UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

APLICACION DE LA TEORIA DE COMPUTACION DISTRIBUIDA (ARQUITECTURA CLIENTE/SERVIDOR) EN LA ELABORACION DE UN SISTEMA PARA EL CONTROL DE SEGUROS DE AUTOMOVILES.

TESIS

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

PRESENTAN:

ORTIZ ZARATE ANABEL
MIRANDA MIRANDA HORACIO
ROMERO MELENDEZ MAURICIO ALEJANDRO

DIRECTOR DE TESIS: ING. JOEL VILLAVICENCIO CISNEROS

MEXICO, D. F. 1995

Alguien dijo que escribir es un acto de masoquismo intelectual. Hay una gran parte de verdad en tal afirmación. En efecto, el autor sufre al escribir, pero también obtiene una inmensa satisfacción por ello, sobre todo cuando se trata de algo tan relevante en la vida personal. Lo mismo podría decirse de quienes vivieron conmigo el transcurso de éste escrito, en mi caso particular, su apoyo y su ánimo, han sido esenciales durante el tiempo que ha transcurrido este largo proyecto y aun antes.

Rigo y Jimmy a quienes debo el que mi carrera no haya sido tan solo mi preparación profesional sino la etapa en que encontré a mis verdaderos amigos.

A Salvador, por tantos años de lealtad y amistad incondicional permaneciendo en el escenario de mi vida.

Los Pérez-Almanza, al tío Paco y en especial a Ceci.

Este trabajo no hubiera sido posible sin el inmenso apoyo de Blanca y Chintegua en mi carrera y hasta la fecha.

Marce, para quien encamino todos mis esfuerzos y le debo todo lo mejor.

Por último, el principal apoyo a todos mis desvelos, tanto en el pasado como en la actualidad, ha sido mi familia: Mis padres a quienes debo absolutamente TODO.

Mauricio Alejandro Romero Meléndez.

Deseo expresar mi agradecimiento a las siguientes personas pues sin ellas nada de esto hubiera sido posible.

A mi madre por todo su sacrificio, comprensión, paciencia y amor. Ha contribuido significativamente a dar sentido a mi vida.

Mi abuelita Onésima Rosette por su apoyo y ejemplo a seguir.

A mis hermanos:
Juan Carlos, por todos aquellos momentos en que estuvo a mi lado a lo largo de la carrera apoyandome y estimulándome.
Marlene, por la confianza y cariño que nos une y a Job Edson por su lealtad, ternura y comprensión.

A mi esposo Horacio por ser la razón de mi existir, por compartir conmigo su amor, su vida y sus logros.

Toda mi gratitud a Bartolomé y Glafira Miranda por su apoyo y compañia en estos meses de intenso trabajo.

Anabel Ortiz Zárate

Con profundo agradecimiento a mis padres ***Bartolomé y Glafira Miranda****, por el empeño que han tenido en brindarle a sus hijos una formación llena de amor, deseos de superación y sentido de responsabilidad.*

A mis hermanos ***Leticia, Oscar y Daniel*** *que siempre han estado a mi lado con gran cariño y comprensión.*

A quien inyecta en mí la energía necesaria para enfrentar las dificultades de la vida, haciendo de ésta el mejor instrumento para crecer juntos, a mi querida esposa, ***Anabel.***

A mis familiares, amigos y todas aquellas personas que han estado conmigo contribuyendo de alguna forma en la culminación de este esfuerzo.

Horacio Miranda Miranda

CONTENIDO

CAPITULO I. FUNDAMENTOS TEORICOS.

I.1. Información, Datos y Archivos. 3
I.1.1. Obtención de información a partir de los datos. 3
I.1.2. Clasificación de los datos. 4
I.1.2.1. Tipos de datos según el objeto. 4
I.1.2.2. Tipos de datos según su naturaleza. 5
I.1.2.3. Tipos de datos según su dirección. 5
I.1.2.4. Características de los datos. 7
I.1.2.5. Recopilación y almacenamiento de la información. 8
I.1.2.5.1. Recopilación de datos. 8
I.1.2.5.2. Datos elementales y agrupaciones. 9
I.1.2.5.3. Almacenamiento de datos. 9
I.1.3. Tipos de Archivo según acceso. 10
I.1.3.1 Archivo de acceso secuencial. 10
I.1.3.2. Archivo de acceso directo. 11
I.2. Bases de Datos. 13
I.2.1. Introducción. 13
I.2.1.1. Necesidad y objetivos. 13
I.2.1.2. Abstracción de la información. 15
I.2.1.3. Modelos de datos. 16
I.2.1.4. Instancias, esquemas e independencia de los datos. 17
I.2.1.5. Lenguajes para la definición y el manejo de datos. 17
I.2.1.6. Manejador de base de datos. 18
I.2.1.7. Administrador y useaurios de la base de datos. 19
I.2.1.8. Estructura del sistema. 21
I.2.2. Modelo entidad-relación. 22
I.2.2.1. Entidades y conjuntos de entidades. 23
I.2.2.2. Relaciones y conjuntos de relaciones. 24
I.2.2.3. Limitantes de mapeo. 25
I.2.2.4. Llaves. 25
I.2.2.5. Diagramación. 26
I.2.2.6. Reducción de diagramas E-R a tablas. 27
I.2.3. Modelo relacional. 30
I.2.3.1. Estructura de las bases de datos relacionales. 31
I.2.3.2. Lenguajes de consulta formales. 31
I.2.3.3. Lenguajes de consulta comerciales. 32
I.2.3.3.1. SQL (Structured Query Language). 33
I.2.3.4. Vistas. 37
I.2.4. Diseño de bases de datos relacionales. 39
I.2.4.1. Introducción. 39
I.2.4.2. Normalización. 40
I.4.2.1. Primera forma normal. 41
I.4.2.2. Segunda forma normal. 42

I.4.2.3. Tercera forma normal. 43
I.2.5. Bases de datos distribuidas. 46
I.2.5.1. Definición y características. 46
I.2.5.2. Estructura de las bases de datos distribuidas. 47
I.2.5.3. Centralización, descentralización y distribución informática. 54
I.2.5.4. Nivel de descentralización y distribución. 56
I.2.5.5. Criterios a favor y en contra de la centralización. 57
I.2.5.6. Componentes del sistema distribuido. 63
I.2.5.7. Arquitectura del sistema distribuido. 65
I.3. Desarrollo de sistemas informáticos. 78
I.3.1. Revisión preliminar. 80
I.3.2. Análisis del sistema. 80
I.3.3. Estudio de viabilidad. 82
I.3.4. La evaluación y adquisición de hardware y software. 84
I.3.5. Desarrollo del sistema (programación). 89
I.3.6. Documentación. 92
I.3.7. Pruebas y puesta en servicio. 95
I.4. Arquitectura cliente/servidor. 96
I.4.1. Introducción a los sistemas distribuidos abiertos y al modelo cliente/servidor 98
I.4.2. ¿Qué es la arquitectura cliente/servidor? 103
I.4.3. Componentes de los sistemas cliente/servidor. 108
I.4.3.1. Interfaces gráficas de usuarios (GUI). 109
I.4.3.2. Sistemas operativos de redes (NOS). 113
I.5. Creación de la plataforma cliente/servidor. 115
I.5.1. El Cliente. 116
I.5.2. El Servidor. 118
I.6. Ventajas y desventajas de la arquitectura cliente/servidor. 121

CAPITULO II. IDENTIFICACION DE LA PROBLEMATICA.

