EQUIPO DE DESARROLLO

Representación del usuario.

Es bastante útil contar con la representación del usuario en un equipo de desarrollo. Esta persona puede ser un analista del negocio al que pertenece la aplicación. La representación del usuario, es entendida para labores de consultoria sobre el negocio, y sus sugerencias son satisfechas en el contexto de la tecnología disponible.

Control de Calidad.

En un ambiente de desarrollo ideal, un desarrollador encargado de realizar pruebas (tester) es asignado por cada desarrollador encargado de codificar. Esta persona es responsable

de realizar pruebas unitarias, de volumen y aquellas requeridas bajo circunstancias especiales, como puede ser la presencia de baja memoria o pocos recursos.

4.7 ORGANIZACIÓN Y ARQUITECTURA DEL PROYECTO

La organización del proyecto debe facilitar el rápido desarrollo de la aplicación. Cada equipo debe trabajar solo en un nivel especifico de la aplicación.

Es recomendable dividir la arquitectura y estructura del sistema en los siguientes tres niveles de servicio:

Nivel de servicios del usuario.

Este nivel incluye la interfaz del usuario. las funciones propias de este nivel no deben contener referencia directa a la Base de Datos. Este punto es critico para el desarrollo de un sistema bien estructurado.

Nivel de servicios del negocio.

Este nivel contiene las reglas y la lógica del negocio. Las decisiones basadas en la información proporcionada por el usuario son procesadas aqui. Este nivel es responsable de la generación de información para mostrar al usuario asi como organizar las requisiciones realizadas al nivel de servicio de datos.

Nivel de servicios para los datos.

Este nivel proporciona información en los requerimientos del nivel de servicios del negocio, incluye cualquier información de Base de Datos local o distribuida.

4.8 IMPLEMENTACIÓN

Para cubrir las necesidades del registro de la información para los módulos anteriormente explicados con los que cuenta el sistema, es necesario implementar una Base de Datos Relacional.

La implementación comienza ya que se realizó el diseño de la Base de Datos, creándola en el servidor. Para el sistema se creó una base de un tamaño de 30 Mega Bytes. Después se crean la tablas con las relaciones hacia las tablas con las cuales va a interactuar

Para la implementación se utilizan sentencias del lenguaje de SQL como:

- Selección. Select * from Cliente
- Inserción. Insert Cliente values(1,Carlos,23)
- Actualización. Update Cliente set Cliente_Nombre = "Carlos"
- Borrado. Delete Cliente
- Inicio de Transacciones. Begin Tran Inserta_Capta
- Confirmación de la transacción. Commit Tran Inserta_Capta
- Retroceso de la transacción. Rollback Tran Inserta_Capta

La Base de Datos está dividida en cuatro partes que son las que corresponden a los módulos del sistema.

En el módulo de contado las operaciones de contado se deben registrar en una forma (pantalla) llamada orden en la cual se registran el cliente, el sector del cliente, la operación (compra o venta), la divisa, el producto, el plazo, la fecha de vencimiento, el monto, el tipo de cambio, el equivalente conforme al tipo de cambio, el producto que se entrega y el que se recibe, la sucursal donde se debe de liquidar y en su caso el corresponsal para la operación. Al realizar la liquidación de la operación se calcula una comisión y se registra. Se requieren principalmente las tablas de Cliente, Corresponsal, Productos, Tipo de Cambio, Usuario,

Cuentas de los clientes en Inbursa, Orden (Operaciones), Transferencias, Remesas, Cheques de Viajero, Efectivo y Giros. De estas tablas se derivan otras con las cuales interactuan para obtener o registrar información relacionada.

A continuación se muestran algunos procedimientos que se utilizan para la realización de la orden de compra venta de productos en el módulo de contado.

El siguiente procedimiento (stor procedure) realizado en SQL realiza la consulta de las ordenes de compra - venta de productos. Aqui se muestra se relacionan las tablas, en el lenguaje SQL

```
CREATE PROCEDURE spS_SOI_Load_Orden (@viOrden_Id int)
AS
BEGIN

        SELECT  A.Orden_Deal                            "Deal",         A.Orden_Status_Id                       "Status_Id",
                A.Operacion_Tipo_Id             "Operacion_Id",  B.Operacion_Tipo_Dsc                "Operacion_Dsc",
                A.CorrespRecibe_Id                      "CorrespRec_Id", J.Corresp_Nom_Corto                 "CorrespRec_Dsc",
                ISNULL(A.CorrespRecibe_Cta, 0)          "CuentaRec_Id", A.CorrespEntrega_Id                  "CorrespEnt_Id",
                K.Corresp_Razon_Soc                     "CorrespEnt_Dsc", ISNULL(A.CorrespEntrega_Cta, 0) "CuentaEnt_Id",
                A.Divisa_Id                             "Divisa_Id",    D.Divisa_Contravalor                 "Divisa_Con",
                D.Divisa_Siglas                         "Divisa_Dsc",   A.Producto_Id                        "Producto_Id",
                G.Producto_Dsc                          "Producto_Dsc",A.Fecha_Val_Tipo_Id                   "Fecha_Id",
                C.Fecha_Val_Tipo_Dsc                    "Fecha_Dsc",    A.Orden_Fecha_Valor                  "Fecha_Valor",
                A.Orden_Importe                         "Importe",      ISNULL(E.Divisa_Siglas,"")"Divisa_Contra",
                A.Orden_Tipo_Cambio                     "Tipo_Cambio",A.Orden_Tipo_Cambio_Mxn    "Tipo_Cambio_Mxn",
                A.Recibimos_Id                          "Recibimos_Id",H.Producto_Dsc           "Recibimos_Dsc",
                A.Entregamos_Id                         "Entregamos_Id",I.Producto_Dsc          "Entregamos_Dsc",
                A.Orden_Comis_Por_Cliente               "Comision_Cliente",  A.Orden_Liquida_Viene          "Viene_Cliente",
                A.Sucursal_Liquida                      "Sucursal_Id",  F.Suc_Nombre                         "Sucursal_Dsc",
                A.Emisor_Id                             "Emisor_Id",    L.Emisor_Dsc                         "Emisor_Dsc",
                ISNULL(A.Orden_Liquida_Instr,"")        "Instrucciones", A.Orden_Fecha_Captura               "Fecha_Captura",
                A.NumTransa_SIBI                        "NumTransa"
        FROM    Orden A, Operacion_Tipo B, Fecha_Valor_Tipo C, Divisa D, Divisa E, Sucursal F, Producto G, Producto H,
                Producto I, Corresponsal J, Corresponsal K, Emisor L
        WHERE   A.Operacion_Tipo_Id = B.Operacion_Tipo_Id
        AND     A.Fecha_Val_Tipo_Id             = C.Fecha_Val_Tipo_Id
        AND     A.Divisa_Id                     = D.Divisa_Id
        AND     D.Divisa_Contravalor *= E.Divisa_Id
        AND     A.Sucursal_Liquida      = F.Suc_Id
        AND     A.Producto_Id                   = G.Producto_Id
        AND     A.Recibimos_Id                  = H.Producto_Id
        AND     A.Entregamos_Id                 = I.Producto_Id
        AND     A.CorrespRecibe_Id      *= J.Corresp_Id
        AND     A.CorrespEntrega_Id     *= K.Corresp_Id
        AND     A.Emisor_Id                     *= L.Emisor_Id
        AND     A.Orden_Id                      = @viOrden_Id
RETURN(@@Error)
END
```

El siguiente procedimiento (stor procedure) realizado en SQL realiza la actualización o la inserción en caso de que la orden no exista de las ordenes de compra - venta de productos.

```
CREATE PROCEDURE spI_SOI_New_Mod_Orden (@viOrden_Id          int,        @viEmisor_Id        identificador,
```

```
                                @vsDeal          varchar(10), @viStatus_Id        identificador,
                                @viRecurrente    identificador, @viCliente_Id     identificador,
                                @viCliente_Abono identificador, @vsCliente        varchar(50),
                                @viSector_Id     identificador, @viOperacion_Id   identificador,
                                @viTipoFecha_Id  identificador, @vdFecha_Valor    datetime,
                                @viTipo_Cambio   float,         @viTipo_Cambio_Mxn float,
                                @viDivisa_Id     identificador, @viOrden_Importe  monto,
                                @viProducto_Id   identificador, @viRecibimos_Id   identificador,
                                @viCorrespRec_Id identificador, @viCta_Recibida   identificador,
                                @viEntregamos_Id identificador, @viCorrespEnt_Id  identificador,
                                @viCta_Entregada identificador, @vsBeneficiario   varchar(50),
                                @viCostis_Cliente identificador, @viLiquida_Viene identificador,
                                @viSucursal_Liq  identificador, @viSuc_Id         identificador,
                                @viPromotor_Id   identificador, @viOrden_Broker   identificador,
                                @viUsuario_Id    int)

AS
BEGIN
        /*          Declaracion de constantes                   */
        DECLARE @NULL_INTEGER int
        DECLARE @NULL_STRING        char(1)
        DECLARE @DEAL_EXISTE        int
        DECLARE @OK                 int
        DECLARE @INSERTAR           int
        DECLARE @ORDENADO           int
        DECLARE @GIRO_INICIAL       int
        DECLARE @EN_SUCURSAL        int
        DECLARE @VENTAS             int
        DECLARE @CHVIAJERO          int
        DECLARE @GIROS              int
        DECLARE @REMESAS            int
        DECLARE @ABONO_CTA          int
        DECLARE @CARGO_CTA          int
        DECLARE @TRANSFERENCIA      int
        DECLARE @SIAC               int
        DECLARE @SPEUA          int
        DECLARE @TITULAR            int
        DECLARE @RESULT             varchar(5)
        DECLARE @CIERTO             int

        /*          Declaracion de variables                    */
        DECLARE @iError         int
        DECLARE @iCliente_Id    int
        DECLARE @iRecibimos_Id  int
        DECLARE @iEntregamos_Id int
        DECLARE @iMonto         money
        DECLARE @iOrden_Id      int
        DECLARE @iMontoRem      money
        DECLARE @iMontoTrans    money

        /*          Asignación de constantes                    */
        SELECT @NULL_INTEGER    = 0
        SELECT @NULL_STRING     = ""
        SELECT @OK              = 0
        SELECT @INSERTAR        = 1
        SELECT @ORDENADO        = 1
        SELECT @GIRO_INICIAL    = 1
        SELECT @EN_SUCURSAL     = 1
        SELECT @VENTAS          = 2
        SELECT @CHVIAJERO       = 2
        SELECT @GIROS           = 3
        SELECT @REMESAS         = 4
        SELECT @ABONO_CTA       = 6
        SELECT @CARGO_CTA       = 7
        SELECT @TRANSFERENCIA   = 20
        SELECT @SIAC            = 21
        SELECT @SPEUA           = 22
        SELECT @TITULAR         = 1
        SELECT @DEAL_EXISTE     = 2601
        SELECT @RESULT          = "ERROR"
        SELECT @CIERTO          = 1
```

```
/*          Asignacion de variables                                    */

BEGIN TRANSACTION ACTUALIZA_ORDEN
/* Se inicializa el código de Error                */
SELECT @iError = @OK

IF @viOrden_Id = @NULL_INTEGER
  BEGIN
          SELECT  @iOrden_Id = ISNULL(Max(Secuencia_Id),0) + 1
          FROM    Secuencia

          /**********          Actualización de la secuencia de ordenes.          **************/
          UPDATE Secuencia SET Secuencia_Id = @iOrden_Id
          SELECT @iError = @@Error
          IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN

          INSERT  Orden
          VALUES  (@iOrden_Id, @viEmisor_Id, @viStatus_Id, NULL, NULL, NULL, NULL, @vsDeal, @viRecurrente,
                   @viOperacion_Id, @viTipoFecha_Id, @vdFecha_Valor, @viTipo_Cambio, @viTipo_Cambio_Mxn,
                   @viDivisa_Id, @viOrden_Importe, @viRecibimos_Id, @viEntregamos_Id, @viComis_Cliente,
                   @viLiquida_Viene,   @viSucursal_Liq, NULL, @viSuc_Id, GETDATE(),    GETDATE(),
                   @NULL_INTEGER, @viProducto_Id, NULL, @viOrden_Brocker,  @viPromotor_Id,
                   @viUsuario_Id)

          SELECT @iError = @@Error

          IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
  END
ELSE
  BEGIN
          SELECT  @iOrden_Id = @viOrden_Id
          SELECT  @iCliente_Id          = B.Cliente_Id,
                  @iRecibimos_Id        = A.Recibimos_Id,
                  @iEntregamos_Id       = A.Entregamos_Id,
                  @iMonto               = A.Orden_Importe
          FROM    Orden A, Ordenante_Recurrente B
          WHERE   A.Orden_Id = B.Orden_Id
          /**********          Actualización de la orden.                 **************/
          UPDATE  Orden
          SET     Emisor_Id                       = @viEmisor_Id,
                  Orden_Deal                      = @vsDeal,
                  Orden_Cliente_Rec               = @viRecurrente,
                  Operacion_Tipo_Id               = @viOperacion_Id,
                  Fecha_Val_Tipo_Id               = @viTipoFecha_Id,
                  Orden_Fecha_Valor               = @vdFecha_Valor,
                  Orden_Tipo_Cambio               = @viTipo_Cambio,
                  Orden_Tipo_Cambio_Mxn           = @viTipo_Cambio_Mxn,
                  Divisa_Id                       = @viDivisa_Id,
                  Orden_Importe                   = @viOrden_Importe,
                  Recibimos_Id                    = @viRecibimos_Id,
                  Entregamos_Id                   = @viEntregamos_Id,
                  Orden_Comis_Por_Cliente         = @viComis_Cliente,
                  Orden_Liquida_Viene = @viLiquida_Viene,
                  Sucursal_Liquida                = @viSucursal_Liq,
                  Promotor_Id                     = @viPromotor_Id,
                  Orden_Brocker           = @viOrden_Brocker,
                  Suc_Id                          = @viSuc_Id,
                  Producto_Id                     = @viProducto_Id
          WHERE   Orden_Id = @iOrden_Id
          SELECT @iError = @@Error
          IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN

          DELETE Ordenante_NoRecurrente WHERE Orden_Id = @iOrden_Id
          SELECT @iError = @@Error
          IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN

          DELETE Ordenante_Recurrente WHERE Orden_Id = @iOrden_Id
          SELECT @iError = @@Error
          IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
```

```
            DELETE Giro WHERE Orden_Id = @iOrden_Id
            SELECT @iError = @@Error
            IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN

  END

/***********          Inserción del cliente.          **************/
IF @viRecurrente = @NULL_INTEGER
  BEGIN
            INSERT  Ordenante_NoRecurrente
            VALUES  (@iOrden_Id,@viSector_Id,@vsCliente,NULL,NULL)
            SELECT @iError = @@Error
            IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
  END
ELSE
  BEGIN
            INSERT  Ordenante_Recurrente
            VALUES  (@iOrden_Id,@viCliente_Id,NULL,NULL,@TITULAR)
            SELECT @iError = @@Error
            IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
  END

/*************          Entregamos "GIROS NUESTROS"          **********/
IF @viEntregamos_Id = @GIROS
  BEGIN
            INSERT  Giro
            VALUES (@GIRO_INICIAL,@iOrden_Id,@NULL_INTEGER,@ORDENADO,@viCorrespEnt_Id,
            @vsBeneficiario,@viOrden_Importe,NULL,NULL,NULL,NULL)

            SELECT @iError = @@Error
            IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN

            UPDATE Orden
            SET   CorrespEntrega_Id = @viCorrespEnt_Id,
                  CorrespEntrega_Cta = @viCta_Entregada
            WHERE Orden_Id = @iOrden_Id
            SELECT @iError = @@Error
            IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
  END

IF @viRecibimos_Id = @REMESAS
  BEGIN
            UPDATE Orden
            SET   CorrespRecibe_Id = @viCorrespRec_Id,
                  CorrespRecibe_Cta = @viCta_Recibida
            WHERE Orden_Id = @iOrden_Id

            SELECT @iError = @@Error
            IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN

  END

/*************          Entregamos "ABONO EN CUENTA"          **********/
IF @viEntregamos_Id = @ABONO_CTA AND @viCliente_Id<>@NULL_INTEGER
  BEGIN
            IF EXISTS(SELECT * FROM Ordenante_Recurrente
                        WHERE Orden_Id = @iOrden_Id
                        AND Cliente_Id = @viCliente_Abono)
              UPDATE Ordenante_Recurrente
              SET       Cuenta_Entrega_Id = @viCta_Entregada
              WHERE  Orden_Id = @iOrden_Id
              AND       Cliente_Id = @viCliente_Abono
            ELSE
              INSERT Ordenante_Recurrente
              VALUES (@iOrden_Id,@viCliente_Abono,@viCta_Entregada,NULL,@NULL_INTEGER)
            SELECT @iError = @@Error
            IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
  END

/*************          Recibimos "CARGO_CTA"          **********/
IF @viRecibimos_Id = @CARGO_CTA AND @viCliente_Id<>@NULL_INTEGER
```

```
        BEGIN
                UPDATE Ordenante_Recurrente
                SET   Cuenta_Recibe_Id = @viCta_Recibida
                WHERE  Orden_Id = @iOrden_Id
                AND   Cliente_Id = @viCliente_Id
                SELECT @iError = @@Error
                IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
        END

        /*************        Recibimos "TRANSFERENCIAS"                    **********/
        IF @viRecibimos_Id = @TRANSFERENCIA or @viRecibimos_Id = @SIAC OR @viRecibimos_Id = @SPEUA
          BEGIN
                UPDATE Orden
                SET   CorrespRecibe_Id = @viCorrespRec_Id,
                      CorrespRecibe_Cta = @viCta_Recibida
                WHERE  Orden_Id = @iOrden_Id
                SELECT @iError = @@Error
                IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
          END

        /*************        Entregamos "TRANSFERENCIAS"                   **********/
        IF @viEntregamos_Id = @TRANSFERENCIA or @viEntregamos_Id = @SIAC OR @viEntregamos_Id = @SPEUA
          BEGIN
                UPDATE Orden
                SET   CorrespEntrega_Id = @viCorrespEnt_Id,
                      CorrespEntrega_Cta = @viCta_Entregada
                WHERE  Orden_Id = @iOrden_Id
                SELECT @iError = @@Error
                IF @iError <> @OK GOTO SALIR_ACTUALIZA_ORDEN
          END

SALIR_ACTUALIZA_ORDEN:
   IF (@iError <> @OK)
          BEGIN
           ROLLBACK TRAN ACTUALIZA_ORDEN
           IF @iError=@DEAL_EXISTE
           SELECT "Error "+LTRIM(STR(@iError))
      return @iError
          END
   ELSE
          BEGIN
           COMMIT TRAN ACTUALIZA_ORDEN
           SELECT @iOrden_Id "Nueva Orden"
      return @OK
          END
END
```

En el módulo de crédito existen dos formas principales: la de captaciones que como ya se mencionó anteriormente es la recaudación de fondos a bajos intereses. En esta forma se necesita registrar el tipo de captación (estas pueden ser cedes, prestamos de bancos nacionales y de bancos extranjeros), el cliente, el banco del cual se va a obtener el préstamo, la divisa, el monto, el dia de inicio, el dia de vencimiento, el periodo de amortización del capital, el periodo de amortizaciones de interés, el tipo de tasa y el porcentaje de la misma. La recaudación de los fondos en la captación se deben prestar a intereses más altos esto es la colocación donde se registra los mismos datos que en la captación. Se requieren principalmente las tablas de Cliente, Corresponsal, Productos, Tipo de Cambio, Usuario, Cuentas de los clientes en Inbursa,

Colocación, Captación, Amortización de captación y colocación, Orden. De estas tablas se derivan otras con las cuales interactuan para obtener o registrar información relacionada.