II.1. Identificación de la problemática del usuario. 123
II.1.1. Identificación de los departamentos usuarios e individuos a contactar. 127

CAPITULO III. PLANEACION DEL PROYECTO

III.1. Planeación del proyecto. 131

CAPITULO IV. ANALISIS DEL SISTEMA.

IV.1. Objetivos del sistema. 135
IV.2. Determinación de la Información utilizada. 136
IV.2.1. Fuentes originadoras. 136
IV.2.2. Procesos. 136
IV.2.3. Productos generados. 137

IV.3. Funciones del sistema propuesto. 138
IV.4. Requerimientos de hardware y software. 139
IV.4.1. Configuración de red con grupo de trabajo. 139
IV.4.2. Configuración de red con servidor. 140

CAPITULO V. DISEÑO Y DESARROLLO DEL SISTEMA.

V.1. Consideraciones genarales. 145
V.2. Diagrama entidad-relación. 148
V.3. Diagrama de flujo de datos. 149
V.4. Diseño y desarrollo de los módulos. 151

CAPÍTULO VI. PRUEBAS E IMPLANTACION DEL SISTEMA.

VI.1. Pruebas del sistema. 187
VI.2. Implantación del sistema. 188

CONCLUSIONES 193

BIBLIOGRAFIA 197

INTRODUCCION

Ante el reto económico por el que estamos atravesando, una de las maneras para hacerle frente con mayor éxito es mantenerse informados en cuanto a la dirección que toma la tecnología.

Para lograr vencer la crisis económica debemos ser más competitivos, una de las formas como podemos lograrlo es aumentando la automatización de nuestros procesos. Con ese enfoque y actitud, se reducirán gastos y aumentará nuestra productividad.

Debido a la importante etapa en el desarrollo del sector financiero mexicano en cuanto al uso de tecnología, los grupos financieros que están en operación han acelerado sus estrategias con miras a modernizar sus procesos y ser realmente competitivos . En la recta final de 1994 entraron en operación varios de los bancos mexicanos nuevos, que buscarán competir en nichos de mercado muy específicos, en tanto llegan también los bancos del extranjero que han visto en el mercado mexicano un vasto potencial de negocios. La inversión en sistemas y equipos de cómputo ha sido esencial para estas empresas, porque saben que el procesamiento de la información financiera puede ser la clave de su competitividad en este medio.

Durante 1995 han llegado varios de los principales grupos financieros del exterior que han obtenido licencia para operar en México. Este sólo hecho promete hacer de 1995 un año todavía más dinámico y competitivo que 1994, donde en muchos casos el tipo de tecnología que se utilice puede hacer la diferencia en la rapidez con la que se alcancen a cubrir las metas de cada empresa. Esto vale tanto para las instituciones extranjeras como para las nacionales que, pese a los rezagos, han ido caminando hacia una superación tecnológica aceptable.

Sin embargo, todavía hasta en los "mejores" bancos mexicanos es frecuente encontrarse con que "se cayo el sistema" o "no hay línea", con lo cual se echan por tierra algunos adelantos que apuntan a brindar un mejor servicio al público.

A pesar de los avances logrados en el pasado, todavía el camino por recorrer es largo en materia de equipamiento tecnológico. La banca mexicana, particularmente, tendrá que

acelerar su modernización frente a los nuevos competidores, en aquellas áreas donde los que vienen no sólo son expertos en el negocio, sino también poseen una experiencia más amplia en la selección, aplicación y administración de la tecnología como herramienta estratégica de negocios.

Uno de los ámbitos que serán vitales en el desarrollo económico contemporáneo es la información, quizás en poco tiempo el parámetro que defina el grado de riqueza de un país no sean sus reservas petroleras, sus exportaciones o el nivel de desarrollo de su planta productiva sino la cantidad, calidad y el manejo que se haga de la *información.*

En el presente trabajo proponemos lo que estamos seguros es una solución óptima para el aprovechamiento de los recursos en un caso específico. En nuestra propuesta se incluye un análisis retrospectivo de los modelos utilizados para el desarrollo de sistemas de cómputo aprovechando de todos ellos las ventajas que presentan lo que nos llevó a seleccionar el modelo de computación distribuida conocido como *Cliente/Servidor.*

El modelo cliente *Cliente/Servidor* permite el aprovechamiento al máximo de la infraestructura de cómputo instalada, disminuyendo la necesidad de invertir en equipos grandes y costosos. Además de la flexibilidad en la interconexión y el poder de procesamiento de datos, esta tecnología nos permite utilizar el potencial de las *Interfaces Gráficas de Usuario (GUI's)* comerciales.

Está vez se logró desarrollar un sistema, no solo un programa, que no únicamente controla y administra las operaciones de los seguros de automóviles de un grupo financiero nacional sino que permite proporcionar un mejor servicio a los usuarios y clientes reflejándose en la competitividad de la institución.

CAPITULO I

FUNDAMENTOS TEORICOS

I.1 Información, Datos y Archivos

¿A qué llamamos dato y a qué información? La palabra *datos* equivale o no a la *información* según en qué contexto se esté hablando, se suelen denominar datos a la información obtenida a nivel primario y se considera que después, en los tratamientos posteriores, se va obteniendo información a niveles superiores. Así, la adquisición de tres pólizas de seguro es un dato y una información a nivel operativo, pero para el nivel directivo es un dato que debe ser tratado antes de obtener información; por ejemplo, sobre el número de pólizas de seguro vendidas durante este mes.

Entonces podemos llamar datos a todo aquel ente que nos produzca información.

I.1.1 Obtención de la información a partir de los datos

Iniciamos a continuación un estudio del concepto de dato y de la mayoría de sus posibles clasificaciones.

Si bien antes hemos hablado de la información como una colección de datos, se entiende en seguida que entre la recopilación de los mismos y la obtención de la información, media un triple proceso: el análisis, la selección y la depuración.

Con el fin de que esas tareas nos lleven a obtener información útil y eficaz, los datos deben reunir una serie de características fundamentales:

- Estar expresados, representados o transmitidos mediante símbolos que puedan ser eficazmente interpretados.

- Ser completos: si un dato ha de dar información por sí mismo sobre un aspecto concreto, debe ser completo; esto es, debe llenar el hueco informativo para el cual fue recopilado.
- Ser útiles: Ningún dato permanecerá como parte de la información si no es útil para responder a la pregunta cuya respuesta es necesario conocer.