A continuación se muestran algunos procedimientos que se utilizan para la realización de captaciones y colocaciones en el módulo de crédito.

El siguiente procedimiento (stor procedure) realizado en SQL realiza las altas de las captaciones de recursos y las amortizaciones para el pago de esa captación

```
CREATE PROCEDURE spI_SOI_Captacion (@vi_Folio                    identificador,  @vi_Cliente_Rec          int,
                                    @vs_Cede                     descorta, @vd_Fecha                      datetime,
                                    @vi_Cliente                  identificador, @vi_Corresp_Fondeo        identificador,
                                    @vi_Linea                    identificador, @vi_Capt_Tipo             identificador,
                                    @vd_Suscripcion              datetime,   @vi_Plazo                    int,
                                    @vd_Vencimiento              datetime,   @vf_Monto                    monto,
                                    @vi_Divisa                   identificador, @vi_Corresp               identificador,
                                    @vi_Periodo_Capital          identificador, @vd_Primera_Amort_Cap     smalldatetime,
                                    @vi_VenceCap                 smallint,   @vi_AjusteCap                smallint,
                                    @vi_DiaCap                   smallint,   @vi_Periodo_Ints             identificador,
                                    @vd_Primera_Amort_Ints       smalldatetime, @vi_VenceAmort            smallint,
                                    @vi_AjusteAmort              smallint,   @vi_DiaAmort                 smallint,
                                    @vi_Tasa_Tipo                identificador, @vi_Tasa_Id               identificador,
                                    @vi_Tasa_Aplicada            tasa, @vf_Puntos                         tasa,
                                    @vf_Bruta                    tasa, @vf_Impuesto                       tasa,
                                    @vf_Neta                     tasa, @vf_Descuento                      tasa,
                                    @vb_Ajuste                   booleano,  @vi_Pago_Int                  identificador,
                                    @vi_Captacion                identificador, @vi_Paterno               char(30),
                                    @vi_Materno                  char(30),   @vi_Nombre                   char(30),
                                    @vi_Telefono                 char(10),   @vi_Fax                      char(10),
                                    @vi_Contacto                 char(30),   @NoAmortCap                  int,
                                    @NoAmortInt                  int, @vi_Funcion                         int,
                                    @vi_BandInts                 int,  @vi_Sucursal                       int,
                                    @vi_Referencia       char(10))

AS

BEGIN

 /* Declaracion de Constantes */

 DECLARE @OK                      int
 DECLARE @NULL_INTEGER            int

 DECLARE @CONSECUTIVO             INT
 DECLARE @AMORTIZACION            INT
 DECLARE @MOV_CARGO               INT
 DECLARE @CONCEPT_DISPOSICION     INT
 DECLARE @MOV_ORDINARIO           INT

 /* Declaracion de Variables */

 DECLARE @iError                  smallint
 DECLARE @iCaptaIdNew             identificador
 DECLARE @iCliente_Ant            identificador
 DECLARE @iCliente_Rec_Ant        smallint

 DECLARE @TCaptacion_Id           int
 DECLARE @TCapta_Monto            money
 DECLARE @TCapta_Fecha_Susc       datetime
 DECLARE @TCapta_Fecha_Venc       datetime
```

```
DECLARE @TCapta_Primera_Amort_Cap datetime
DECLARE @TCapta_Primera_Amort_Ints datetime
DECLARE @TPer_Amort_Capital      int
DECLARE @TPer_Amort_Ints         int
DECLARE @TPago_Int_Forma_Id      int
DECLARE @TVencimiento_Id         int
DECLARE @TVencimiento_Amort_Id   int
DECLARE @TCapta_Dia_Unico_Cap    int
DECLARE @TCapta_Dia_Unico_Int    int
DECLARE @TCapta_Tasa_Neta        tasa

DECLARE @MaxTransac              int

/* Asignación de Constantes */

SELECT @OK          = 0
SELECT @NULL_INTEGER = 0

SELECT @CONSECUTIVO        = 1
SELECT @AMORTIZACION       = 1
SELECT @MOV_CARGO          = 1
SELECT @CONCEPT_DISPOSICION  = 4
SELECT @MOV_ORDINARIO      = 10


/* Se Inicia la Transacción */

BEGIN TRAN INSERTA_CAPTA

 /* Se Inicializa el Código de Error */

 SELECT @iError = @OK

 /* Elimina Registros de Clientes Recurrentes o no Recurrentes */

 DELETE Credito_Cliente_NoRec
 WHERE  Captacion_Id = @vi_Folio
 SELECT @iError = @@Error
 IF @iError<>@OK GOTO SALIR_INSERTA_CAPTA

 DELETE Credito_Cliente_Rec
 WHERE  Captacion_Id = @vi_Folio
 SELECT @iError = @@Error
 IF @iError<>@OK GOTO SALIR_INSERTA_CAPTA

 /* Intenta Actualizar */
 IF EXISTS(SELECT * FROM Captacion WHERE Captacion_Id=@vi_Folio)
        UPDATE Captacion
        SET     Capta_CEDE                      = @vs_Cede,
                Capta_Fecha_Captura             = @vd_Fecha,
                Capta_Cliente_Rec               = @vi_Cliente_Rec,
                Corresp_Fondeo                  = @vi_Corresp_Fondeo,
                Linea_Id                        = @vi_Linea,
                Capt_Tipo_Id                    = @vi_Capt_Tipo,
                Capta_Fecha_Susc                = @vd_Suscripcion,
                Capta_Plazo             = @vi_Plazo,
                Capta_Fecha_Vene                = @vd_Vencimiento,
                Capta_Monto                     = @vf_Monto,
                Divisa_Id                       = @vi_Divisa,
                Corresp_Id                      = @vi_Corresp,
                Capta_Per_Amort_Capital         = @vi_Periodo_Capital,
                Capta_Primera_Amort_Cap         = @vd_Primera_Amort_Cap,
                Vencimiento_Id                  = @vi_VeneCap,
                Capta_Ajuste_Dia_Cap            = @vi_AjusteCap,
                Capta_Dia_Unico_Cap             = @vi_DiaCap,
                Capta_Per_Amort_Ints            = @vi_Periodo_Ints,
                Capta_Primera_Amort_Ints        = @vd_Primera_Amort_Ints,
                Vencimiento_Capt_Amort_Id       = @vi_VeneAmort,
                Capta_Ajuste_Dia_Amort          = @vi_AjusteAmort,
                Capta_Dia_Unico_Int             = @vi_DiaAmort,
                Tasa_Tipo_Id                    = @vi_Tasa_Tipo,
```

```
                    Tasa_Id                     = @vi_Tasa_Id,
                    Capta_Tasa_Aplicada         = @vi_Tasa_Aplicada,
                    Capta_Puntos_Mas            = @vt_Puntos,
                    Capta_Tasa_Bruta            = @vt_Bruta,
                    Capta_Imp_WHT               = @vt_Impuesto,
                    Capta_Tasa_Neta             = @vt_Neta,
                    Capta_Tasa_Desc             = @vt_Descuento,
                    Capta_Ajuste_Ints           = @vb_Ajuste,
                    Pago_Int_Forma_Id           = @vi_Pago_Int,
                    Capta_Referencia            = @vi_Referencia
            WHERE Captacion_Id                  = @vi_Folio
    ELSE
            BEGIN
                    SELECT @vi_Folio = ISNULL(MAX(Secuencia_Id), @NULL_INTEGER) + 1 FROM  Secuencia
                    UPDATE Secuencia SET Secuencia_Id = @vi_Folio
                    SELECT @iError = @@Error
                    IF @iError<>@OK GOTO SALIR_INSERTA_CAPTA

                    INSERT Captacion

                    VALUES  (@vi_Folio, @vs_Cede, @vd_Fecha, @vi_Cliente_Rec, @vi_Corresp_Fondeo, @vi_Linea,
                             @vi_Capt_Tipo, @vd_Suscripcion, @vi_Plazo, @vd_Vencimiento, @vf_Monto, @vi_Divisa,
                             @vi_Corresp, @vi_Periodo_Capital, @vd_Primera_Amort_Cap, @vi_VenCap, @vi_AjusteCap,
                    @vi_DiaCap, @vi_Periodo_Ints, @vd_Primera_Amort_Ints, @vi_VenAmort, @vi_AjusteAmort,
            @vi_DiaAmort, @vi_Tasa_Tipo, @vi_Tasa_Id, @vi_Tasa_Aplicada, @vt_Puntos, @vt_Bruta,
            @vt_Impuesto, @vt_Neta, @vt_Descuento, @vb_Ajuste, @vi_Pago_Int, 1, @vi_Sucursal,
            @vi_Referencia)
            END

    SELECT @iError = @@Error
    IF @iError<>@OK GOTO SALIR_INSERTA_CAPTA


    DELETE Captacion_Amort WHERE Captacion_Id = @vi_Folio
    SELECT @iError = @@Error
    IF @iError<>@OK GOTO SALIR_INSERTA_CAPTA

    SELECT @TCaptacion_Id           = Captacion_Id,
        @TCapta_Monto               = Capta_Monto,
        @TCapta_Fecha_Suse          = Capta_Fecha_Suse,
        @TCapta_Fecha_Venc          = Capta_Fecha_Venc,
        @TCapta_Primera_Amort_Cap   = Capta_Primera_Amort_Cap,
        @TCapta_Primera_Amort_Ints  = Capta_Primera_Amort_Ints,
        @TPer_Amort_Capital         = Capta_Per_Amort_Capital,
        @TPer_Amort_Ints            = Capta_Per_Amort_Ints,
        @TCapta_Tasa_Neta           = Capta_Tasa_Neta,
        @TPago_Int_Forma_Id         = Pago_Int_Forma_Id,
        @TVencimiento_Id            = Vencimiento_Id,
        @TVencimiento_Amort_Id      = Vencimiento_Capt_Amort_Id,
        @TCapta_Dia_Unico_Cap       = Capta_Dia_Unico_Cap,
        @TCapta_Dia_Unico_Int       = Capta_Dia_Unico_Int
    FROM  Captacion
    WHERE Captacion_Id = @vi_Folio

    EXEC @iError=spl_SOL_Amon_Capital_Capt @TCaptacion_Id, @TCapta_Monto, @TCapta_Fecha_Suse,
                                           @TCapta_Fecha_Venc, @TCapta_Primera_Amort_Cap, @TCapta_Primera_Amort_Ints,
                                           @TPer_Amort_Capital, @TPer_Amort_Ints, @NoAmortCap, @NoAmortInt,
                                           @vt_Bruta, @TPago_Int_Forma_Id, @TVencimiento_Id, @TVencimiento_Amort_Id,
                                           @TCapta_Dia_Unico_Cap, @TCapta_Dia_Unico_Int, @vi_BandInts
    SELECT @iError = @@Error
    IF @iError<>@OK GOTO SALIR_INSERTA_CAPTA

/*  IF NOT EXISTS (SELECT * FROM Movimiento WHERE Captacion_Id = @vi_Folio)
        BEGIN
        SELECT @MaxTransac = ISNULL(MAX(Movimiento_Transac),0) + 1 FROM  Movimiento

        /* Se inserta el Movimiento */
        Insert into Movimiento(Movimiento_Transac, Movimiento_Id, Captacion_Id, Captacion_Amort_Id,
                               Movimiento_Fecha,   Movimiento_Cantidad, Movimiento_Cargo, Concepto_Id,
                               Tipo_Movimiento_Id)
        Values  (@MaxTransac, @CONSECUTIVO, @vi_Folio, @AMORTIZACION, GetDate(), @vf_Monto,
```

```
                @MOV_CARGO, @CONCEPT_DISPOSICION, @MOV_ORDINARIO)

        SELECT @iError = @@Error
        IF @iError<>@OK GOTO SALIR_INSERTA_CAPTA
        END*/

SALIR_INSERTA_CAPTA:
IF @iError=@OK
  BEGIN
    COMMIT TRAN INSERTA_CAPTA
    SELECT @vi_Folio "Captacion Afectada"
  END
ELSE
  ROLLBACK TRAN INSERTA_CAPTA

RETURN(@iError)
END
```

En el módulo de contabilidad lo que se realiza es el registro de las cuantas contables las cuales se verán afectadas en el registro de las operaciones, la liquidación de las mismas, en su vencimiento, entre otras; la verificación de las cuentas en cuanto a su afectación diaria. Para este registro se requieren principalmente las tablas de Cuenta Contable, Productos, Tipo de Cambio, Cuentas de los clientes en Inbursa, Orden, Movimientos generados. De estas tablas se derivan otras con las cuales interactuan para obtener o registrar información relacionada.