I.1.2 Clasificación de los datos

I.1.2.1 Tipos de datos según el objeto

En ocasiones se establece una clasificación de los datos en dos grandes grupos generales, según sea la naturaleza del objetivo sobre el que se desea informar:

Datos sobre entidades

Este concepto incluye:

- Individuos aislados y conjuntos por entidad propia (clientes, apoderados, proveedores, empleados, etc.).
- Lugares o productos (equipos, materias primas, productos, etc.)

Datos relacionados con hechos

Cubren otro aspecto fundamental: Producir información a partir de hechos relacionados con una actividad concreta. Sea cual fuere el área que se contemple (finanzas, administración, legislación, etc.) la información sobre los hechos es imprescindible.

Se entiende por hecho o suceso todo lo que puede acontecer en un momento dado.

Un suceso es, por ejemplo, la apertura de una póliza de seguro para automóvil en una institución financiera.

I.1.2.2 Tipos de datos según su naturaleza

Los datos necesarios para emprender cualquier actividad pueden clasificarse también atendiendo a su naturaleza y, así, se puede hablar de:

- Datos numéricos: número de cuenta corriente, número de póliza de seguros, número de sucursal, etc.
- Datos alfabéticos: nombre de un cliente, nombre de una entidad bancaria, nombre de una calle, etc.
- Datos alfanuméricos (combinaciones de números y palabras): una dirección completa (Insurgentes 2020), 20,000.00 nuevos pesos , etc.

I.1.2.3 Por su dirección

Todavía puede hacerse otra reflexión bajo un nuevo punto de vista para enfatizar aún más la naturaleza de un dato.

La clasificación que estudiamos ahora permite ver de una forma más amplia el flujo de datos producido en un proceso completo de información. Hasta ahora no se ha establecido ninguna diferencia entre los datos según sea la dirección de su movimiento; es decir, el punto del que proceden y aquel al que se dirigen; tampoco se ha mencionado la cuestión evidente de que si se procede a recopilar una serie de datos, será con el fin de procesarlos de alguna manera.

Datos de entrada

Son los que se reciben del exterior. Si se elige el ejemplo de una entidad bancaria y la apertura de una póliza de seguro de automóvil, los datos de entrada necesarios para tal suceso serían, entre otros:

- Número del empleado
- Nombre del titular
- Tipo de cobertura solicitada

- Tipo del automóvil
- Modelo del automóvil

En este caso, la entidad bancaria actuaría de núcleo que solicita datos, y los puntos antes mencionados serían los datos de entrada hacia él.

De esta manera, el núcleo recoge e introduce hacia sí mismo los datos de entrada, actuando en esta primera etapa de receptor de frente al empleado suministrador de los mismos, cuyo papel es, el de emisor.

Concluido este flujo de información, el núcleo llevará a cabo diversos procesos en función de los datos recibidos. Frente al papel pasivo que antes ha jugado, pasa ahora a desarrollar su propia actividad. En el ejemplo mencionado se pondrían en funcionamiento todos los mecanismos propios de la entidad bancaria, orientados a crear e incluir esa nueva póliza de seguro de automóvil.

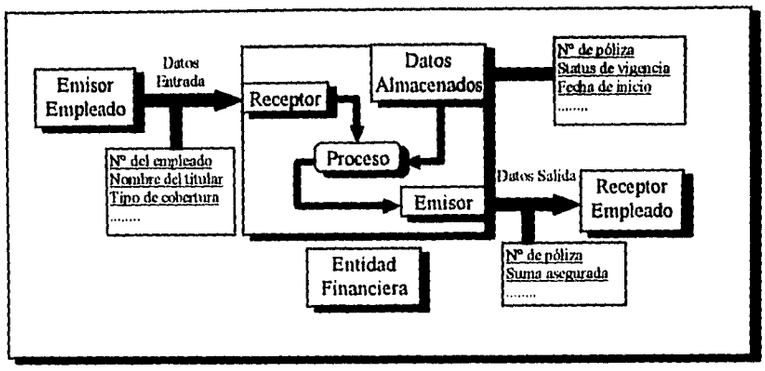
Datos de salida

Superada la etapa intermedia de procesado, el núcleo se convierte en emisor y da salida a determinados datos.

En el caso que nos ocupa, los datos de salida incluyen la confirmación de la creación de la póliza de seguro, el número de póliza que le corresponde, etc.

Puede observarse como ahora el proceso se invierte. Por lo cual, tenemos dos tipos de datos de *entrada* y de *salida.* Ambos tienen en común que pueden ser asociados con un movimiento en una dirección dada; entran o salen con respecto a un núcleo.

Lo anterior nos lleva a un concepto nuevo, los *datos almacenados,* éstos entrarán en juego cuando se deseara hacer uso del seguro por parte del empleado titular de la póliza, pues es necesario, primero, comprobar la existencia de la póliza y, segundo, cotejar que ésta esté vigente, si es así, registrar el siniestro ocurrido. Tal información no entra en ese instante porque ya existía, solo es consultada.



I.1.2.4 Características de los datos

He aquí otro punto importante para el manejo cómodo de la información. Ante una tarea determinada uno de los primeros pasos a seguir es concretar claramente y sin ambigüedad las características que describen los datos necesarios para realizar dicha labor.

Para la descripción de los datos se debe incluir información sobre:

- El *nombre* de los datos. Estará constituido por una serie de caracteres comprensibles y describirán el contenido del dato concreto. Hay que considerar que en la mayoría de las ocasiones no se utilizan para denominación de los datos su nombre completo. A veces se utilizan abreviaturas, iniciales o ciertos símbolos.

- *Valor* de los resultados. Es el contenido de un dato. Son pues, valores específicos, tanto cuantitativos como cualitativos.

- *Formato* de los datos. Este atributo hace referencia a la naturaleza de los caracteres que componen el valor de un dato. Así se puede hablar de formato *numérico* o *alfabético.*

- *Longitud* de los datos. Es decir, el número máximo de caracteres del valor de un dato.

1.1.2.5 Recopilación y almacenamiento de la información

Una vez concluida la revisión global de las características generales de los datos y su clasificación es el momento de abordar la forma de almacenarlos.

El registro de los datos (de entidades y de hechos) es un aspecto fundamental para su procesamiento adecuado.

1.1.2.5.1 Recopilación de los datos

Este punto afecta naturalmente a los datos de entrada: no se trata más que de la forma de conseguirlos.

Tal información suele recogerse y registrarse con base a formularios en los que se detallan estructuralmente los nombres de los datos de los que se desea disponer.

Una vez completados estos documentos se dispondrá de los datos de entrada requeridos.

El responsable de su posterior procesamiento deberá previamente:

- Especificar los datos de entrada necesarios y proporcionar su descripción.
- Diseñar el formulario: Este punto es primordial ya que debe conllevar a la recopilación rápida y evitar la existencia de errores.
- Determinar dónde deben recopilarse los datos.
- Especificar el *soporte* en el que se han de registrar (almacenar) los datos así como los dispositivos con los que se deben recopilar

I.1.2.5.2 Datos elementales y agrupaciones

En muchas ocasiones existe cierta información que a su vez se podría subdividir en elementos básicos relacionados entre sí por el hecho de englobarse dentro de la misma estructura informativa.

Muchos de los datos manejados por los sistemas de procesamientos tienen esta estructura jerárquica, que evidentemente se crea con el fin de que exista la posibilidad de manejar por separado datos elementales en aquellas operaciones en las que interese.

I.1.2.5.3 Almacenamiento de datos

Una vez recopilada la información por la vía elegida, el siguiente paso es conseguir un almacenamiento eficaz.

El almacenamiento por archivos es el procedimiento clásico.

Generalmente la unidad básica de almacenamiento es el carácter, que puede ser un número o una letra. El carácter ocupa el nivel inferior de almacenamiento y casi nunca se trabaja directamente con ellos aisladamente, sino con una agrupación que por sí misma posee una significación. Estos grupos de caracteres se denominan *campos*.

Los campos se agrupan para formar lo que se conoce como *registro*. Cada registro contiene la información completa de un ente en especial.

A su vez, el conjunto de todos los registros constituye el *archivo*.

La forma concreta que se establece para un registro dado recibe el nombre de *formato de registro*. Para su perfecta descripción se deben planificar:

- Nombres de los campos.
- Secuencia en la que se disponen los campos en el registro.
- Características que deben reunir los valores dados de los campos para que sean válidos.

- Longitud de cada campo. Se trata de especificar el número de caracteres que deben constituir el valor de cada campo.
- Naturaleza del valor de cada campo.
- Concretar si los caracteres antes mencionados serán números, letras, ambos u otros símbolos.

Lo más eficaz para la especificación de todas estas características es confeccionar patrones que se utilizarán siempre para todos los registros de estos sistemas.

De esta manera, se asegura que los datos se almacenen adecuadamente.

Para la identificación de cada uno de los registros es necesario definir el concepto de *llave*, esta es un campo del registro que lo identifica dentro de todo el archivo. Es necesario definir la llave que tendrá cada registro, que será única y servirá para identificar a éste. El concepto de llave existirá siempre sea cual fuere el tipo de archivo utilizado.

I.1.3 Tipos de archivo según acceso

Una vez estudiado cómo se organiza el almacenamiento de datos, el paso siguiente es elegir la forma de almacenamiento que determinará, a su vez, el sistema de acceso de archivo.

I.1.3.1 Archivos de acceso secuencial

El tipo de archivo más sencillo es aquel en que los registros están en un orden preestablecido o secuencial, como puede ser el nombre de empleado, número de pieza, etc. Muchas de las aplicaciones de cómputo están apoyadas en archivos secuenciales y, aunque actualmente se presentan otro tipo de archivos, los secuenciales seguirán utilizándose durante algún tiempo más.

Se dice que el acceso a un archivo es secuencial cuando la localización de un registro dado se hace iniciando la búsqueda desde el principio del archivo.

Comúnmente, los registros se disponen secuencialmente en orden por algún campo, que es el que se denominó campo llave. De esta manera, los registros se buscarán secuencialmente, atendiendo a la llave del registro.

El proceso requiere cierto tiempo, denominado *tiempo de acceso*. Pero si se desea que la localización sea instantánea, hay que elegir una forma distinta de almacenamiento que realice la búsqueda lo más brevemente posible. Además, este tipo de archivo no es práctico si se quiere mantener actualizado, suprimiendo o añadiendo registros.

I.1.3.2 Archivos de acceso directo

Los archivos de acceso directo proporcionan una gama más rápida de recuperación de la información en las aplicaciones en línea.

Habitualmente el software suministrado por el fabricante asigna a cada registro capturado una dirección de almacenamiento en particular. Por tanto se considera al archivo como un conjunto de registros numerados separadamente sin necesidad de preocuparse por la dirección física ya que el registro lógico está asignado a una dirección y su acceso se toma casi instantáneo.

Para relacionar una llave de interés con un registro lógico del archivo, se utiliza una tabla, pero el verdadero problema es encontrar la situación de un registro correspondiente a una llave en especial; es lo que se denomina *problema de transformación de llave a dirección*. Existen tres métodos básicos para la transformación de la llave en dirección:

Directo
Direccionamiento por tabla
Aleatorio

Acceso directo es el método más inmediato y de más escasa utilización. La llave coincide con la dirección. Así, si la llave utilizada es el campo de número de póliza, el registro almacenado con el número de póliza 120, está archivado en la dirección 120.

Evidentemente este sistema no siempre es posible utilizarlo, solo será posible usarlo sí:

- El campo llave es numérico
- Hay como máximo tantos campos llave como direcciones en el archivo.

Direccionamiento por tabla (o por asociación) es el sistema que se emplea con más frecuencia. Consiste en asociar a cada llave una dirección y formar una tabla con las llaves y las direcciones. Ésta puede estar en memoria, lo que optimiza la velocidad de búsqueda. Si la tabla es muy grande será preciso guardarla en disco, lo cual, implica que para accesarla se cargarán porciones de esta a memoria para su proceso.

Direccionamiento aleatorio es otra forma de recuperación de registros de forma directa utilizando una llave. En ocasiones se denominan acceso directo sin más adjetivos. Con este sistema se gana en velocidad a costa del espacio de almacenamiento. El sistema aleatorio consiste en realizar algún cálculo con la llave y utilizar el resultado de este cálculo como dirección. Es evidente que nada garantiza que el cálculo no proporcionará la misma dirección para dos llaves distintas. Este hecho se denomina *colisión* y a las llaves con la misma dirección se las llama *sinónimos*. En caso de colisión se puede calcular de nuevo la dirección o buscar el primer hueco que exista a continuación en el archivo y colocar ahí los datos. Para utilizar este sistema se necesita un archivo con lugares vacíos, o el proceso se tendrá que volver secuencial. La experiencia indica que se necesita un archivo que sea 50% mayor que el número total de registros para que se pueda utilizar este sistema.

I.2. Bases de Datos

I.2.1. Introducción

Los *Sistemas Manejadores de Bases de Datos* son un conjunto de archivos de datos independientes y relacionados entre sí (*Bases de Datos*) y una variedad de programas para accesar la información. El objetivo de estos sistemas es crear una interface transparente al usuario para guardar, modificar y recuperar la información de la base de datos en forma eficiente.

Durante la década de los 60's los Sistemas de Bases de Datos toman gran importancia teórica y comercial. En los siguientes años este campo de la Computación sufre grandes cambios conceptuales y tecnológicos. Los Sistemas de Bases de Datos se crean para el mantenimiento y explotación de un gran volumen de información. El manejo de los datos incluye tanto la definición de las estructuras para el almacenamiento de la información como los mecanismos para el manejo de la información. Para el análisis y comprensión de los Sistemas de Base de Datos se describen a continuación los conceptos fundamentales.