A continuación se muestra un procedimiento que se utiliza para la generación de movimientos contables según las operaciones realizadas.

```
CREATE PROCEDURE spI_SOI_Genera_MovHoy( @xdFecha          datetime, @xdFechaAnt          datetime)
AS
BEGIN
        /*      Declaracion de constantes                       */
        DECLARE @OK                     int
        DECLARE @NULL_INTEGER int
        DECLARE @LIQUIDADO              int
        DECLARE @LIQUIDADOSIBI int
        DECLARE @CANCELADO              int
        DECLARE @HOY                    int
        DECLARE @CAMBIOS                int
        DECLARE @GIROS                  int
        DECLARE @REMESAS                int
        DECLARE @CIERTO                 int
        DECLARE @CHEQUESBC              int
        DECLARE @DIRECTO                int
        DECLARE @SUCLIQ                 int
        DECLARE @CUENTACLI              int
        DECLARE @SUCEMI                 int
        DECLARE @BANCAEMP               int

        /*      Declaracion de variables                        */
        DECLARE @iError                 int
        DECLARE @iMovto_Id              int
        DECLARE @iOrden_Id              int
        DECLARE @iOperacion             int
        DECLARE @iProducto              int
```

```
DECLARE @iDivisa_Oper                   int
DECLARE @iCuenta_Id                     int
DECLARE @sCuentaNumero                  char(12)
DECLARE @iAfecta_Id                     int
DECLARE @iDivisa_Id                     int
DECLARE @sCuenta            varchar(4)
DECLARE @sSCuenta                       varchar(2)
DECLARE @sSSCuenta                      varchar(2)
DECLARE @sSSSCuenta                     varchar(2)
DECLARE @sConcepto                      varchar(2)
DECLARE @sDivisa            varchar(2)
DECLARE @sNatura            varchar(1)
DECLARE @sSector                        varchar(3)
DECLARE @fImporte                       float
DECLARE @fEquiv                         float
DECLARE @sMovto_Dsc                     varchar(100)
DECLARE @sCuenta_Dsc                    varchar(100)
DECLARE @sFecha                         varchar(10)
DECLARE @iTipoCliente                   int
DECLARE @iRequiereSec                   int
DECLARE @sFolio                         char(8)
DECLARE @iCentroResp                    int
DECLARE @iAfecta_Tipo                   int


/*          Asignación de constantes                    */
SELECT @OK                  0
SELECT @NULL_INTEGER        0
SELECT @HOY                 1
SELECT @LIQUIDADO           5
SELECT @LIQUIDADOSIDI       4
SELECT @BANCAEMP            10
SELECT @CANCELADO           8
SELECT @CAMBIOS             27
SELECT @GIROS               3
SELECT @REMESAS             4
SELECT @CIERTO              1
SELECT @CHEQUESBC           11
SELECT @DIRECTO             1
SELECT @SUCLIQ              2
SELECT @CUENTACLI           3
SELECT @SUCEMI              4


CREATE  TABLE #Temporal
        (Orden_Id                       int,
        Operacion_Tipo_Id               int,
        Producto_Id                     int,
        Divisa_Oper                     int,
        Cuenta_Conta_Id                 int,
        Cuenta_Cheques                  char(12) null,
        Afectacion_Id                   int,
        Divisa_Id                       int,
        Importe                         money,
        Equivalente                     money,
        Natura                          char(1),
        Fecha                           char(10))

/*      COMPRA VENTA DE EFECTIVO                        */
INSERT  #Temporal
SELECT  A.Orden_Id                                              "Orden",
        B.Operacion_Tipo_Id                             "Operacion",
        B.Producto_Id                                   "Producto",
        B.Divisa_Id                                     "Divisa_Oper",
        C.Cuenta_Conta_Id                               "Cuenta",
        NULL,
        C.Afectacion_Contable_Id                        "Afectacion",
        D.Divisa_Id                                     "Divisa_Id",
        A.Orden_Efectivo_Importe                        "Importe",
        A.Orden_Efectivo_Importe * B.Orden_Tipo_Cambio  "Equivalente",
        SUBSTRING("12",C.Afectacion_Contable_Abono+1,1)         "Natura",
```

```
                CONVERT(varchar(4),DATEPART(year,@vdFecha))+"-"+SUBSTRING(CONVERT(varchar(8),@vdFecha,1),1,2)+"-
"+SUBSTRING(CONVERT(varchar(8),@vdFecha,1),4,2) "Fecha"
        FROM    Orden_Efectivo A, Orden B, Afectacion_Contable_Oper C, Afectacion_Contable D
        WHERE   A.Orden_Id = B.Orden_Id
        AND     B.Divisa_Id = C.Divisa_Operativa_Id
        AND     B.Operacion_Tipo_Id = C.Operacion_Tipo_Id
        AND     B.Fecha_Val_Tipo_Id = C.Fecha_Val_Tipo_Id
        AND     A.Producto_Id = C.Producto_Id
        AND     B.Fecha_Val_Tipo_Id = C.Fecha_Val_Tipo_Id
        AND     A.Orden_Efectivo_Recibimos = C.Entrega_Recibe
        AND     C.Cuenta_Conta_Id = D.Cuenta_Conta_Id
        AND     C.Divisa_Operativa_Id = D.Divisa_Operativa_Id
        AND     C.Afectacion_Contable_Id = D.Afectacion_Contable_Id
        AND     D.Afecta_Comision = @NULL_INTEGER
        AND     D.Afecta_IVA = @NULL_INTEGER
        AND     CONVERT(char(10),B.Orden_Fecha_Valor,102) = CONVERT(char(10),@vdFecha,102)
        AND     B.Orden_Status_Id in (@LIQUIDADO, @LIQUIDADOSIBI,@BANCAEMP)
        AND     B.Fecha_Val_Tipo_Id = @HOY


/* inicia la transaccion */

BEGIN TRANSACTION CIERRE_MOVTO_HOY
SELECT @iError = @OK

DECLARE Movtos_Por_Orden CURSOR FOR
SELECT * FROM #Temporal

OPEN Movtos_Por_Orden
FETCH   Movtos_Por_Orden INTO @iOrden_Id, @iOperacion, @iProducto, @iDivisa_Oper,
        @iCuenta_Id, @sCuentaNumero, @iAfecta_Id, @iDivisa_Id, @fImporte, @fEquiv, @sNatura, @sFecha
WHILE (@@sqlstatus = @NULL_INTEGER)
  BEGIN

        SELECT @iCentroResp = Centro_Resp_Id , @iAfecta_Tipo = Afectacion_Tipo_Id
        FROM   Afectacion_Contable
        WHERE  Cuenta_Conta_Id = @iCuenta_Id
        AND    Divisa_Operativa_Id = @iDivisa_Oper
        AND    Afectacion_Contable_Id = @iAfecta_Id
        AND    Divisa_Id = @iDivisa_Id

        IF @iAfecta_Tipo  = @SUCLIQ
                SELECT @iCentroResp = A.Centro_Resp_Id
                FROM   Centro_Resp A, Orden B
                WHERE  B.Sucursal_Liquida  = A.Suc_Id
                AND    B.Orden_Id = @iOrden_Id

        IF @iAfecta_Tipo  = @CUENTACLI
                SELECT @iCentroResp = Centro_Resp_Id
                FROM   Centro_Resp
                WHERE  Centro_Resp_Id_SIBI = substring(@sCuentaNumero,1,3)
        IF @iAfecta_Tipo  = @SUCEMI
                SELECT @iCentroResp = A.Centro_Resp_Id
                FROM   Centro_Resp A, Orden B
                WHERE  B.Suc_Id  = A.Suc_Id
                AND    B.Orden_Id = @iOrden_Id


        SELECT  @sFolio = "O"+RIGHT("0000000"+CONVERT(VARCHAR(7),@iOrden_Id),7)

        SELECT  @iMovto_Id = ISNULL(MAX(Movto_Generado_Id+1),1)
        FROM    Movimiento_Generado

        SELECT  @iRequiereSec = Cuenta_Conta_Sector
        FROM    Cuenta_Contable
        WHERE   Cuenta_Conta_Id = @iCuenta_Id

        IF @iRequiereSec = @CIERTO
          BEGIN
          SELECT @iTipo_Cliente = Orden_Cliente_Rec FROM Orden WHERE Orden_Id=@iOrden_Id
```

```
                IF @iTipoCliente <> @NULL_INTEGER
                        SELECT  @sSector = B.Sector_SIBI
                        FROM    Ordenante_NoRecurrente A, Sector B
                        WHERE   A.Orden_Id = @iOrden_Id
                        AND     A.Sector_Id = B.Sector_Id
                ELSE
                        SELECT  @sSector = C.Sector_SIBI
                        FROM    Ordenante_Recurrente A, Cliente B, Sector C
                        WHERE   A.Orden_Id = @iOrden_Id
                        AND     A.Cliente_Id = B.Cliente_Id
                        AND     B.Sector_Id = C.Sector_Id
                END
                ELSE
                  SELECT @sSector = "  "

                SELECT  @sCuenta = Substring(Cuenta_Conta_Numero,1,4),
                        @sSCuenta = Substring(Cuenta_Conta_Numero,5,2),
                        @sSSCuenta = Substring(Cuenta_Conta_Numero,7,2),
                        @sSSSCuenta = Substring(Cuenta_Conta_Numero,9,2),
                        @sConcepto = Substring(Cuenta_Conta_Numero,11,2),
                        @sCuenta_Dsc = Cuenta_Conta_Dsc
                FROM    Cuenta_Contable
                WHERE   Cuenta_Conta_Id = @iCuenta_Id

                SELECT  @sMovto_Dsc = B.Operacion_Tipo_Dsc+" DE "+C.Producto_Dsc+" RECIBIMOS "+D.Producto_Dsc+"
ENTREGAMOS "+E.Producto_Dsc
                FROM    Orden A, Operacion_Tipo B, Producto C, Producto D, Producto E
                WHERE   A.Operacion_Tipo_Id = B.Operacion_Tipo_Id
                AND     A.Producto_Id = C.Producto_Id
                AND     A.Recibimos_Id = D.Producto_Id
                AND     A.Entregamos_Id = E.Producto_Id
                AND     A.Orden_Id = @iOrden_Id

                SELECT  @sDivisa = Divisa_Id_SIBI
                FROM    Divisa
                WHERE   Divisa_Id = @iDivisa_Id

                INSERT  Movimiento_Generado
                VALUES  (@iMovto_Id, @iCentroResp, @iCuenta_Id, @iDivisa_Oper, @iAfecta_Id, 1,
                        @sCuenta, @sSCuenta, @sSSCuenta, @sSSSCuenta, @sConcepto, @sDivisa,
                        "01", @sNatura, @fImporte, @sMovto_Dsc, @sCuenta_Dsc, @sSector,
                        "01", @sFecha, 0, NULL, @sFolio)

                SELECT @iError = @@Error
                IF @iError <> @OK GOTO SALIR_CIERRE_MOVTO
                FETCH   Movtos_Por_Orden INTO @iOrden_Id, @iOperacion, @iProducto, @iDivisa_Oper,
                        @iCuenta_Id, @sCuentaNumero, @iAfecta_Id, @iDivisa_Id, @fImporte, @fEquiv, @sNatura, @sFecha
        END

        CLOSE Movtos_Por_Orden
        DEALLOCATE CURSOR Movtos_Por_Orden

SALIR_CIERRE_MOVTO:
IF (@iError <> @OK)
  ROLLBACK TRAN ACTUALIZA_MOVTO_HOY
ELSE
  COMMIT TRAN ACTUALIZA_MOVTO_HOY
END
```

En el módulo de administración una de las cosas más importantes es la seguridad tanto del sistema como de la Base de Datos en cuanto a los permisos que tienen los usuarios.

El siguiente es un procedimiento con el cual se pueden otorgar o quitar permisos en cuanto al proyecto.

```
IF EXISTS (SELECT * FROM sysobjects WHERE id = object_id('dbo.spI_CreaUsuariosSOI'))
```

```
    DROP procedure dbo.spI_CreaUsuariosSOI
go
create procedure dbo.spI_CreaUsuariosSOI(  @vsUsuario  char(10),
                                           @vsPassword char(70),
                                  @sGroupSOI  char(30) )

As

BEGIN

 /*Constantes*/
 DECLARE @OK tinyint
 SELECT  @OK = 0

 DECLARE @NULL_INTEGER  tinyint
 SELECT  @NULL_INTEGER  = 0

 DECLARE @sDESASOI varchar(10)
 SELECT  @sDESASOI = "desasoi"


 /*Variables*/
 Declare @iError         int

 DECLARE @sServerSIBI    varchar(20)
 DECLARE @sDataBaseSIBI  varchar(20)
 DECLARE @sGroupSIBI     varchar(20)
 SELECT  @sGroupSIBI =   'public'

 DECLARE @sServerSOI     varchar(20)
 DECLARE @sDataBaseSOI   varchar(20)
/*  DECLARE @sGroupSOI      varchar(20)
 SELECT  @sGroupSOI =    'public'*/

 Declare @sRuta          varchar(255)


 /*Limpia Variables */
 SELECT @vsUsuario   = ltTrim(rTrim(@vsUsuario))
 SELECT @vsPassword  = ltTrim(rTrim(@vsPassword))

 Select @iError = @OK    /*Inicializa la Var. de Error*/


 /**************Obtiene Datos Base de Datos*/

 /* Datos del SOI */
 Select @sServerSOI    = ltrim(rtrim(Server_Name_SOI)),
     @sDataBaseSOI  = ltrim(rtrim(DB_Name_SOI))

From   InfoServer

 /* Datos de SIBI */
 Select @sServerSIBI    = ltrim(rtrim(Server_Name_SIBI)),
     @sDataBaseSIBI  = ltrim(rtrim(DB_Name_SIBI))

From   InfoServer


/* Verifica si Existe el Login en el Servidor de SOI */
If Exists ( Select name From master.dbo.syslogins Where name = @vsUsuario )
 Begin
  Select @iError = 15300   /* El Login ya Existe en SOI */
  Select @iError Error     /* Retornamos Error a la Aplicación */
  Return @iError           /* Interrumpimos Ejecución */
 End

/* Verifica si Existe el User en el Servidor de SOI y Base de Datos SOI */
If Exists ( Select name From sysusers Where name = @vsUsuario )
 Begin
  Select @iError = 15200   /* El User ya Existe en la Base de Datos del SOI */
```

```
    Select @iError Error    /* Retornamos Error a la Aplicación */
    Return @iError          /* Interrumpimos Ejecución */
   End

  /*********************************************/
  /* Crea el Login (SOI)                       */
  /*********************************************/

  Exec @iError = sp_addlogin @vsUsuario, @vsPassword, @sDataBaseSOI

  /* Verifica Error de Regreso */
  If @iError <> @OK
   Begin
    Select @iError = 15400  /* Error al Generar el Login en el Servidor del SOI*/
    Select @iError Error    /* Retornamos Error a la Aplicación */
    Return @iError          /* Interrumpimos Ejecución */
   End

  /*********************************************/
  /* Crea el User (SOI)                        */
  /*********************************************/

  Exec @iError = sp_adduser  @vsUsuario, @vsUsuario, @sGroupSOI

  /* Verifica el Error de Regreso */
  If @iError <> @OK
   Begin
    /* Elimina el Login de SOI*/
    Exec sp_droplogin @vsUsuario

    /* Retornamos Error */
    Select @iError = 15500  /* Error al Generar el User en la Base de Datos del SOI */
    Select @iError Error    /* Retornamos Error a la Aplicación *
    Return @iError          /* Interrumpimos Ejecución */
   End


  /*********************************************/
  /* Crea el Login (SIBI)                      */
  /*********************************************/

  /* Verificamos Si <Servidor SIBI> = <Servidor SOI> */
  If ( @sServerSIBI = @sServerSOI )
   Begin

    /* Generamos sólo un Usuario en la Base de Datos de SIBI, */
    /* ya que el Login ya existe.                             */
    Select @sRuta  = @sDataBaseSIBI + "..sp_adduser"
    Exec  @iError = @sRuta  @vsUsuario, @vsUsuario, @sGroupSIBI

    /* Verifica el Error de Regreso */
    If @iError <> @OK
     Begin
      /* Elimina el Usuario de la Base de Datos del SOI */
      Exec sp_dropuser  @vsUsuario
      /* Elimina el Login del Servidor del SOI */
      Exec sp_droplogin @vsUsuario

      /* Retornamos Error */
      Select @iError Error    /* Retornamos Error a la Aplicación */
      Return @iError          /* Interrumpimos Ejecución */
     End

   End  /* Fin Si <Servidor SIBI> = <Servidor SOI> */

  /* Verificamos Si <Servidor SIBI> != <Servidor SOI> */
  If ( @sServerSIBI != @sServerSOI )
   Begin

    /* Generamos Login Remoto en SIBI para Login de SOI */
    Select @sRuta = @sServerSIBI + "." + "master..sp_addremotelogin"
```

```
      Exec @iError = @sRuta @sServerSOI, @sDESASOI, @vsUsuario

      If @iError <> @OK
        Begin
         /* Elimina el Usuario de la Base de Datos del SOI */
         Exec sp_dropuser @vsUsuario
         /* Elimina el Login del Servidor del SOI */
         Exec sp_droplogin @vsUsuario

         /* Retornamos Error */
         Select @iError Error     /* Retornamos Error a la Aplicación */
         Return @iError           /* Interrumpimos Ejecución */
        End

      /* Activa la opción Trusted para el Login Remoto en SIBI para Login de SOI */
      Select @sRuta = @sServerSIBI +"."+"..sp_remoteoption"
      Exec @iError = @sRuta @sServerSOI, @sDESASOI, @vsUsuario, trusted, true

      If @iError <> @OK
        Begin
         /* Elimina el Login Remoto en SIBI para Login de SOI */
         Select @sRuta = @sServerSIBI + "." + "master..sp_dropremotelogin"
         Exec @sRuta @sServerSOI, @sDESASOI, @vsUsuario
         /* Elimina el Usuario de la Base de Datos del SOI */
         Exec sp_dropuser @vsUsuario
         /* Elimina el Login del Servidor del SOI */
         Exec sp_droplogin @vsUsuario

         /* Retornamos Error */
         Select @iError Error     /* Retornamos Error a la Aplicación */
         Return @iError           /* Interrumpimos Ejecución */
        End

    End  /* Fin Si <Servidor SIBI> != <Servidor SOI> */

  /* Retornamos estado de Error a la Aplicación */
  IF @@ERROR <> 0
    SELECT @iError = @@ERROR

  SELECT @iError Error

END

go
begin
  Grant Exec On dbo.spI_CreaUsuariosSOI to Public
end
```

CONCLUSIONES

Con respuesta a la hipótesis planteada y verificar que se alcanzaron los objetivos del presenta trabajo se dan los siguientes beneficios de la implementación de una Base de Datos Relacional.

Potencial para imponer normas.

Con el enfoque de la Bases de Datos el administrador de Bases de Datos define imponer normas a los usuarios de la Base de Datos en una organización grande. Esto facilita la comunicación y cooperación entre diversos departamentos, proyectos y usuarios de esa organización. Es posible definir normas para los nombres y formatos de los elementos de información, para los formatos de presentación, para las estructuras de los informes, para la terminologia, etc. Es más fácil que el administrador de Bases de Datos imponga normas en un entorno centralizado de la Base de Datos que en un entorno en el que cada grupo de usuarios tenga el control de sus propios archivos y programas.

Menor tiempo de creación de aplicaciones.

Una de la caracteristicas más convincentes a favor del enfoque de las Bases de Datos es que la aplicación de una aplicación nueva, como la obtención de cierta información de la Base de Datos para imprimir un nuevo informe, requiere muy poco tiempo. Diseñar e implementar una nueva Base de Datos desde cero puede tardar más que escribir una sola aplicación de archivos especializada; sin embargo, una vez que la Base de Datos esta construida y en funciones, casi siempre se requerirá mucho menos tiempo para crear nuevas aplicaciones con los recursos del Sistema de Gestión de Bases de Datos. Se estima que el tiempo de creación con un Sistema de Gestión de Bases de Datos es de una sexta a una cuarta parte del requerido en un sistema de archivos tradicional.

Flexibilidad.

En ocasiones es necesario modificar la estructura de una Base de Datos cuando cambian los requerimientos. Por ejemplo, podia surgir un nuevo grupo de usuarios que necesite

información adicional que no se encuentra actualmente en la Base de Datos. Para atenderlos, tal vez sea necesario añadir un nuevo archivo a la Base de Datos o extender los elementos de un archivo ya existente. Algunos Sistemas de Gestión de Bases de Datos permiten efectuar estas modificaciones en la estructura de la Base de Datos sin afectar los datos almacenados y a los programas de aplicación ya existentes.

Disponibilidad de información actualizada.

Los Sistemas de Gestión de Bases de Datos pone la Base de Datos a disposición de todos los usuarios. En el momento en que un usuario actualiza la Base de Datos, todos los demás usuarios pueden ver de inmediato esta actualización. Esta disponibilidad de información actualizada es indispensable en muchas aplicaciones de procesamiento de transacciones, como los sistemas de reservaciones o las Bases de Datos bancarias como en el caso practico realizado en el Grupo Financiero Inbursa, y se hace posible a los subsistemas de control de concurrencia y de recuperación del Sistema de Gestión de Bases de Datos.

Economías en escala.

El enfoque del Sistema de Gestión de Bases de Datos permite consolidar los datos y las aplicaciones, reduciéndose asi el desperdicio por traslapo entre las actividades del personal de procesamiento de datos en los diferentes proyectos o departamentos. Esto permite que la organización completa invierta en procesadores más potentes, dispositivos de almacenamiento o equipo de comunicación, en vez de que cada departamento tenga que adquirir por separado su propio equipo (de menor capacidad). Esto reduce los costos totales de operación y control.