I.2.1. 1. Necesidad y Objetivos.

En un *sistema de procesamiento de archivos convencional,* los datos del sistema se guardan en diversos archivos permanentes, creados durante el tiempo con el fin de satisfacer distintas necesidades de la organización, y se escriben varios programas de aplicaciones para agregar y extraer información. Estos sistemas tienen ciertas desventajas importantes:

Redundancia e inconsistencia de los datos. Es posible que un mismo dato esté repetido en varios sitios (archivos). Esta redundancia aumenta los costos de almacenamiento y acceso, además de incrementar la posibilidad de que exista inconsistencia en la información, es decir, que las distintas copias de la misma información no concuerdan entre sí.

Dificultad para tener acceso a los datos. Este ambiente no permite recuperar la información requerida en forma conveniente o eficiente, dada su poca flexibilidad tanto en las estructuras utilizadas para el almacenamiento de la información como en los programas de aplicación específicos. Por lo tanto deben desarrollarse sistemas de recuperación de información de aplicación general.

Aislamiento de los datos. Dado que los datos están almacenados en varios archivos, y éstos pueden tener diferentes formatos, es difícil escribir nuevos programas para extraer los datos apropiados.

Usuarios múltiples. En un ambiente *multiusuario*, la interacción de las actualizaciones concurrentes puede resultar en información inconsistente, puesto que muchos programas de aplicaciones diferentes sin coordinación previa pueden tener acceso a los datos.

Problemas de seguridad. No es recomendable que todos los usuarios del sistema de base de datos puedan tener acceso a toda la información, puesto que los programas de aplicaciones se agregan al sistema en un forma precisa, es difícil implantar estas limitantes de seguridad.

Problemas de integridad. Los valores de los datos que se guardan en la base de datos deben de satisfacer ciertos tipos de *limitantes de consistencia*. El sistema debe obligar al cumplimiento de estas limitantes. Esto puede hacerse agregando el código apropiado a los distintos programas de aplicaciones. Sin embargo, cuando se añaden nuevas limitantes, es difícil cambiar los programas para asegurar su cumplimiento. El problema se complica aún más cuando las limitantes implican varios elementos de información de distintos archivos.

Estos y otros problemas han fomentado el desarrollo de los sistemas de manejo de bases de datos.

Al manejar gran cantidad de información los sistemas de base de datos involucran ciertos conceptos necesarios para su eficiencia.

La *integridad* en un sistema de base de datos implica considerar la unificación de varios archivos de datos independientes, es decir, se elimina la *redundancia* total o parcial entre los datos, cuando se requiere insertar un valor en la base de datos se deben satisfacer limitantes de *consistencia.*

Una consecuencia de la integridad en la base de datos es la *compartibilidad* de la misma, esta característica consiste en la capacidad de que varios usuarios puedan accesar al mismo subconjunto de la base de datos para ser utilizada con diferentes propósitos.

I.2.1.2. Abstracción de la información

Uno de los objetivos principales de un sistema de base de datos es proporcionar a los usuarios una visión *abstracta* de la información. Es decir, el sistema oculta ciertos detalles relativos a la forma como los datos se almacenan y mantienen. Sin embargo, para que el sistema sea útil la información debe recuperarse en forma eficiente. Existen tres niveles de abstracción de la información.

Nivel Físico.- Este nivel comprende la forma real de almacenamiento de los datos. En este nivel se describen en detalle las estructuras de datos complejas del nivel más bajo.

Nivel Conceptual.- En este segundo nivel se describe los datos reales almacenados y las relaciones que existen entre ellos. Este nivel contiene toda la base de datos en términos de unas cuantas estructuras relativamente sencillas. El nivel conceptual lo utilizan los administradores de la base de datos; quienes deciden que información se guarda en la base de datos.

Nivel de Visión.- Para este nivel, el sistema de bases de datos puede mostrar diferentes vistas para cada usuario.

I.2.1.3. Modelos de datos.

Los *modelos de datos* son un grupo de herramientas conceptuales para describir los datos, sus relaciones, su semántica y sus limitaciones. Los modelos de datos se dividen en tres grupos y a continuación se explican.

Los modelos lógicos basados en objetos. Se utilizan para describir los datos en los niveles conceptual y de visión. Su principal característica es la flexibilidad de la estructura de la base de datos y la clara especificación de las limitantes de los datos. Existen diversos modelos de este tipo, sin embargo, el más conocido es el modelo entidad-relación.

El modelo de datos entidad-relación es el más utilizado de esta clasificación. Se basa en la descripción del mundo real a través de la diferenciación de objetos básicos (entidades) y su interacción con otros objetos (relaciones). De esta forma la entidad es un ente que se distingue de los demás, conteniendo una serie de características propias (atributos).

Modelos lógico basados en registros. Describen los datos a nivel conceptual y de visión. Sin embargo a diferencia de los modelos basados en objetos, los modelos lógicos basados en registros especifican la estructura lógica general así como un nivel mas alto de la implantación. Los modelos mas comunes son:

Modelo Relacional. Los datos y las relaciones de describen a través de tablas, cada una de las cuales tiene varias columnas con nombres únicos.

Modelo de Red. Los datos se representan por medio de un conjunto de registros y las relaciones entre los datos por medio de apuntadores.

Modelo Jerárquico. El modelo jerárquico organiza los registros por medio de un conjunto de árboles, a pesar de que los datos y relaciones entre datos se describen a través de registros y apuntadores.

Modelos físicos de los datos. Estos modelos describen a los datos en el nivel más bajo de la abstracción de la información, Algunos de los más conocidos son: modelo

unificador y modelo de la memoria a cuadros; estos modelos frecuentemente son utilizados para la implantación de los sistemas de base de datos.

I.2.1.4. Instancias, esquemas e independencia de los datos

Es importante mencionar que el conjunto de información almacenada en la base de datos se le llamará *instancia*, y a la estructura o diseño general de la base de datos se denominará *esquema de la base de datos*. En la base de datos existen varios esquemas dependiendo del número de los niveles de abstracción de datos.

Otro concepto necesario es la *independencia de datos*, es decir, la posibilidad de modificar un tipo de esquema de la base de datos sin afectar el esquema del nivel inmediato superior. De esta forma la independencia de datos permite la capacidad de realizar cambios a la estructura de almacenamiento y de las estrategias de acceso sin afectar a las aplicaciones. Se distinguen dos tipos de independencia de datos y son :

Independencia Física. Si se modifica el esquema físico no se necesita rediseñar los programas de aplicación.

Independencia lógica. Cuando se modifica el esquema conceptual no se necesita volver a escribir los programas de aplicación, este último concepto es difícil de lograr, puesto que gran parte de los programas de aplicación dependen de la estructura lógica de los datos.