Con los anteriores beneficios que otorgan las Bases de Datos solo queda decir que las Bases de Datos en toda organización hoy en dia es la mejor herramienta para el manejo de la información.

BIBLIOGRAFÍA

· Tensel, VZ Abdullah
Temporal DataBase
1997, editorial the Benjamin/Cummings Publishing Company.
630 p.

· Rivera
Bases de Datos Relaciónales
Segunda edición, Madrid, editorial Paraninfo Magallnes, 310 p.

· Harryskiewyez, Igortitus
Análisis y diseño de Bases de Datos
Primera edición, México D.F. 1996, editorial Limusa, 680 p.

· Harrington, Jan L.
Relational data Base management
Editorial Holt, Rinebart and Winston, inc. U.S.A. 1996, 378 p

· Jackson Glenn A.
Introducción al diseño de Bases de Datos relaciónales
Madrid España, editorial Anaya Multimedia, 203 p

· Date, C. J.
Introducción a los sistemas de Bases de Datos
México D.F. 1995 , editorial Addison-Wesley Iberoamericana, 648 p

· Miguel Castaño, Adoración.
Concepción y diseño de Bases de Datos: del modelo E/R al modelo relacional
México D.F. 1996 ,editorial Addison-Wesley Iberoamericana, 989 p

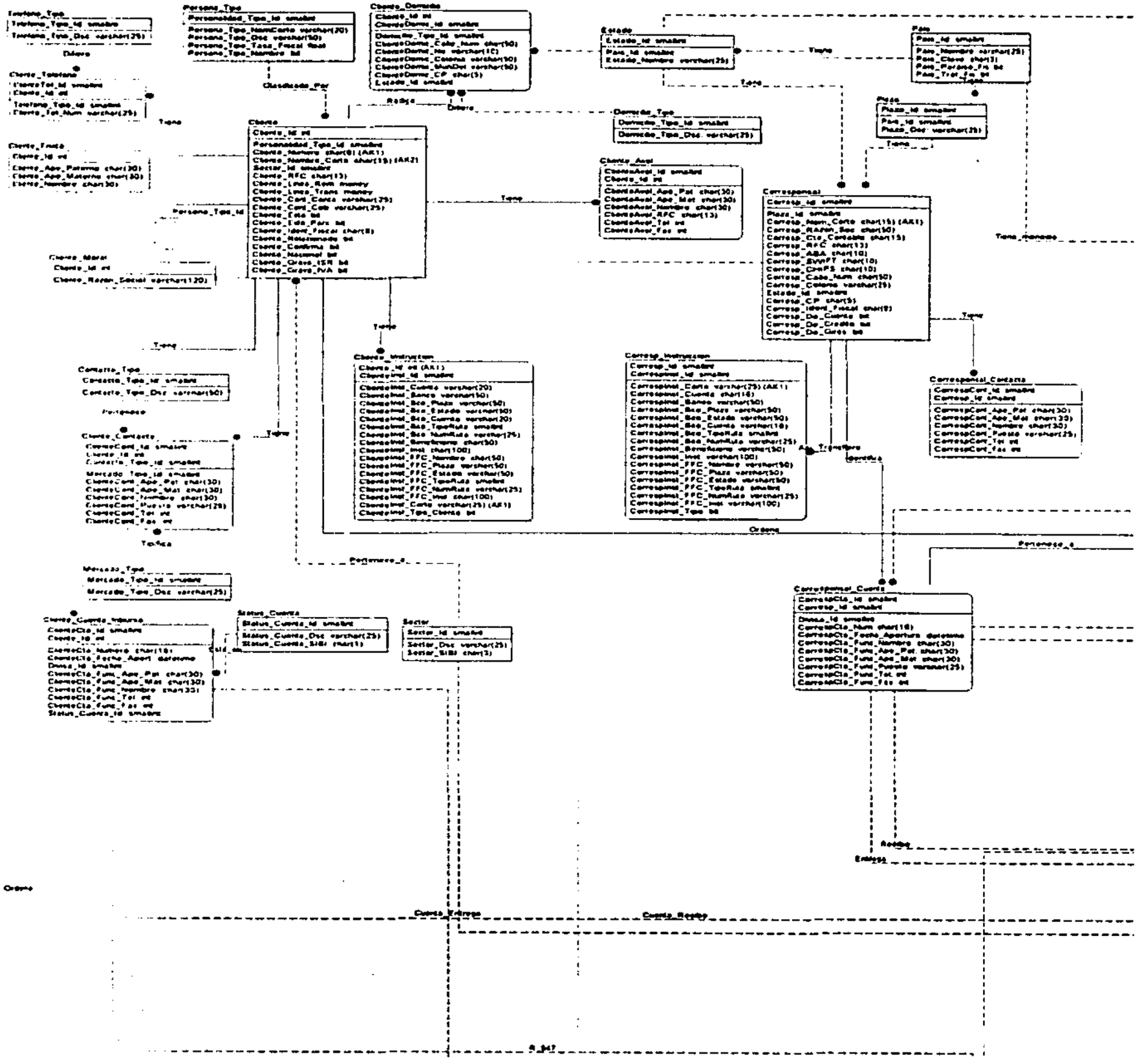
· Ramez Elmasari, Shamkant B. Navathe.
Sistemas de Bases de Datos.
Segunda edición, México D.F. 1997, editorial Addison-Wesley Iberoamericana, 887 p.

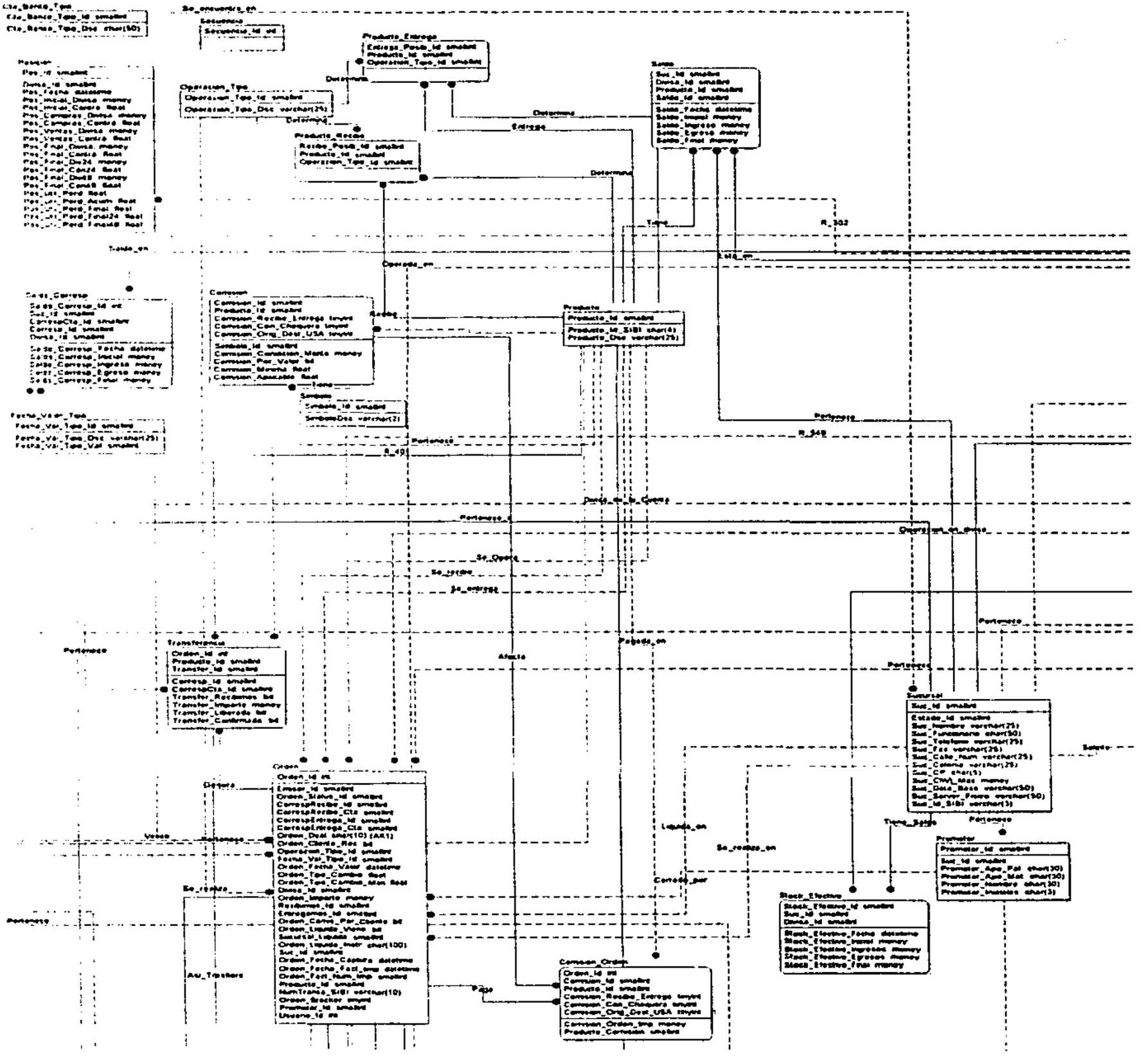
· Centro educacional DDEMESIS.
Seminario desarrollo Cliente/Servidor con Visual Basic y SQL Server.
Primera edición, México D.F. 1997, editorial grupo DDEMESIS, 143 p.

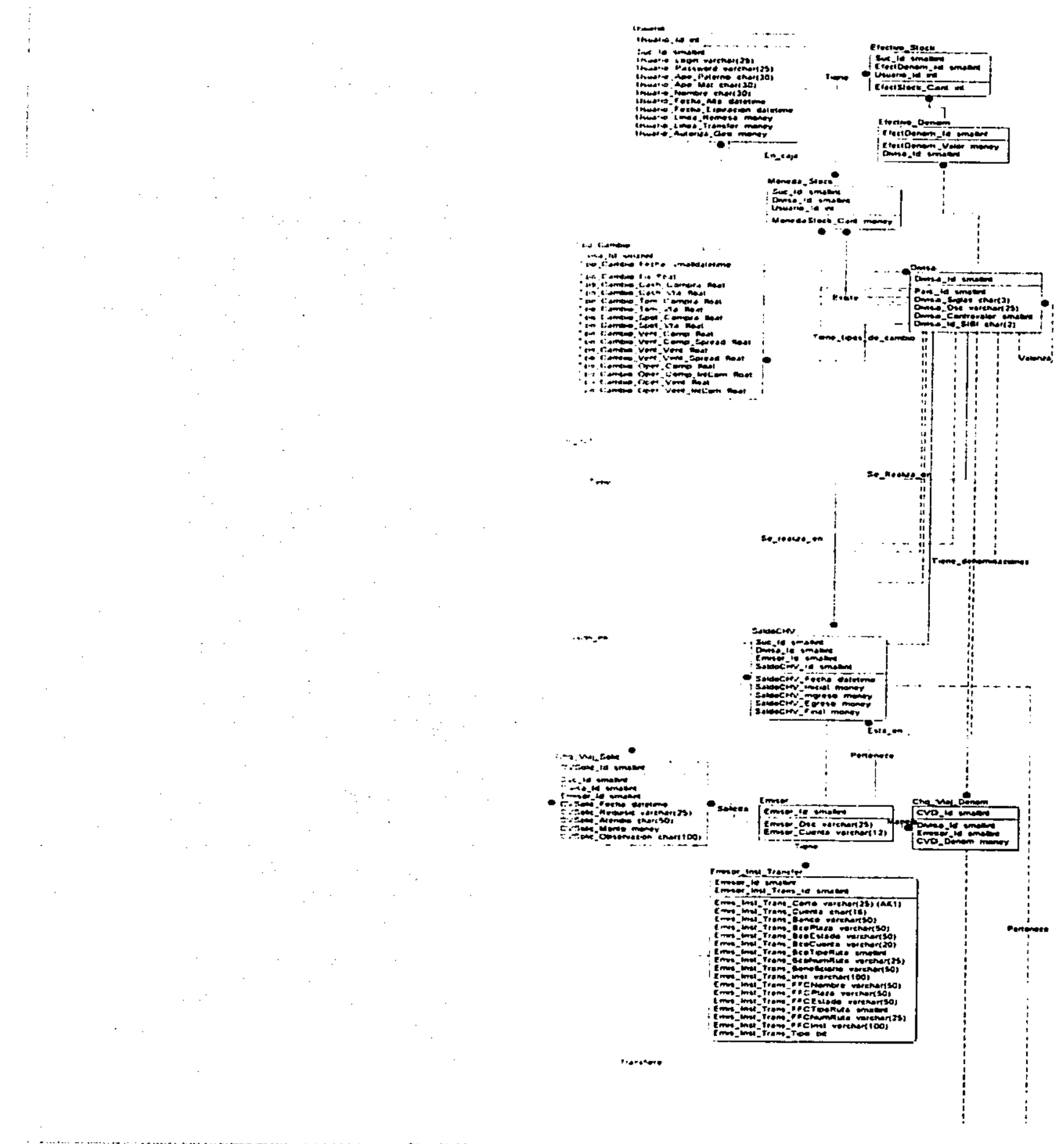
· Microsoft.
Manual Microsoft SQL Server DataBase Implementation ver. 6.5.
México D. F. 1997, Microsoft.

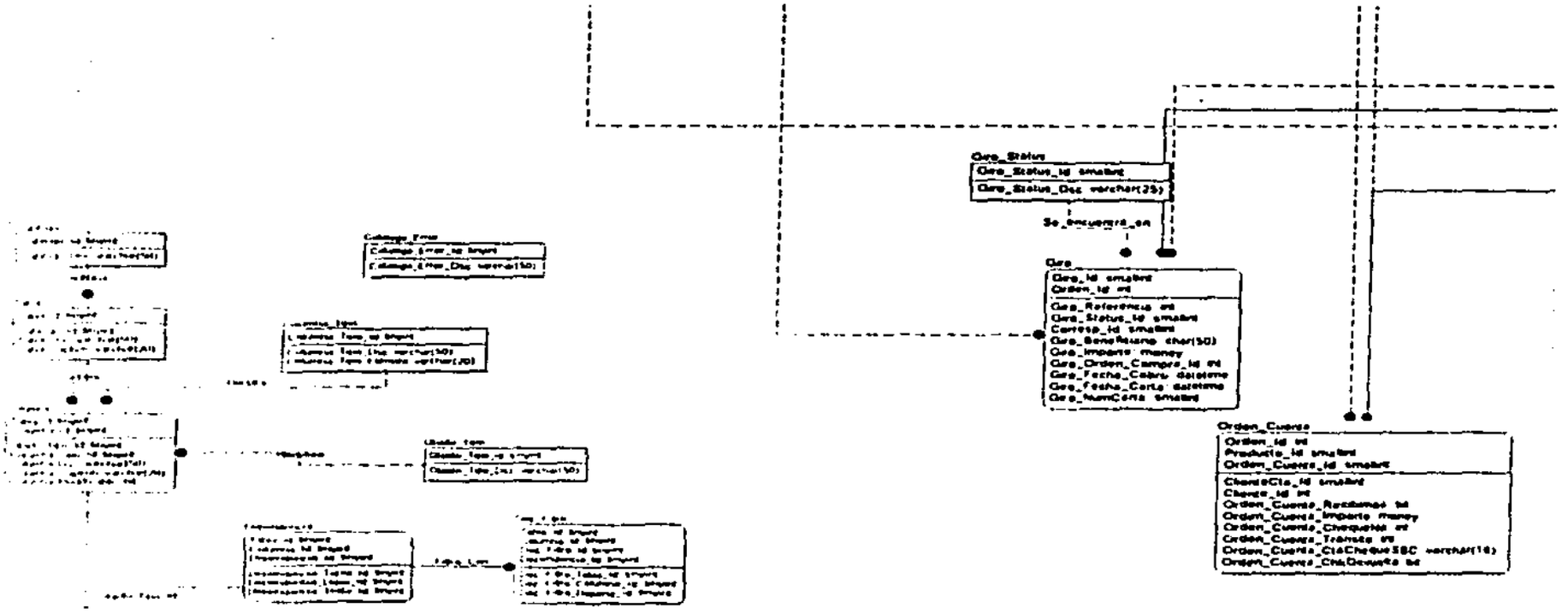
ANEXOS

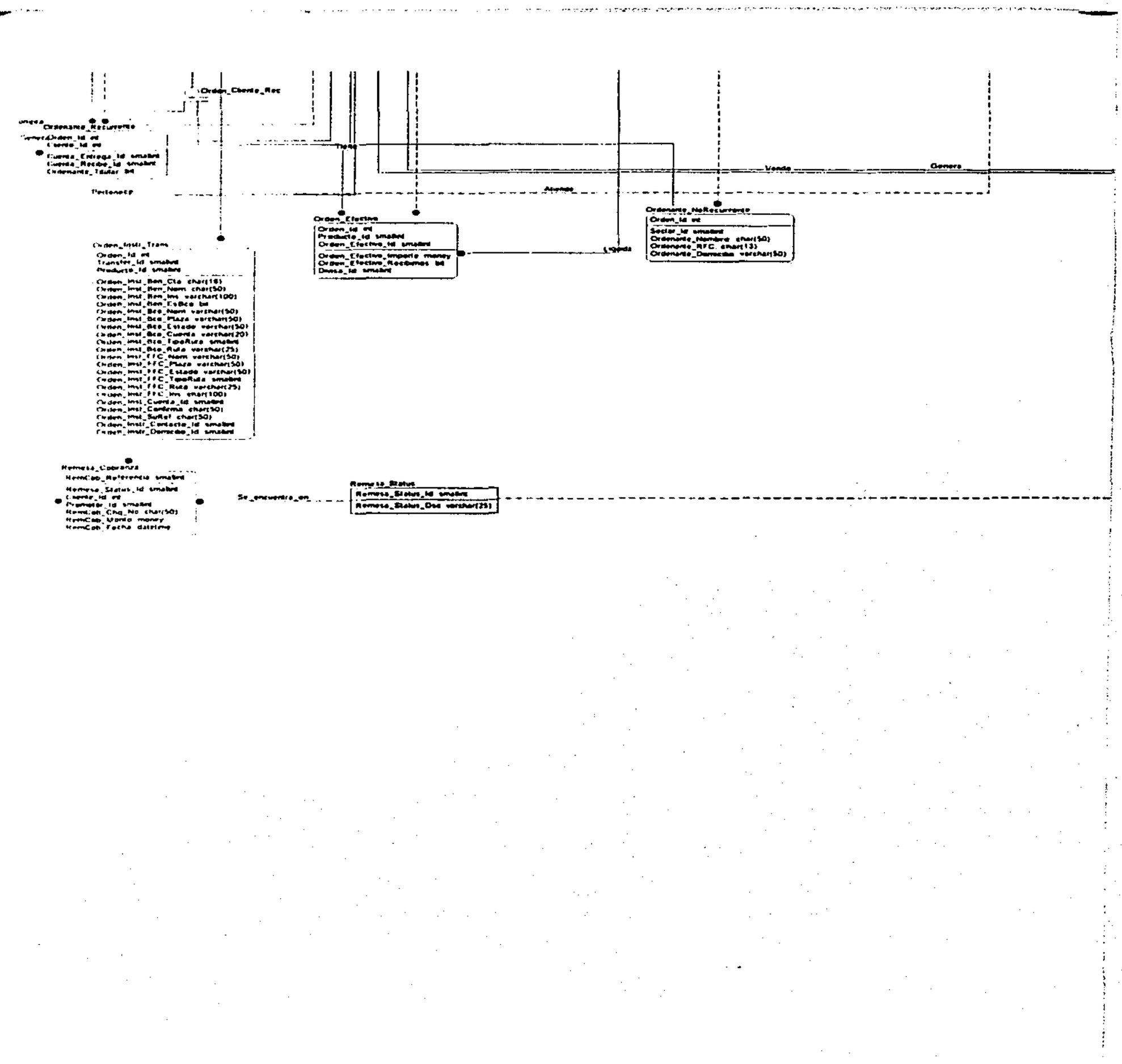
CONTADO

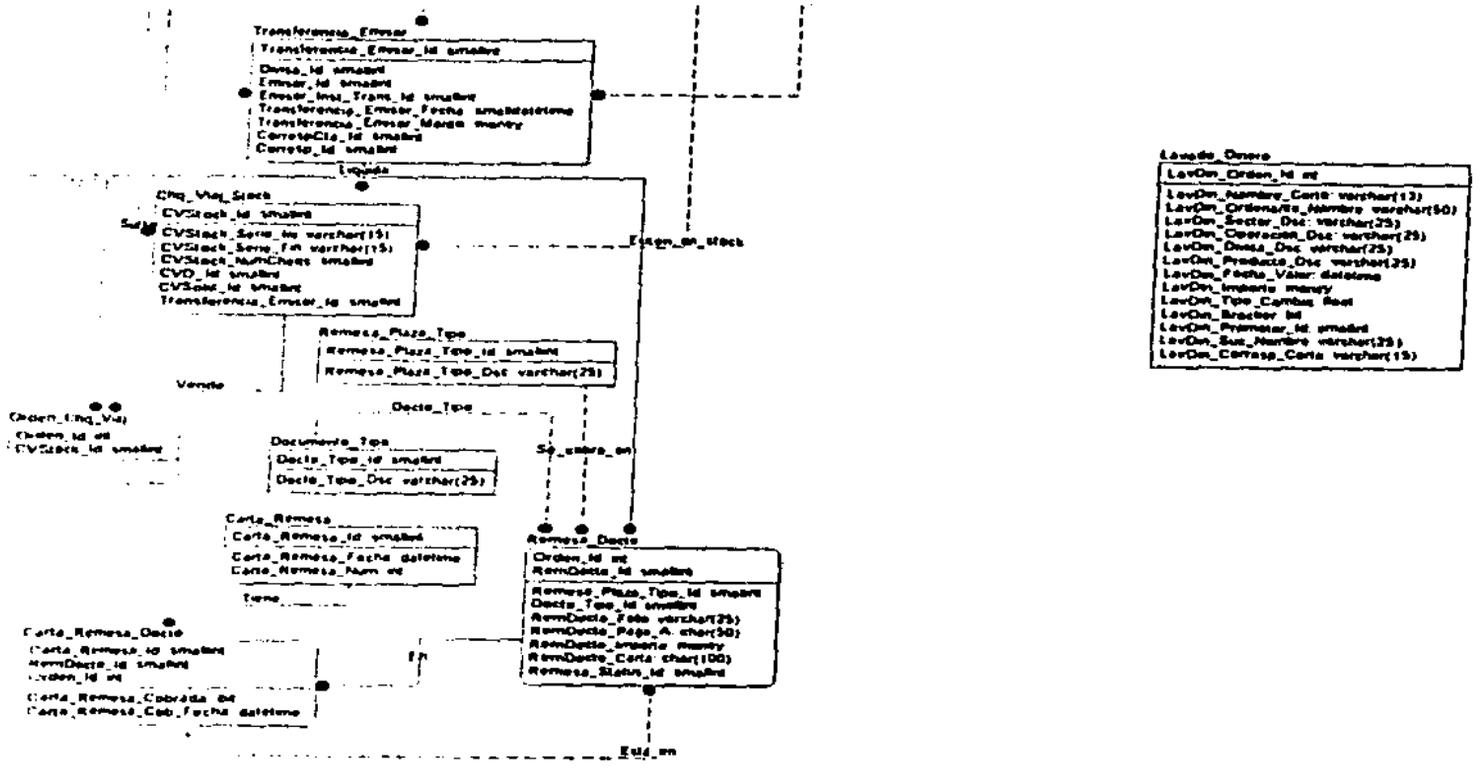


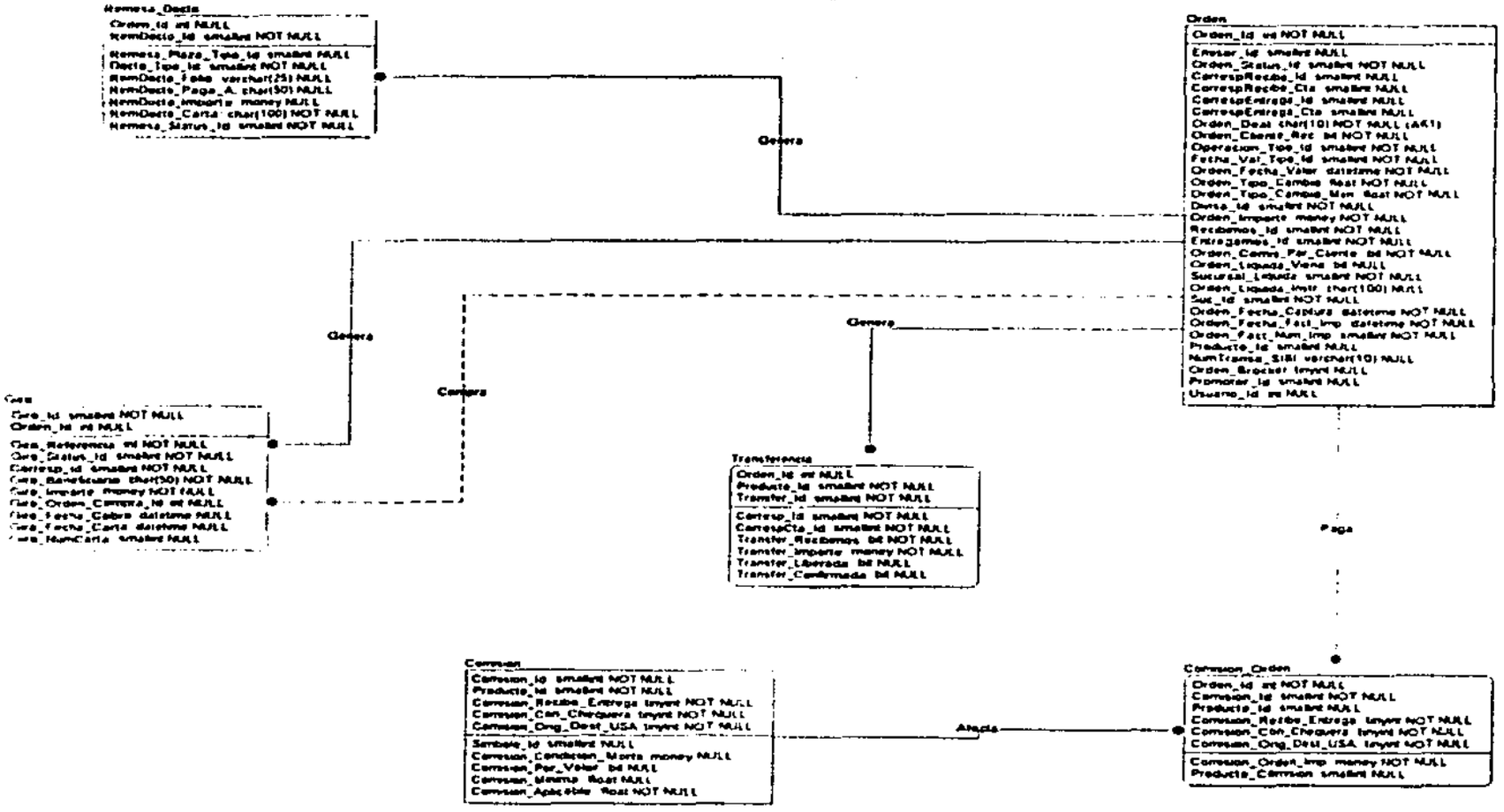



Efectivo_Stock
Suc_Id smallint
EfectDenom_Id smallint
Usuario_Id int
EfectStock_Cant int
Tiene
Efectivo_Denom
EfectDenom_Id smallint
EfectDenom_Valor money
Divisa_Id smallint
En_caja
Moneda_Stock
Suc_Id smallint
Divisa_Id smallint
Usuario_Id int
MonedaStock_Cant money
Divisa
Divisa_Id smallint
Pais_Id smallint
Divisa_Siglas char(3)
Divisa_Obs varchar(25)
Divisa_Controlar smallint
Divisa_Id_SIBI char(3)
Pide
Tiene_tipos_de_cambio
Valores
Se_Realiza_en
Se_realiza_en
Tiene_denominaciones
SaldoCHV
Suc_Id smallint
Divisa_Id smallint
Emisor_Id smallint
SaldoCHV_Id smallint
SaldoCHV_Fecha datetime
SaldoCHV_Inicial money
SaldoCHV_Ingreso money
SaldoCHV_Egreso money
SaldoCHV_Final money
Esta_en
Pertenece
Emisor
Emisor_Id smallint
Emisor_Obs varchar(25)
Emisor_Cuenta varchar(12)
Tiene
Chq_Vial_Denom
CVD_Id smallint
Divisa_Id smallint
Emisor_Id smallint
CVD_Denom money
Emisor_Inst_Transfer
Emisor_Id smallint
Emisor_Inst_Trans_Id smallint
Emisor_Inst_Trans_Cuenta varchar(25) (AK1)
Emisor_Inst_Trans_Cuenta char(18)
Emisor_Inst_Trans_Banco varchar(50)
Emisor_Inst_Trans_BcoPlaza varchar(50)
Emisor_Inst_Trans_BcoEstado varchar(50)
Emisor_Inst_Trans_BcoCuenta varchar(20)
Emisor_Inst_Trans_BcoTipoRuta smallint
Emisor_Inst_Trans_BcoNumRuta varchar(25)
Emisor_Inst_Trans_BcoObs varchar(50)
Emisor_Inst_Trans_Inst varchar(100)
Emisor_Inst_Trans_FFCNombre varchar(50)
Emisor_Inst_Trans_FFCPlaza varchar(50)
Emisor_Inst_Trans_FFCEstado varchar(50)
Emisor_Inst_Trans_FFCTipoRuta smallint
Emisor_Inst_Trans_FFCNumRuta varchar(25)
Emisor_Inst_Trans_FFCInst varchar(100)
Emisor_Inst_Trans_Tipo bit
Pertenece

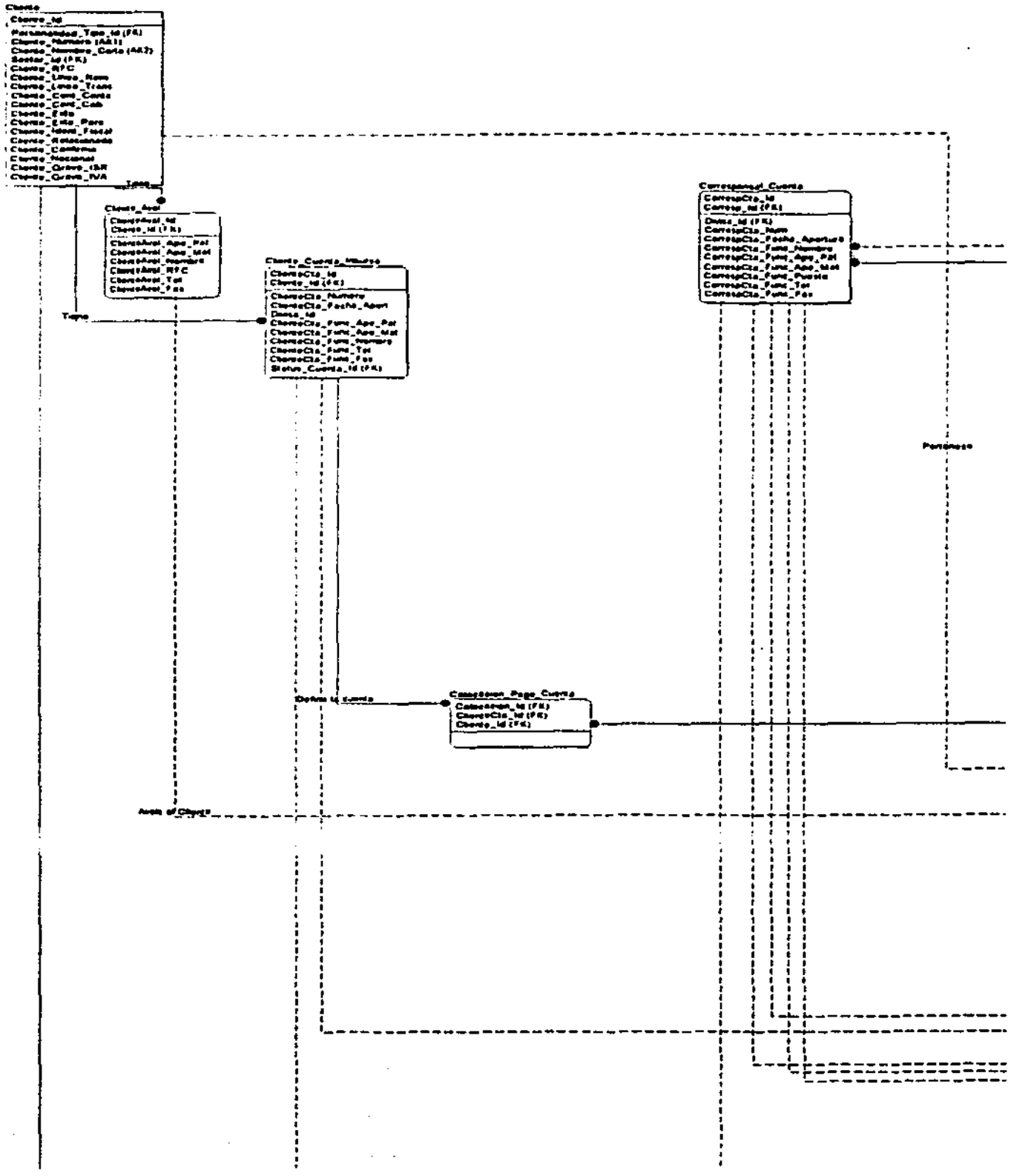

Giro_Status
Giro_Status_Id smallint
Giro_Status_Des varchar(25)
Se_encuentra_en
Giro
Giro_Id smallint
Orden_Id int
Giro_Referencia int
Giro_Status_Id smallint
Corresp_Id smallint
Giro_BeneficiarioP char(50)
Giro_Importe money
Giro_Orden_Compra_Id int
Giro_Fecha_Cobro datetime
Giro_Fecha_Corte datetime
Giro_NumCorte smallint
Orden_Cuenta
Orden_Id int
Producto_Id smallint
Orden_Cuenta_Id smallint
ClienteCta_Id smallint
Cliente_Id int
Orden_Cuenta_Rendimiento int
Orden_Cuenta_Importe money
Orden_Cuenta_Chequera int
Orden_Cuenta_Transito int
Orden_Cuenta_CtaChequeSBC varchar(18)
Orden_Cuenta_CtaDeposito int