I.2.1.5. Lenguajes para la definición y el manejo de datos

Lenguaje de definición de datos. Es una serie de definiciones que especifican un esquema de base de datos. Por medio de este lenguaje se obtiene un conjunto de tablas que se almacena en un archivo especial llamado *diccionario de datos*, este diccionario de datos contiene *metadatos*, es decir, "datos acerca de los datos". La estructura de almacenamiento y los métodos de acceso se especifican por medio de un grupo de definiciones de un tipo especial de DDL llamado *lenguaje de almacenamiento y definición de los datos*. Cuando se

lleva a cabo la compilación se obtiene una serie de instrucciones que especifican los detalles de implantación de los esquemas de la base de datos.

Lenguaje de manejo de datos. Permite a los usuarios manejar o tener acceso a los datos que estén organizados por medio del modelo apropiado. Se tienen básicamente dos tipos de DML:

De procedimientos. El usuario especifica cuáles datos necesita y la forma en que se deben de obtener.

Sin procedimientos. El usuario especifica cuales datos quiere sin indicar la metodología para obtenerlos.

Una *consulta* es una proposición que solicita la recuperación de la información. Un *lenguaje de consultas* es la parte de un DML que implica la recuperación de la información.

I.2.1.6. Manejador de base de datos

La transferencia de información a través de la memoria principal y el disco constituye una gran problemática en los sistemas, puesto que dicha transferencia de datos del disco a memoria y de memoria a disco es lento comparado con la velocidad de la unidad de procesamiento central de la computadora, de tal manera que es indispensable que el sistema de bases de datos estructure la información para reducir el nivel de transacciones entre el disco y la memoria principal.

El objetivo de los manejadores de bases de datos es simplificar y facilitar el acceso a los datos. Las vistas de alto nivel ayudan a lograrlo. Si el tiempo de respuesta para una consulta es muy largo, el valor del sistema se reduce. El funcionamiento del sistema depende de la eficiencia de las estructuras de datos utilizados para la abstracción de los mismos y de que tan eficiente pueda operar el sistema con esas estructuras. Los manejadores de base de datos son un conjunto de programas que realizan la interfaz entre los datos del nivel bajo de almacenamiento y los programas de aplicación y las consultas hechas al sistema, son responsables de las siguientes actividades:

Interacción con el manejador de archivos. Los datos no procesados se almacenan en el disco mediante el sistema de archivos del sistema operativo convencional. El manejador de base de datos traduce las proposiciones en DML a comandos de sistema de archivos de bajo nivel. De esta forma el manejador de base de datos se encarga de almacenar, recuperar y actualizar los datos de la base de datos.

Implantación de la integridad. Deben existir límites de consistencia que permitan al sistema realizar filtros para los valores de los datos almacenados. El administrador de la base de datos debe establecer dichos criterios en forma explícita.

Puesta en práctica de la seguridad. Establece las restricciones y privilegios de los usuarios para el acceso a la información.

Respaldo y Recuperación. Es importante detectar las posibles fallas en el suministro de energía, así como los posibles errores que se lleguen a presentar, para restaurar la base de datos al momento de la falla.

Control de Concurrencia. Es posible que no se conserve la consistencia de los datos cuando más de un usuario realiza modificaciones al mismo dato, por lo tanto es necesario que el manejador de base de datos controle la interacción entre los usuarios concurrentes.

I.2.1.7. Administrador y usuarios de la base de datos

Administrador de base de datos. El administrador de la base de datos tiene la tarea de llevar un control centralizado de los datos y programas que tienen acceso a ellos, así como de hacer los ajustes necesarios a medida que los requerimientos cambian. Para llevar a acabo dicha actividad se deben realizar las siguientes funciones:

Definición de esquema. Realizar la creación del esquema original de la base de datos, escribiendo una serie de definiciones que el compilador DDL traduce en tablas que se almacenan en el *diccionario de datos.*

Definición de la estructura de almacenamiento y del método de acceso. Se escribe una serie de definiciones que serán traducidas por el compilador del lenguaje de almacenamiento y definición de datos.

Modificación del esquema y de la organización física. Estas modificaciones se realizan a través del compilador de DDL o por el compilador del lenguaje de almacenamiento y definición de datos.

Control de autorizaciones para el acceso a los datos. Establecer políticas de autorización y privilegios para que los distintos usuarios puedan accesar los datos del sistema.

Especificación de las limitantes de integridad. Verificar dichas limitantes para llevar a cabo las actualizaciones al sistema.

Usuarios de la base de datos. Existen tres tipos de usuarios de un sistema de base de datos y se distinguen por el modo en que interactuan en el sistema.

Programadores de Aplicación. Son profesionales en informática que interactuan con el sistema mediante llamadas en DML, incrustadas en algún programa escrito en un lenguaje *huésped* (Pascal, Cobol, C, etc.), a éstos se les llama *programas de aplicaciones*, y su objetivo es permitir al usuario manipular los datos que necesite.

Usuarios casuales. Son aquellos usuarios que realizan consultas a través de un lenguaje de consulta de la base de datos. Los requerimientos de estos usuarios se manejan a través de un *procesador de consultas*, tomando la proposición en DML y traduciéndola a instrucciones que pueda obtener el manejador de base de datos.

Usuarios ingenuos. Estos usuarios interactuan con el sistema a través de los programas de aplicación desarrollados para un fin determinado.

Usuarios especializados. Los requerimientos de estos usuarios no encajan en las necesidades tradicionales de procesamiento de datos. Escriben sus propias aplicaciones.

I.2.1.8. Estructura del sistema

Los sistemas de base de datos se divide en módulos para realizar diversas tareas, dejando al sistema operativo la realización de los procesos más elementales. De tal forma, se debe constituir una interfaz entre el sistema de base de datos y el sistema operativo. A continuación se presentan los diversos elementos de un sistema de base de datos.

El manejador de archivos. Realiza la asignación de espacio en disco y de las estructuras de datos que se van a emplear para representar la información almacenada en disco.

El manejador de base de datos. Se encarga de establecer la interfaz entre los datos de bajo nivel y los programas de aplicación y las consultas al sistema.

El procesador de consultas. Traduce las proposiciones de consulta a lenguaje de bajo nivel que pueda entender el manejador de la base de datos. Asimismo trata de encontrar la estrategia más eficiente para ejecutar una consulta.

EL precompilador de DML. Convierte las instrucciones en DML insertadas en un programa de aplicación en llamadas normales a procedimientos en el lenguaje huésped. Este precompilador interactua con el procesador de consultas para obtener el código apropiado.

El compilador de DDL. Convierte las proposiciones en DDL en un conjunto de tablas que contiene información de los datos (metadatos).

Archivos de datos. Este tipo de estructura de datos se requiere para la implantación del sistema físico, guarda la base de datos.

Diccionario de Datos. Almacena la información relativa a la estructura de la base de datos.

Índices. Perniten el acceso rápido a elementos de información que contienen valores determinados.

I.2.2. Modelo entidad-relación

El modelo de datos de entidad-relación es un modelo lógico de objetos que se basan en una percepción del mundo real transformándolo en un conjunto de objetos llamados *entidades* y de *relaciones* entre estos objetos.

Es frecuentemente usado cuando se mapea un sistema del mundo real a un *sistema manejador de base de datos relacional (RDBMS)*. El modelo de entidad-relación categoriza todos los elementos de un sistema, o bien como una *entidad* (una persona, un lugar o cosa) o como una *relación* entre las entidades.

La aplicación del modelo de datos de entidad-relación requiere los siguientes pasos. Primero, identificar las entidades en el sistema y construir una tabla para representar entidades. Segundo, identificar las relaciones entre las entidades. Tercero, identificar atributos de cada identidad y extender las tablas para incluir estos atributos.

Cuando se modela un sistema utilizando el modelo entidad-relación se debe de incluir un paso denominado *normalización*. Normalización es el proceso de asegurarse de que una tabla tiene una llave primaria completa y que las restantes columnas o llaves no primarias dependen por completo de la llave primaria.

La aplicación correcta del modelo entidad-relación resulta en un conjunto de tablas bien diseñadas. Los beneficios de un conjunto de tablas bien diseñadas incluyen:

- Almacenamiento reducido y eliminación de datos redundantes.
- Incrementa la integridad de la base de datos.
- Incrementa la habilidad de realizar cambios al sistema.
- Incrementa la productividad basada en la flexibilidad inherente a un sistema relacional bien diseñado.

I.2.2.1. Entidades y Conjuntos de entidades

Una *entidad* se define como un objeto que existe y puede distinguirse de otro objeto. Por ejemplo, el empleado "Mauricio Romero Meléndez", con número de empleado 9300010, es una entidad ya que identifica en forma única a un empleado específico de determinado grupo financiero. Del mismo modo, la empresa "Casa de Bolsa" con clave 05, es una entidad ya que identifica en forma única a una empresa de determinado grupo financiero.

Se le llama *conjunto de entidades* a un grupo de entidades del mismo tipo. Por ejemplo, el conjunto de todos los empleados de un grupo financiero constituyen el conjunto de entidades *empleados*.

Una entidad está representada por un conjunto de *atributos* (datos). Los posibles atributos del conjunto de entidades *empleados* son *número de empleado, apellido paterno, apellido materno* y *nombre*. El *dominio* del atributo es el rango de valores permitidos para éste, de este modo, el dominio del atributo *nombre* podría ser el conjunto de todas las cadenas de texto de 20 caracteres de longitud. De manera más formal, un atributo es una función que mapea un conjunto de entidades a un dominio. Así, cada entidad se describe por medio de un conjunto de parejas (atributo, valor del atributo), una pareja para cada atributo del conjunto de entidades.

9300010	Mauricio	Romero	Meléndez
9300011	Anabel	Ortiz	Zárate
9300012	Horacio	Miranda	Miranda
9300013	Hilda	González	Martínez
9400010	Jorge	Landa	León
9400011	Aarón	Pérez	López
9400012	Roberto	Arroyo	Garza

empleados

01	Banca
02	Asesoría
03	Seguros
04	Factoraje
05	Casa de Bolsa
06	Arrendadora
07	Instituto
08	Casa de Cambio

empresas

Conjuntos de entidades *empleados* y *empresas*

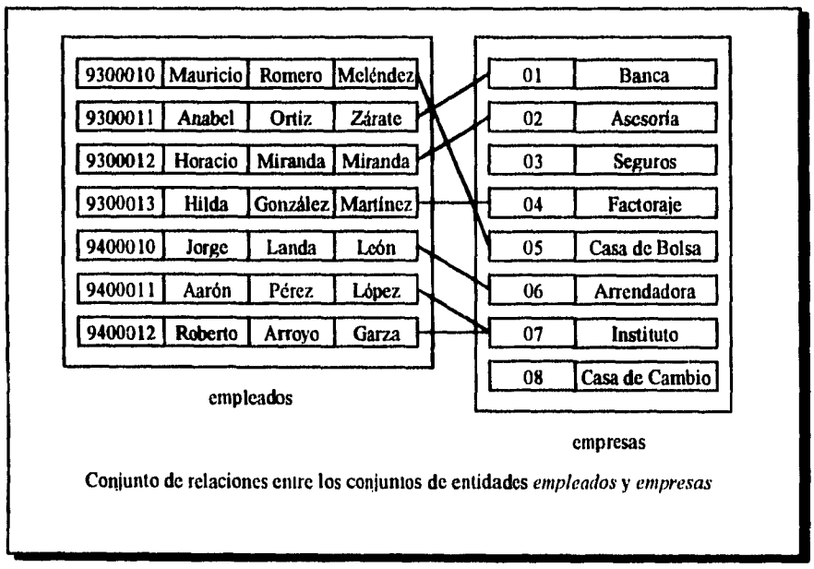
I.2.2.2. Relaciones y Conjunto de relaciones

Una *relación* se define como una asociación entre varias entidades. Es posible definir una relación que asocia al empleado "Mauricio Romero Meléndez" con la empresa 05; esto indica que el empleado "Mauricio Romero Meléndez" trabaja en la empresa 05.

Un *conjunto de relaciones* es un grupo de relaciones del mismo tipo.

Como ejemplo considérense los dos conjuntos de entidades *empleados y empresas*. Se define el conjunto de relaciones *pertenece a empresa* para denotar la asociación entre los empleados y las empresas del grupo financiero.

Una relación también puede tener atributos descriptivos. Por ejemplo, *fecha de ingreso* podría ser un atributo del conjunto de relaciones *pertenece a empresa*. Este atributo especifica la fecha en que el empleado ingresó a la empresa.



Conjunto de relaciones entre los conjuntos de entidades *empleados* y *empresas*

I.2.2.3. Limitantes de mapeo

Un esquema entidad-relación empresarial puede definir limitantes que deben cumplir los datos de la base de datos. Una limitante importante es la de las *cardinalidades de mapeo* que expresa el número de entidades con las que se puede asociar otra entidad mediante una relación.

Para un conjunto binario de relaciones, la cardinalidad de mapeo debe ser una de las siguientes: una a una, una a muchas, muchas a una ó muchas a muchas.

Dependencia por existencia. Se trata de otra clase importante de limitantes. Si la existencia de la entidad x depende de la existencia de la entidad y, entonces se dice que x es dependiente por existencia de y. Esto quiere decir que si se elimina y también se elimina x. Se dice que la entidad y es una entidad dominante y que x es una entidad subordinada.

I.2.2.4. Llaves

Surgen de la necesidad de distinguir las entidades y las relaciones al modelar una base de datos, la diferencia entre ellas debe expresarse en términos de sus atributos. Para hacer estas distinciones se asigna una *superllave* a cada conjunto de entidades. La superllave es un conjunto de uno o más atributos que, juntos, permite identificar en forma única a una entidad dentro de un conjunto de entidades.