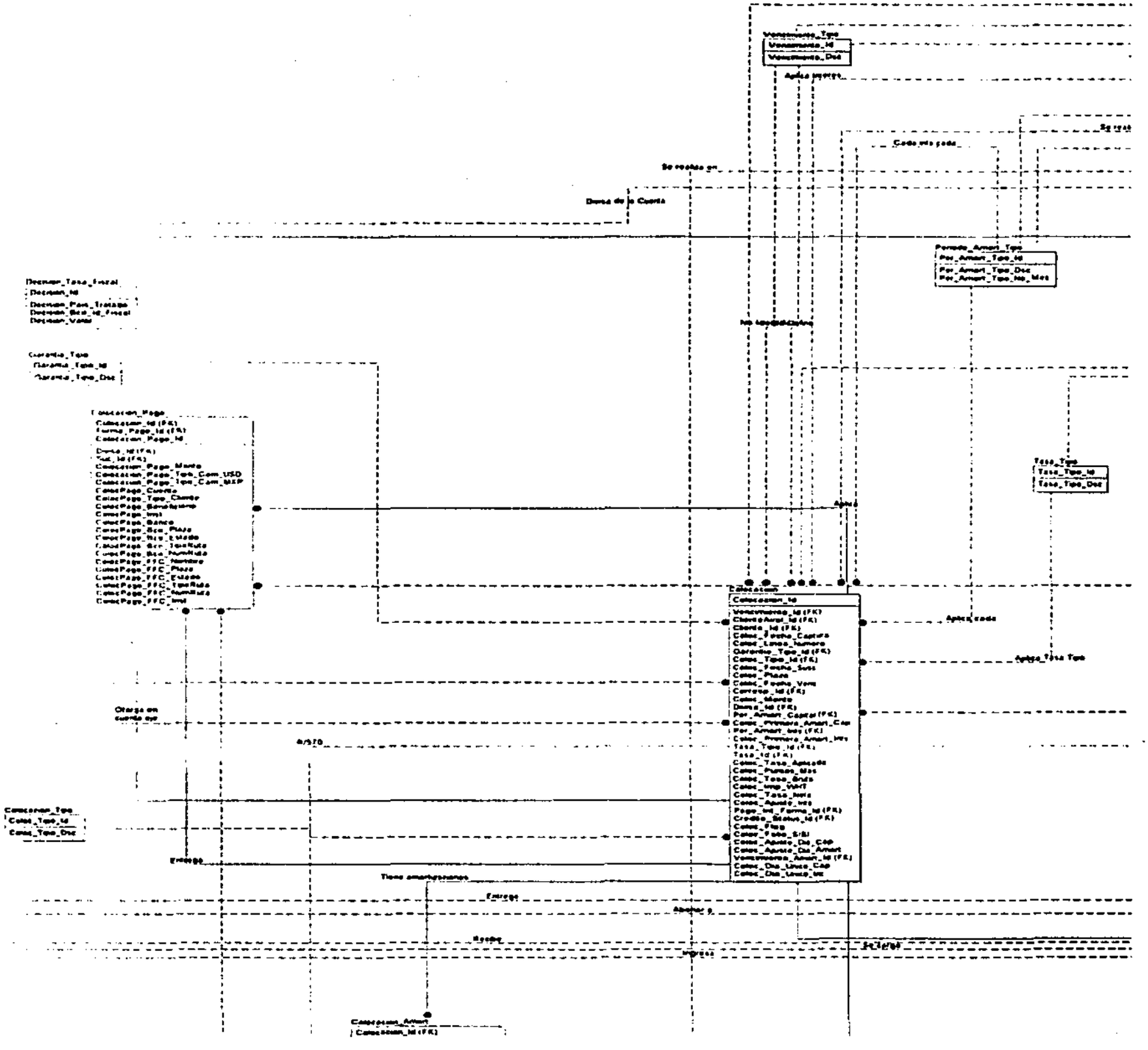

Orden_Cliente_Rec
Tiene
Vende
Genera
Pertenece
Atiende
Orden_Efectivo
Orden_Id int
Producto_Id smallint
Orden_Efectivo_Id smallint
Orden_Efectivo_Importe money
Orden_Efectivo_Recibimos bit
Divisa_Id smallint
Liquida
Ordenante_NoRecurrente
Orden_Id int
Sector_Id smallint
Ordenante_Nombre char(50)
Ordenante_RFC char(13)
Ordenante_Domicilio varchar(50)
Orden_Inst_Trans
Orden_Id int
Transfer_Id smallint
Producto_Id smallint
Orden_Inst_Ben_Cta char(18)
Orden_Inst_Ben_Nom char(50)
Orden_Inst_Ben_Ins varchar(100)
Orden_Inst_Ben_EsBco bit
Orden_Inst_Bco_Nom varchar(50)
Orden_Inst_Bco_Plaza varchar(50)
Orden_Inst_Bco_Estado varchar(50)
Orden_Inst_Bco_Cuenta varchar(20)
Orden_Inst_Bco_TipoRuta smallint
Orden_Inst_Bco_Ruta varchar(25)
Orden_Inst_FFC_Nom varchar(50)
Orden_Inst_FFC_Plaza varchar(50)
Orden_Inst_FFC_Estado varchar(50)
Orden_Inst_FFC_TipoRuta smallint
Orden_Inst_FFC_Ruta varchar(25)
Orden_Inst_FFC_Ins char(100)
Orden_Inst_Cuenta_Id smallint
Orden_Inst_Conferma char(50)
Orden_Inst_SuRef char(50)
Orden_Inst_Contacto_Id smallint
Orden_Inst_Domicilio_Id smallint
Remesa_Cobranza
RemCob_Referencia smallint
Remesa_Status_Id smallint
Cuenta_Id int
Producto_Id smallint
RemCob_Chq_No char(50)
RemCob_Monto money
RemCob_Fecha datetime
Se_encuentra_en
Remesa_Status
Remesa_Status_Id smallint
Remesa_Status_Des varchar(25)

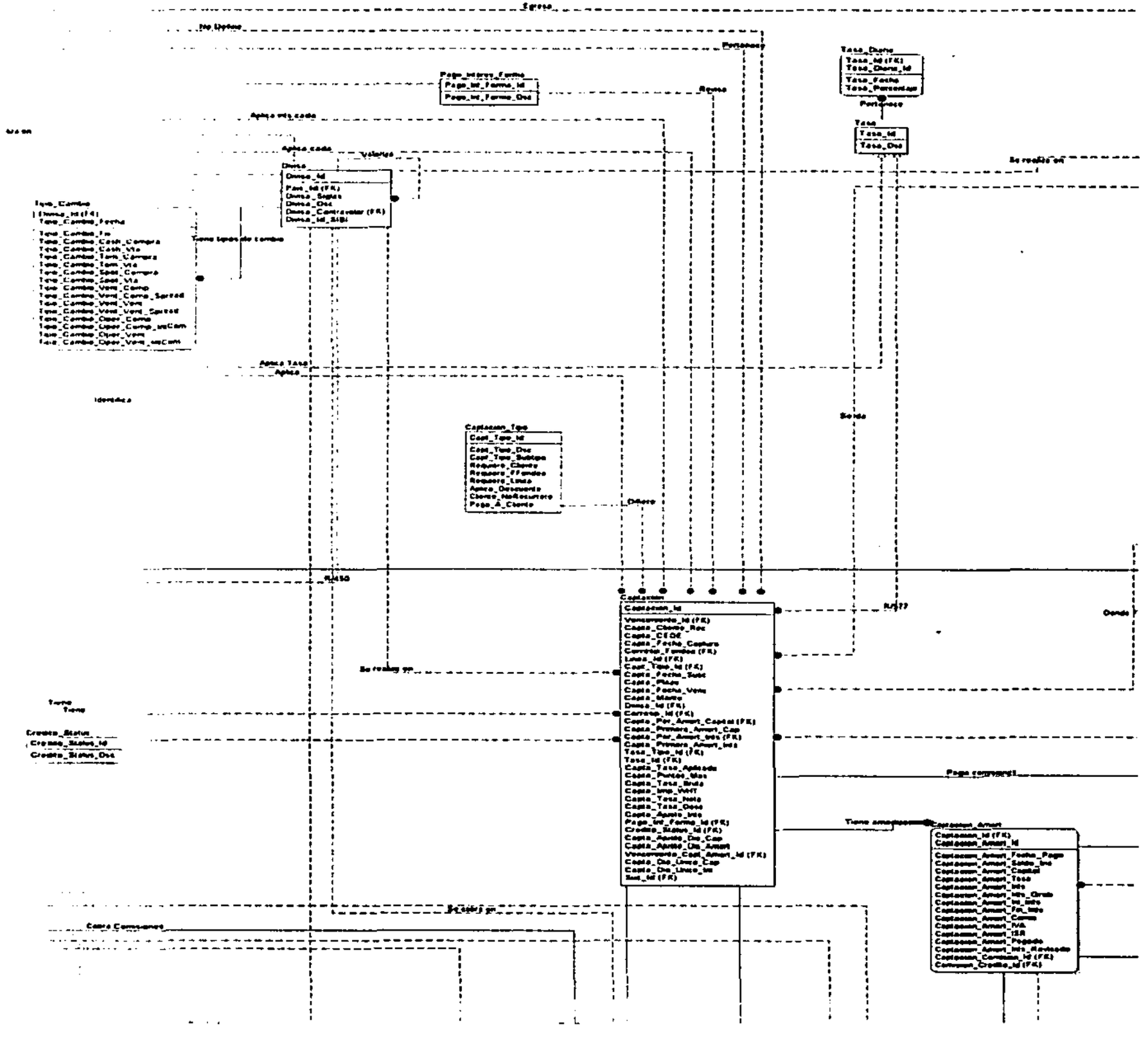

Transferencia_Emisor
Transferencia_Emisor_Id smallint
Divisa_Id smallint
Emisor_Id smallint
Emisor_Inst_Trans_Id smallint
Transferencia_Emisor_Fecha smalldatetime
Transferencia_Emisor_Monto money
CorresplCta_Id smallint
Corresp_Id smallint
Liquida
Chq_Vial_Stock
CVStock_Id smallint
CVStock_Serie_Ini varchar(15)
CVStock_Serie_Fin varchar(15)
CVStock_NumCheque smallint
CVD_Id smallint
CVSolic_Id smallint
Transferencia_Emisor_Id smallint
Sale
Existen_en_stock
Remesa_Plaza_Tipo
Remesa_Plaza_Tipo_Id smallint
Remesa_Plaza_Tipo_Dsc varchar(25)
Vende
Docto_Tipo
Documento_Tipo
Docto_Tipo_Id smallint
Docto_Tipo_Dsc varchar(25)
Se_cobra_en
Carta_Remesa
Carta_Remesa_Id smallint
Carta_Remesa_Fecha datetime
Carta_Remesa_Num int
Tiene
Remesa_Docto
Orden_Id int
RemDocto_Id smallint
Remesa_Plaza_Tipo_Id smallint
Docto_Tipo_Id smallint
RemDocto_Fdoc varchar(25)
RemDocto_Pago_A char(50)
RemDocto_Importe money
RemDocto_Carta char(100)
Remesa_Status_Id smallint
Carta_Remesa_Docto
Carta_Remesa_Id smallint
RemDocto_Id smallint
Orden_Id int
Carta_Remesa_Cobrada bit
Carta_Remesa_Cob_Fecha datetime
Es
Esta_en
Lavado_Dinero
LavDin_Orden_Id int
LavDin_Nombre_Carta varchar(13)
LavDin_Ordena/da_Nombre varchar(50)
LavDin_Sector_Dsc varchar(25)
LavDin_Operacion_Dsc varchar(25)
LavDin_Divisa_Dsc varchar(25)
LavDin_Producto_Dsc varchar(25)
LavDin_Fecha_Valor datetime
LavDin_Importe money
LavDin_Tipo_Cambio float
LavDin_Broker bit
LavDin_Promotor_Id smallint
LavDin_Sus_Nombre varchar(25)
LavDin_Corresp_Carta varchar(15)

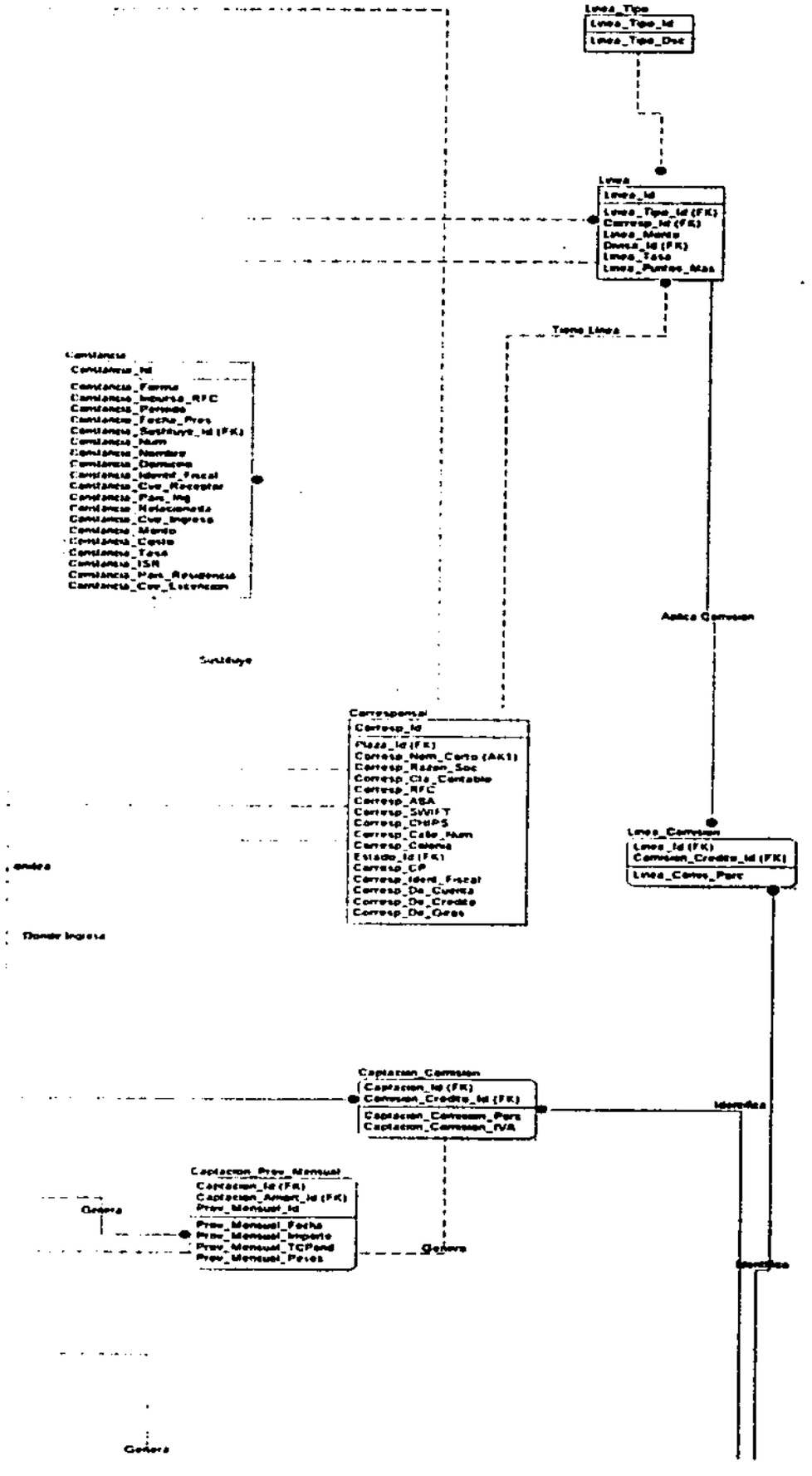

Genera
Genera
Genera
Genera
Paga
Afecta

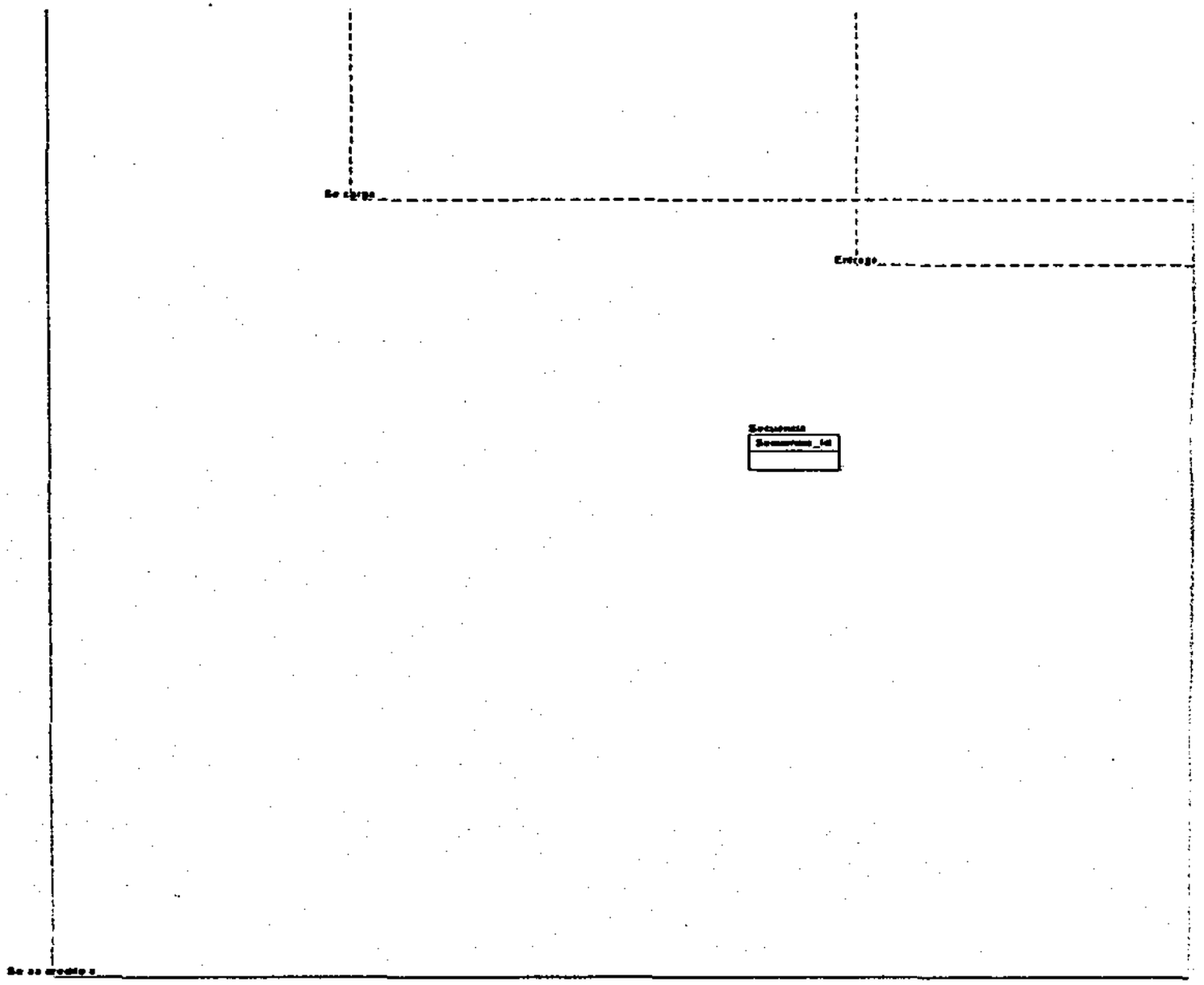
CREDITO

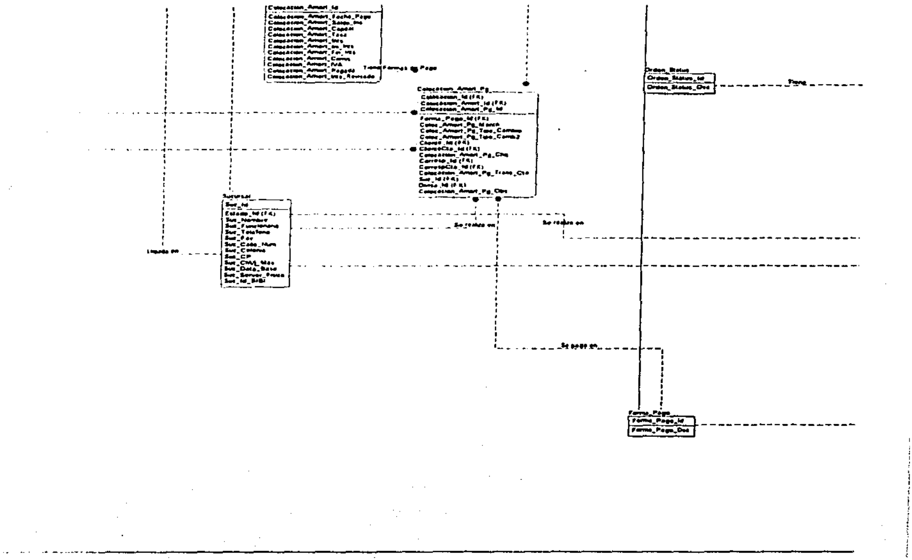

Cliente
Cliente_Id
Cliente_Numero (AK1)
Cliente_Nombre_Corto (AK2)
Cliente_RFC
Cliente_Limite_Num
Cliente_Limite_Trans
Cliente_Cont_Conta
Cliente_Cont_Cob
Cliente_Edo
Cliente_Edo_Pers
Cliente_Ident_Fiscal
Cliente_Confirma
Cliente_Nacional
Cliente_Grava_ISR
Cliente_Grava_IVA
Tiene
Cliente_Aval
ClienteAval_Id
Cliente_Id (FK)
ClienteAval_Apep_Pat
ClienteAval_Apep_Mat
ClienteAval_Nombre
ClienteAval_RFC
ClienteAval_Tel
ClienteAval_Fax
Cliente_Cuenta_Ahorro
ClienteCta_Id
Cliente_Id (FK)
ClienteCta_Numero
ClienteCta_Fecha_Apert
Divisa_Id
ClienteCta_Fuma_Apep_Pat
ClienteCta_Fuma_Apep_Mat
ClienteCta_Fuma_Nombre
ClienteCta_Fuma_Tel
ClienteCta_Fuma_Fax
Status_Cuenta_Id (FK)
Tiene
Corresponsal_Cuenta
CorrespCta_Id
Corresp_Id (FK)
Divisa_Id (FK)
CorrespCta_Num
CorrespCta_Fecha_Apertura
CorrespCta_Fuma_Nombre
CorrespCta_Fuma_Apep_Pat
CorrespCta_Fuma_Apep_Mat
CorrespCta_Fuma_Puesto
CorrespCta_Fuma_Tel
CorrespCta_Fuma_Fax
Pertenece
Divisa Id Cuenta
Colectaron_Pago_Cuenta
Colectaron_Id (FK)
ClienteCta_Id (FK)
Cliente_Id (FK)
Avala al Cliente


Vencimiento_Tipo
Vencimiento_Id
Vencimiento_Dsc
Aplica interes
Se realiza en
Cuenta de la Cuenta
Periodo_Amort_Tipo
Per_Amort_Tipo_Id
Per_Amort_Tipo_Dsc
Per_Amort_Tipo_No_Mes
Tasa_Tipo
Tasa_Tipo_Id
Tasa_Tipo_Dsc
Aplica
Aplica tasa
Aplica Tasa Tipo
Colocacion
Colocacion_Id
Vencimiento_Id (FK)
ClienteAsal_Id (FK)
Cliente_Id (FK)
Colocacion_Tipo
Colocacion_Id
Colocacion_Tipo_Id
Colocacion_Dsc
Entrega
Tiene amortizaciones
Abonos a
Ingresa
Colocacion_Amort
Colocacion_Id (FK)

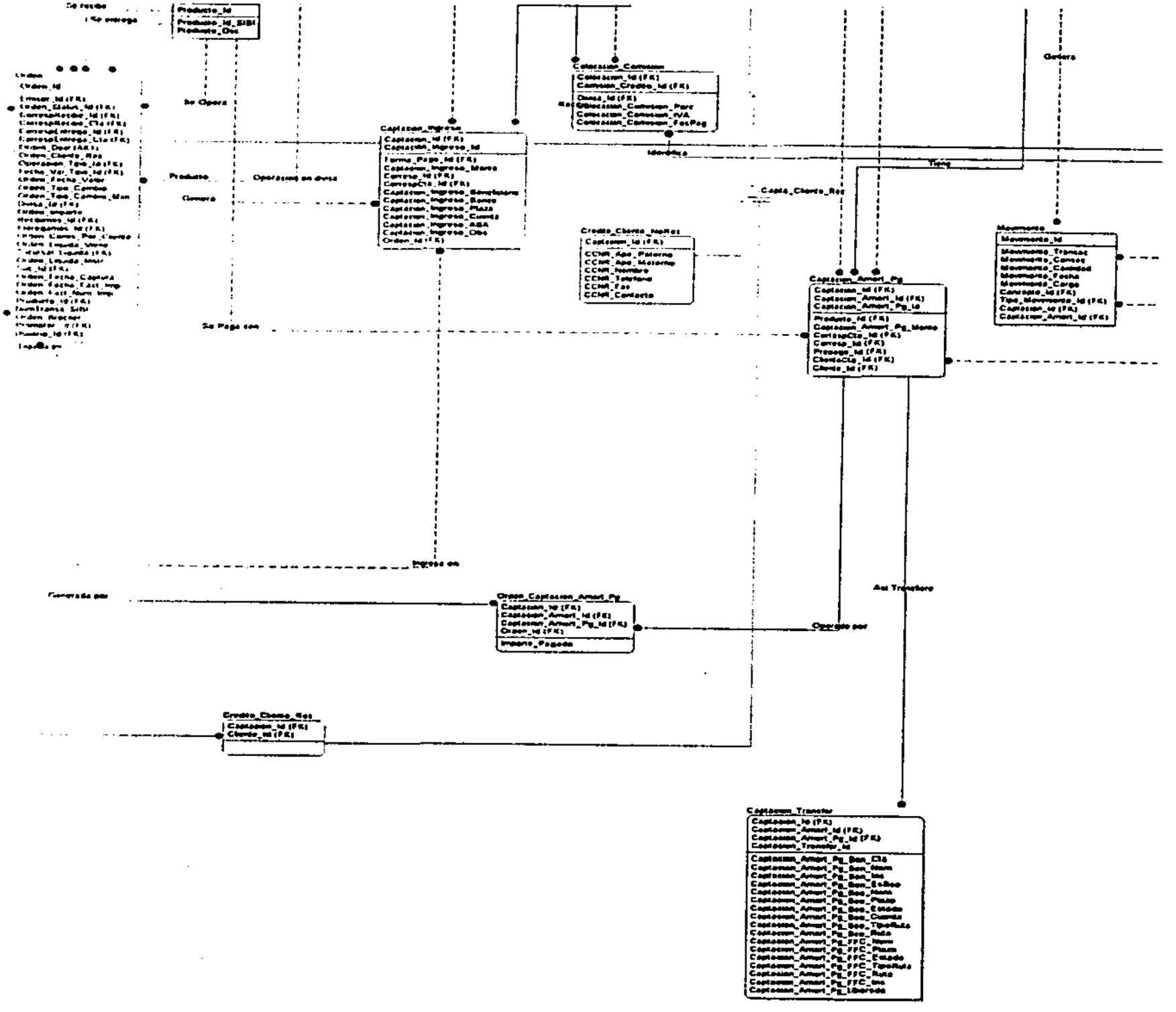

Divisa
Divisa_Id
Divisa_Siglas
Divisa_Desc
Tasa_Cambio
Tasa_Cambio_Fecha
Tasa_Cambio_Fin
Tasa_Cambio_Cash_Compra
Tasa_Cambio_Cash_Vta
Tasa_Cambio_Tom_Compra
Tasa_Cambio_Tom_Vta
Tasa_Cambio_Spot_Compra
Tasa_Cambio_Spot_Vta
Tasa_Cambio_Vom_Comp
Tasa_Cambio_Vom_Comp_Spread
Tasa_Cambio_Vom_Vom
Tasa_Cambio_Vom_Vom_Spread
Tasa_Cambio_Oper_Comp
Tasa_Cambio_Oper_Comp_wCom
Tasa_Cambio_Oper_Vom
Tasa_Cambio_Oper_Vom_wCom
Captacion_Tipo
Capt_Tipo_Id
Capt_Tipo_Des
Capt_Tipo_Subtipo
Requiere_Cliente
Requiere_Fondeo
Requiere_Linea
Aplica_Descuento
Cliente_NoResidente
Pago_A_Cliente
Captacion
Captacion_Id
Vencimiento_Id (FK)
Capta_Cliente_Rec
Capta_CEDE
Capta_Fecha_Captura
Corredor_Fondeo (FK)
Linea_Id (FK)
Capt_Tipo_Id (FK)
Capta_Fecha_Sust
Capta_Plazo
Capta_Fecha_Vcto
Capta_Monto
Divisa_Id (FK)
Corredor_Id (FK)
Capta_Por_Amort_Capital (FK)
Capta_Primera_Amort_Cap
Capta_Por_Amort_Int (FK)
Capta_Primera_Amort_Int
Tasa_Tipo_Id (FK)
Tasa_Id (FK)
Capta_Tasa_Aplicada
Capta_Puntos_Mas
Capta_Tasa_Bruta
Capta_Imp_ISR
Capta_Tasa_Neta
Capta_Tasa_Obsv
Capta_Aporte_Ini
Pago_Int_Forma_Id (FK)
Credito_Status_Id (FK)
Capta_Aporte_Dia_Cap
Capta_Aporte_Dia_Amort
Vencimiento_Capt_Amort_Id (FK)
Capta_Dia_Unico_Cap
Capta_Dia_Unico_Int
Sect_Id (FK)
Captacion_Amort
Captacion_Id (FK)
Captacion_Amort_Id
Captacion_Amort_Fecha_Pago
Captacion_Amort_Saldo_Ini
Captacion_Amort_Capital
Captacion_Amort_Tasa
Captacion_Amort_Int
Captacion_Amort_Int_Gravable
Captacion_Amort_Int_Ret
Captacion_Amort_Por_Ret
Captacion_Amort_Comis
Captacion_Amort_IVA
Captacion_Amort_ISR
Captacion_Amort_Pagado
Captacion_Amort_Int_Reinvertido
Captacion_Corredor_Id (FK)
Corresp_Credito_Id (FK)
Pago_Intereses_Forma
Pago_Int_Forma_Id
Pago_Int_Forma_Des
Tasa_Diaria
Tasa_Id (FK)
Tasa_Diaria_Id
Tasa_Fecha
Tasa_Porcentaje
Tasa
Tasa_Id
Tasa_Des
Creditos_Status
Credito_Status_Id
Credito_Status_Des

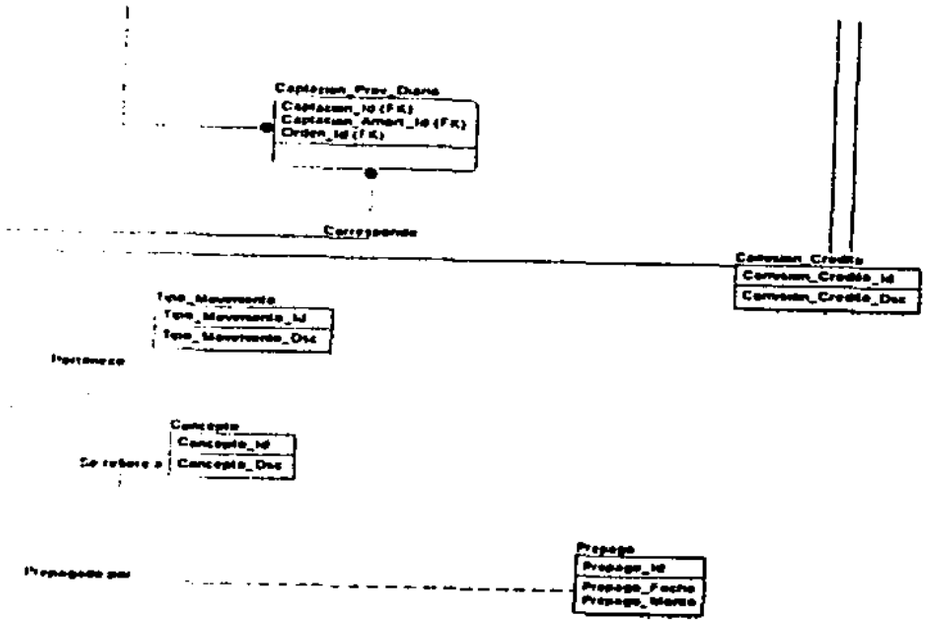

Linea_Tipo
Linea_Tipo_Id
Linea_Tipo_Des
Linea
Linea_Id
Linea_Tipo_Id (FK)
Corresp_Id (FK)
Linea_Monto
Divisa_Id (FK)
Linea_Tasa
Linea_Plazos_Max
Tiene Linea
Constancia
Constancia_Id
Constancia_Forma
Constancia_Importe_RFC
Constancia_Periodo
Constancia_Fecha_Pres
Constancia_Sustituye_Id (FK)
Constancia_Num
Constancia_Nombre
Constancia_Domicilio
Constancia_Ident_Fiscal
Constancia_Cve_Receptor
Constancia_Pais_Imp
Constancia_Relacionada
Constancia_Cve_Ingreso
Constancia_Monto
Constancia_Costo
Constancia_Tasa
Constancia_ISR
Constancia_Pais_Residencia
Constancia_Cve_Exencion
Sustituye
Aplica Comision
Corresponsal
Corresp_Id
Plaza_Id (FK)
Corresp_Nom_Corto (AK1)
Corresp_Razon_Soc
Corresp_Cta_Contable
Corresp_RFC
Corresp_ABA
Corresp_SWIFT
Corresp_ChIPS
Corresp_Cabe_Num
Corresp_Colonia
Estado_Id (FK)
Corresp_CP
Corresp_Ident_Fiscal
Corresp_De_Cuenta
Corresp_De_Credito
Corresp_De_Gral
Linea_Comision
Linea_Id (FK)
Comision_Credito_Id (FK)
Linea_Comi_Porc
Captacion_Comision
Captacion_Id (FK)
Comision_Credito_Id (FK)
Captacion_Comision_Porc
Captacion_Comision_IVA
Identifica
Captacion_Prev_Mensual
Comision_Id (FK)
Captacion_Arrend_Id (FK)
Prev_Mensual_Id
Prev_Mensual_Fecha
Prev_Mensual_Importe
Prev_Mensual_TCPond
Prev_Mensual_Pesos
Genera
Genera
Genera


Se cierra
Entrega
Se acredita

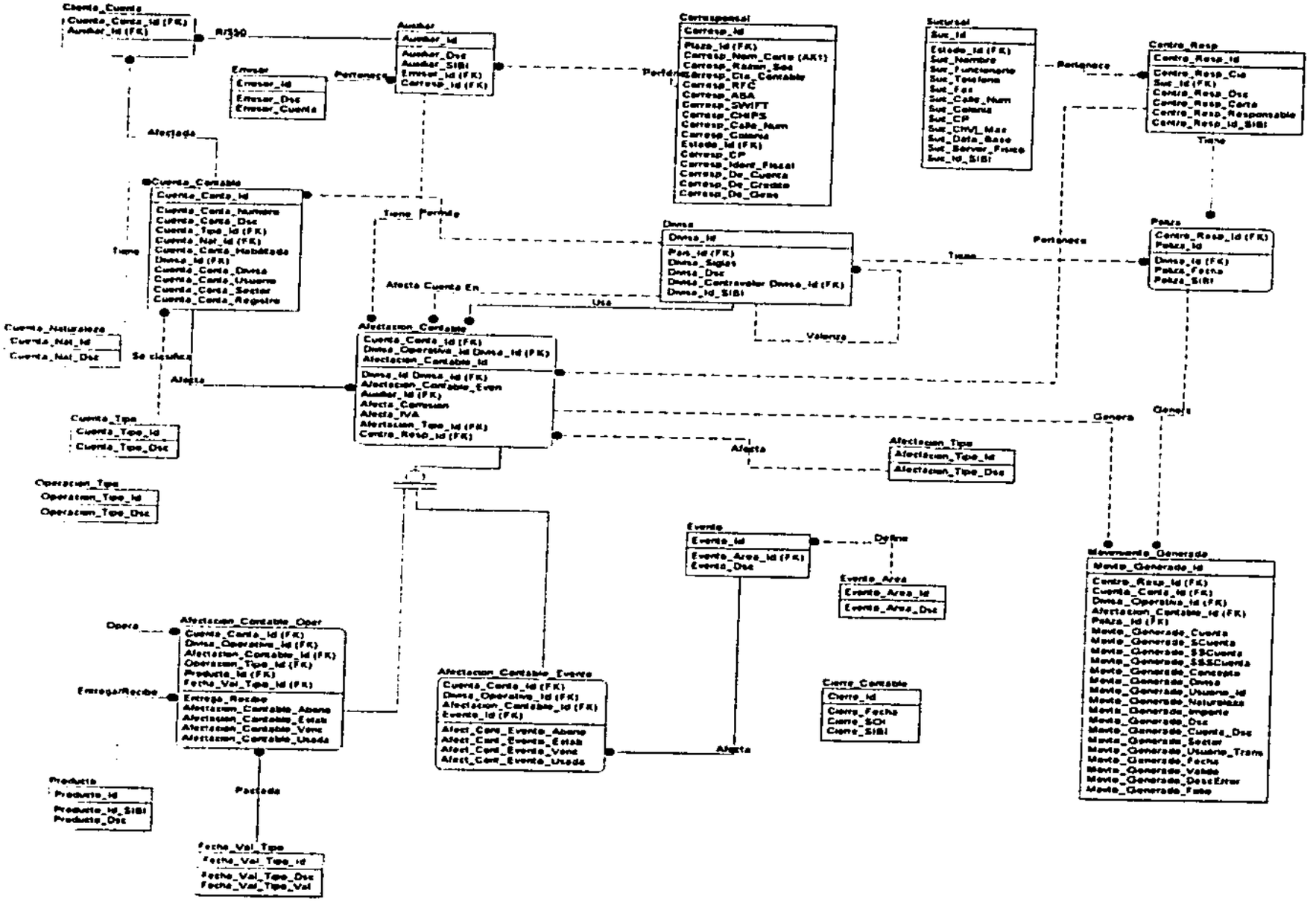


Tiene Formas de Pago
Colocacion_Amort_Pg
Sucursal
Suc_Id
Suc_Telefono
Suc_Fax
Suc_CP
Suc_Data_Base
Se realiza en
Se realiza en
Se paga en
Orden_Status
Orden_Status_Id
Orden_Status_Des
Tiene
Forma_Pago
Forma_Pago_Id
Forma_Pago_Des

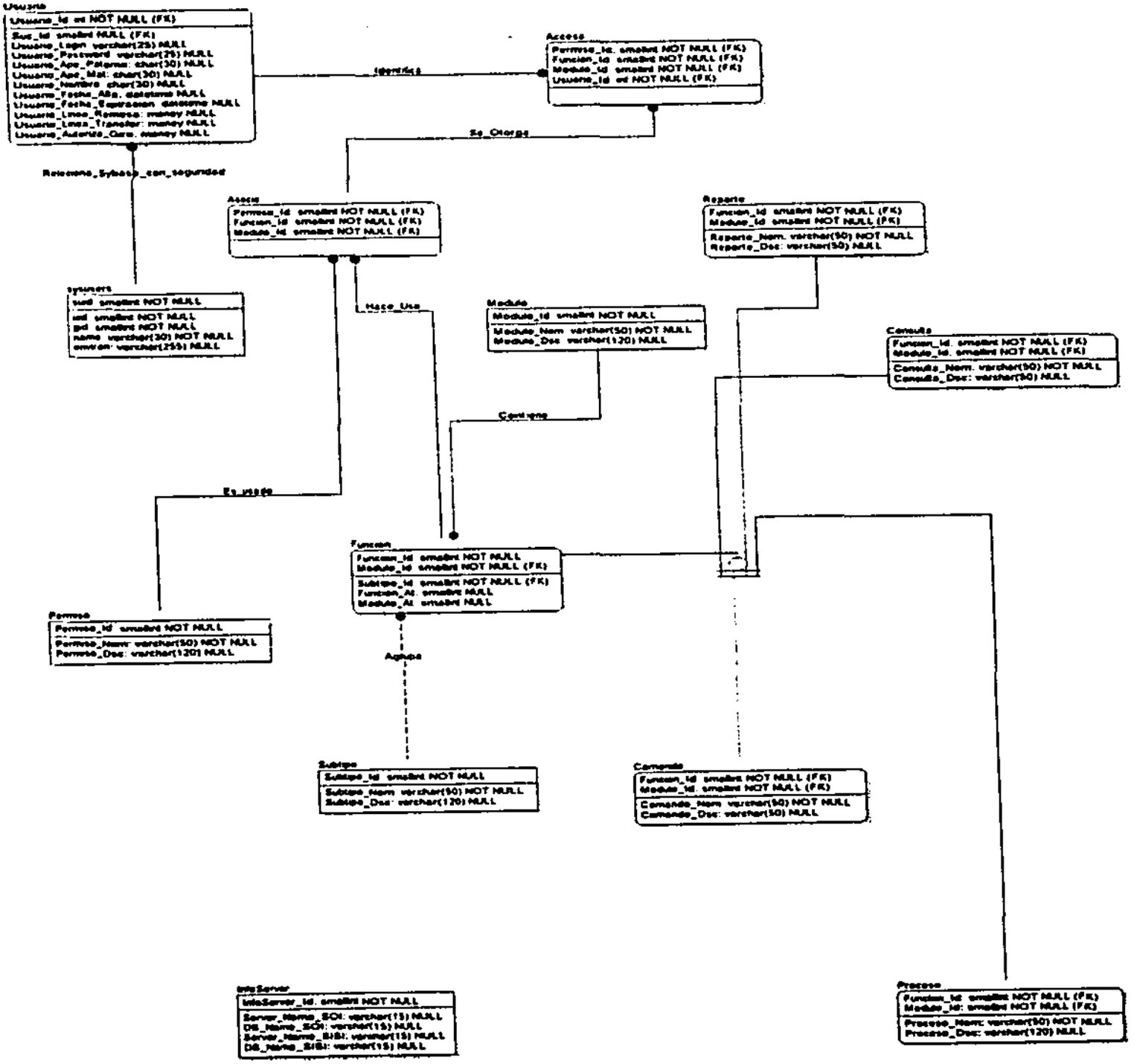

Se recibe
Se entrega
Producto_Id
Producto_Id_SIBI
Producto_Des
Se Opera
Captacion_Ingreso
Captacion_Id (FK)
Captacion_Ingreso_Id
Forma_Pago_Id (FK)
Captacion_Ingreso_Monto
Corresp_Id (FK)
CorrespCta_Id (FK)
Captacion_Ingreso_Beneficiario
Captacion_Ingreso_Banco
Captacion_Ingreso_Plaza
Captacion_Ingreso_Cuenta
Captacion_Ingreso_ABA
Captacion_Ingreso_Obs
Orden_Id (FK)
Captacion_Comision
Comision_Id (FK)
Comision_Credito_Id (FK)
Oficina_Id (FK)
Oficina_Id (FK)
Captacion_Comision_Pers
Captacion_Comision_IVA
Captacion_Comision_FecPag
Genera
Identifica
Tiene
Produce
Operacion en divisa
Genera
Costa_Cliente_Rec
Credito_Cliente_NoRec
Captacion_Id (FK)
CCNR_Ape_Paterno
CCNR_Ape_Materno
CCNR_Nombre
CCNR_Telefono
CCNR_Fax
CCNR_Contacto
Movimiento
Movimiento_Id
Movimiento_Tipo
Movimiento_Concepto
Movimiento_Cantidad
Movimiento_Fecha
Movimiento_Cargo
Cancepto_Id (FK)
Tipo_Movimiento_Id (FK)
Captacion_Id (FK)
Captacion_Amort_Pg_Id (FK)
Se Paga con
Captacion_Amort_Pg
Captacion_Id (FK)
Captacion_Amort_Id (FK)
Captacion_Amort_Pg_Id
Producto_Id (FK)
Captacion_Amort_Pg_Monto
CorrespCta_Id (FK)
Corresp_Id (FK)
Presupp_Id (FK)
ClienteCta_Id (FK)
Cliente_Id (FK)
Ingreso en
Generada por
Orden_Captacion_Amort_Pg
Captacion_Id (FK)
Captacion_Amort_Id (FK)
Captacion_Amort_Pg_Id (FK)
Orden_Id (FK)
Importe_Pagado
Operado por
Ase Transfiere
Credito_Cliente_Rec
Captacion_Id (FK)
Cliente_Id (FK)
Captacion_Transfer
Captacion_Id (FK)
Captacion_Amort_Id (FK)
Captacion_Amort_Pg_Id (FK)
Captacion_Transfer_Id
Captacion_Amort_Pg_Ben_Cta
Captacion_Amort_Pg_Ben_Nom
Captacion_Amort_Pg_Ben_Ide
Captacion_Amort_Pg_Ben_EnBco
Captacion_Amort_Pg_Ben_Num
Captacion_Amort_Pg_Ben_Plaza
Captacion_Amort_Pg_Ben_Estado
Captacion_Amort_Pg_Ben_Cuenta
Captacion_Amort_Pg_Ben_TipoRuta
Captacion_Amort_Pg_Ben_Ruta
Captacion_Amort_Pg_FFC_Nom
Captacion_Amort_Pg_FFC_Plaza
Captacion_Amort_Pg_FFC_Estado
Captacion_Amort_Pg_FFC_TipoRuta
Captacion_Amort_Pg_FFC_Ruta
Captacion_Amort_Pg_FFC_Ide
Captacion_Amort_Pg_Liberada


Captacion_Prov_Chans
Captacion_Id (FK)
Captacion_Amort_Id (FK)
Origen_Id (FK)
Corresponde
Comision_Credito
Comision_Credito_Id
Comision_Credito_Dsc
Tipo_Movimiento
Tipo_Movimiento_Id
Tipo_Movimiento_Dsc
Pertenece
Concepto
Concepto_Id
Concepto_Dsc
Se refiere a
Propaga
Propaga_Id
Propaga_Fecha
Propaga_Monto
Propagada por

CONTABILIDAD


Cuenta_Cuenta
Cuenta_Conta_Id (FK)
Auxiliar_Id (FK)
Rige
Auxiliar
Auxiliar_Id
Auxiliar_Dsc
Auxiliar_SIBI
Emisor_Id (FK)
Corresp_Id (FK)
Pertenece
Emisor
Emisor_Id
Emisor_Dsc
Emisor_Cuenta
Afectada
Corresponsal
Corresp_Id
Plaza_Id (FK)
Corresp_Num_Cuenta (AK1)
Corresp_Razon_Soc
Corresp_Dia_Contable
Corresp_RFC
Corresp_ABA
Corresp_SWIFT
Corresp_CHIPS
Corresp_Calle_Num
Corresp_Colonia
Estado_Id (FK)
Corresp_CP
Corresp_Idioma_Fiscal
Corresp_De_Cuenta
Corresp_De_Credito
Corresp_De_Cargo
Sucursal
Suc_Id
Estado_Id (FK)
Suc_Nombre
Suc_Funcionario
Suc_Telefono
Suc_Fax
Suc_Calle_Num
Suc_Colonia
Suc_CP
Suc_Chvl_Mae
Suc_Data_Base
Suc_Server_Prince
Suc_Id_SIBI
Pertenece
Centro_Resp
Centro_Resp_Id
Centro_Resp_Cve
Suc_Id (FK)
Centro_Resp_Dsc
Centro_Resp_Corto
Centro_Resp_Responsable
Centro_Resp_Id_SIBI
Tiene
Cuenta_Contable
Cuenta_Conta_Id
Cuenta_Conta_Numero
Cuenta_Conta_Dsc
Cuenta_Tipo_Id (FK)
Cuenta_Nat_Id (FK)
Cuenta_Conta_Habilitada
Divisa_Id (FK)
Cuenta_Conta_Divisa
Cuenta_Conta_Usuario
Cuenta_Conta_Sector
Cuenta_Conta_Registro
Tiene
Tiene Periodo
Divisa
Divisa_Id
Pais_Id (FK)
Divisa_Siglas
Divisa_Dsc
Divisa_Contravalor Divisa_Id (FK)
Divisa_Id_SIBI
Tiene
Pertenece
Plaza
Centro_Resp_Id (FK)
Plaza_Id
Divisa_Id (FK)
Plaza_Fecha
Plaza_SIBI
Afecta Cuenta En
Usa
Valoriza
Cuenta_Naturaleza
Cuenta_Nat_Id
Cuenta_Nat_Dsc
Se clasifica
Afectacion_Contable
Cuenta_Conta_Id (FK)
Divisa_Operativa_Id Divisa_Id (FK)
Afectacion_Contable_Id
Divisa_Id Divisa_Id (FK)
Afectacion_Contable_Even
Auxiliar_Id (FK)
Afecta_Corresan
Afecta_IVA
Afectacion_Tipo_Id (FK)
Centro_Resp_Id (FK)
Afecta
Genera
Genera
Cuenta_Tipo
Cuenta_Tipo_Id
Cuenta_Tipo_Dsc
Afectacion_Tipo
Afectacion_Tipo_Id
Afectacion_Tipo_Dsc
Operacion_Tipo
Operacion_Tipo_Id
Operacion_Tipo_Dsc
Evento
Evento_Id
Evento_Area_Id (FK)
Evento_Dsc
Define
Evento_Area
Evento_Area_Id
Evento_Area_Dsc
Movimiento_Generado
Movto_Generado_Id
Centro_Resp_Id (FK)
Cuenta_Conta_Id (FK)
Divisa_Operativa_Id (FK)
Afectacion_Contable_Id (FK)
Plaza_Id (FK)
Movto_Generado_Cuenta
Movto_Generado_sCuenta
Movto_Generado_$$Cuenta
Movto_Generado_$$$Cuenta
Movto_Generado_Concepto
Movto_Generado_Divisa
Movto_Generado_Usuario_Id
Movto_Generado_Naturaleza
Movto_Generado_Importe
Movto_Generado_Dsc
Movto_Generado_Cuenta_Dsc
Movto_Generado_Sector
Movto_Generado_Usuario_Trans
Movto_Generado_Fecha
Movto_Generado_Valido
Movto_Generado_DescError
Movto_Generado_Fase
Opera
Afectacion_Contable_Oper
Cuenta_Conta_Id (FK)
Divisa_Operativa_Id (FK)
Afectacion_Contable_Id (FK)
Operacion_Tipo_Id (FK)
Producto_Id (FK)
Fecha_Val_Tipo_Id (FK)
Entrega/Recibe
Entrega_Recibe
Afectacion_Contable_Abono
Afectacion_Contable_Estab
Afectacion_Contable_Venc
Afectacion_Contable_Usada
Afectacion_Contable_Evento
Cuenta_Conta_Id (FK)
Divisa_Operativa_Id (FK)
Afectacion_Contable_Id (FK)
Evento_Id (FK)
Afect_Cont_Evento_Abono
Afect_Cont_Evento_Estab
Afect_Cont_Evento_Venc
Afect_Cont_Evento_Usada
Afecta
Cierre_Contable
Cierre_Id
Cierre_Fecha
Cierre_SOI
Cierre_SIBI
Producto
Producto_Id
Producto_Id_SIBI
Producto_Dsc
Pactada
Fecha_Val_Tipo
Fecha_Val_Tipo_Id
Fecha_Val_Tipo_Dsc
Fecha_Val_Tipo_Val

ADMINISTRACION


Usuario
Usuario_Id int NOT NULL (FK)
Sus_Id smallint NULL (FK)
Usuario_Login varchar(25) NULL
Usuario_Password varchar(25) NULL
Usuario_Ape_Paterno char(30) NULL
Usuario_Ape_Mat. char(30) NULL
Usuario_Nombre char(30) NULL
Usuario_Fecha_Alta datetime NULL
Usuario_Fecha_Expiracion datetime NULL
Usuario_Linea_Rompeo money NULL
Usuario_Linea_Transfer money NULL
Usuario_Aviso_Cero money NULL
Identifica
Acceso
Permiso_Id smallint NOT NULL (FK)
Funcion_Id smallint NOT NULL (FK)
Modulo_Id smallint NOT NULL (FK)
Usuario_Id int NOT NULL (FK)
Se_Otorga
Asocia
Permiso_Id smallint NOT NULL (FK)
Funcion_Id smallint NOT NULL (FK)
Modulo_Id smallint NOT NULL (FK)
Reporte
Funcion_Id smallint NOT NULL (FK)
Modulo_Id smallint NOT NULL (FK)
Reporte_Nom. varchar(50) NOT NULL
Reporte_Doc varchar(50) NULL
sysusers
suid smallint NOT NULL
sid smallint NOT NULL
gid smallint NOT NULL
name varchar(30) NOT NULL
environ varchar(255) NULL
Hace_Uso
Modulo
Modulo_Id smallint NOT NULL
Modulo_Nom varchar(50) NOT NULL
Modulo_Doc varchar(120) NULL
Contiene
Consulta
Funcion_Id smallint NOT NULL (FK)
Modulo_Id smallint NOT NULL (FK)
Consulta_Nom. varchar(50) NOT NULL
Consulta_Doc varchar(50) NULL
Es_usado
Funcion
Funcion_Id smallint NOT NULL
Modulo_Id smallint NOT NULL (FK)
Subtipo_Id smallint NOT NULL (FK)
Funcion_Al smallint NULL
Modulo_Al smallint NULL
Permiso
Permiso_Id smallint NOT NULL
Permiso_Nom varchar(50) NOT NULL
Permiso_Doc varchar(120) NULL
Agrupa
Subtipo
Subtipo_Id smallint NOT NULL
Subtipo_Nom varchar(50) NOT NULL
Subtipo_Doc varchar(120) NULL
Comando
Funcion_Id smallint NOT NULL (FK)
Modulo_Id smallint NOT NULL (FK)
Comando_Nom varchar(50) NOT NULL
Comando_Doc varchar(50) NULL
InfoServer
InfoServer_Id smallint NOT NULL
Server_Name_SOI varchar(15) NULL
DB_Name_SOI varchar(15) NULL
Server_Name_SIBI varchar(15) NULL
DB_Name_SIBI varchar(15) NULL
Proceso
Funcion_Id smallint NOT NULL (FK)
Modulo_Id smallint NOT NULL (FK)
Proceso_Nom varchar(50) NOT NULL
Proceso_Doc varchar(120) NULL