El concepto de superllave no es suficiente para el modelado de la base de datos, ya que una superllave puede incluir atributos ajenos. Si K es una superllave, entonces también lo será cualquier superconjunto de K. Muchas veces lo que se busca es la superllave más pequeña posible. Es decir, se buscan superllaves para las cuales ningún subconjunto propio es una superllave. Estas superllaves mínimas se denominan *llaves candidato.*

Es posible que existan varios conjuntos de atributos distintos que pudieran servir como llaves candidato. Se utiliza el término *llave primaria* para referirse a la llave candidato que elija el diseñador de la base de datos como la forma principal de identificar a las entidades dentro de un conjunto de éstas.

Es posible que un conjunto de entidades no tenga suficientes atributos para formar una llave primaria. Una entidad de un conjunto de este tipo se denomina *entidad débil.* Una entidad que cuenta con una llave primaria recibe el nombre de *entidad fuerte.*

Un conjunto de entidades débiles no cuenta con una llave primaria, sin embargo es preciso tener una forma de distinguir entre todas esas entidades aquellas que dependen de una entidad fuerte determinada. El *discriminador* de un conjunto de entidades débiles es un conjunto de atributos que permite hacer esta distinción.

La llave primaria de un conjunto de entidades débiles está formada por la llave primaria de la entidad fuerte de la que depende su existencia y su discriminador.

Los conjuntos de relaciones también tienen llaves primarias. Sus llaves primarias se forman tomando todos los atributos que constituyen las llaves primarias de los conjuntos de entidades que definen al conjunto de relaciones.

I.2.2.5. Diagramación

Los componentes que integran un diagrama entidad relación son los siguientes:

Rectángulos. Representan conjuntos de entidades.
Elipses. Representan atributos.
Rombos. Representan conjuntos de relaciones.
Líneas. Conectan los atributos a los conjuntos de entidades, y los conjuntos de entidades a los conjuntos de relaciones.

Cada componente se etiqueta con su nombre correspondiente. Para ilustrar lo anterior, véase el siguiente ejemplo, supóngase que se tienen dos conjuntos de entidades: *empleado* y *empresa,* vinculados entre sí mediante un conjunto binario de relaciones *pertenece a empresa.* Los atributos asociados con *empleado* son *número de empleado, apellido paterno, apellido materno* y *nombre.* Los atributos relacionados con *empresa* son *número de empresa* y *nombre de empresa*

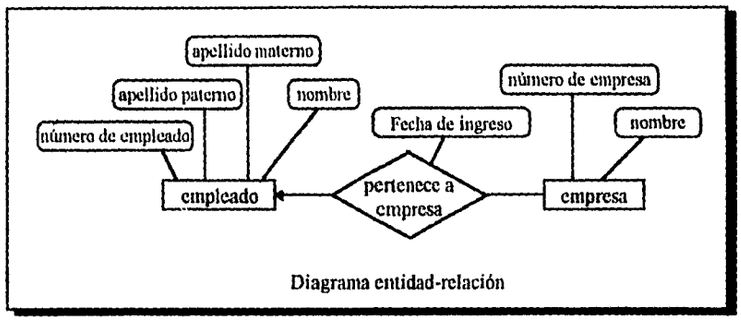


Diagrama entidad-relación

Para distinguir la cardinalidad de mapeo, se dibuja una línea con o sin dirección entre el conjunto de relaciones y el de entidades en cuestión. Una línea con dirección del conjunto de relaciones *pertenece a empresa* al conjunto de entidades *empleado* especifica que el conjunto de entidades *empleado* participa, ya sea en una relación una a una o muchas a una con el conjunto de entidades *empresa*. No puede participar ni en una relación muchas a muchas ni una a muchas con el conjunto de entidades *empresa*. Una línea sin dirección del conjunto de relaciones *pertenece a empresa* al conjunto de entidades *empleado* especifica que la entidad empleado participa, ya sea en una relación muchas a muchas o muchas a una con el conjunto de entidades *empresas*.

Por lo tanto, se puede ver que en el diagrama de la figura anterior el conjunto de relaciones *pertenece a empresa* es muchos a una.

Un conjunto de entidades débiles se indica en los diagramas entidad relación por medio de un rectángulo de doble pared. La relación que lo conecta al conjunto de entidades fuertes en el que se forma una llave primaria se señala mediante líneas gruesas.

I.2.2.6. Reducción de diagramas entidad-relación a tablas

Una *tabla* es una estructura de datos que almacena datos en una base de datos; está compuesta de renglones y columnas. Una tabla puede representar una entidad simple que se desea manejar dentro del sistema. Por ejemplo, puede representar una lista de empleados en alguna organización.

Una tabla puede representar también una relación entre dos entidades. Con esto, puede ser utilizada para representar la asociación entre empleados y la empresa donde trabaja. La mayoría de las tablas deben representar ya sea una entidad o una relación.

Representación de conjuntos de entidades. Sea E un conjunto de entidades con los atributos descriptivos $a_1,a_2,a_3,...,a_n$. Este conjunto de entidades se representa por medio de una tabla denominada E con n columnas diferentes, cada una de las cuales corresponde a uno de los atributos de E. Cada renglón de esta tabla corresponde a una entidad del conjunto de entidades de E.

Para ejemplificar lo anterior, considérese el conjunto de entidades *empresa* del diagrama entidad-relación de la figura anterior. Este conjunto de entidades tiene dos atributos *número de empresa* y *nombre de la empresa;* se representa por medio de la tabla llamada *empresa,* con dos columnas en la tabla como se muestra en la figura siguiente:

número de empresa	*nombre empresa*
01	Banca
02	Asesoría
03	Seguros
04	Factoraje
05	Casa de Bolsa
06	Arrendadora
07	Instituto
08	Casa de Cambio

La tabla *empresa*

Representación de conjuntos de relaciones. Sea R un conjunto de relaciones que implica los conjuntos de entidades $E_1,E_2,E_3,...,E_n$. Sea llave-primaria (E_i) el conjunto de atributos que constituyen a la llave primaria del conjunto de entidades E_i. Supóngase que R no tiene atributos descriptivos. Entonces la tabla que corresponde al conjunto de relaciones R tiene el siguiente conjunto de atributos.

$$\bigcup_{i=1}^{n} \text{llave primaria } (E_i)$$

En el caso de que R tenga atributos descriptivos, los cuales serían $\{a_1,a_2,a_3,\ldots,a_m\}$, entonces la tabla correspondiente a R tendrá el siguiente conjunto de atributos

$$\bigcup_{i=1}^{n} \text{llave primaria } (E_i) \cup \{a_1,a_2,a_3,\ldots,a_m\}$$

Para ejemplificar lo anterior, considérese el conjunto de relaciones *pertenece a empresa* del diagrama entidad-relación de la figura del punto I.2.2.5. Este conjunto de relaciones implica a los dos conjuntos de entidades: *empleado,* cuya llave primaria es número de empleado y *empresa,* cuya llave primaria es *número de empresa.*

Puesto que el conjunto de relaciones tiene un atributo descriptivo, *fecha de ingreso,* la tabla *pertenece a empresa* tiene tres columnas tituladas *número de empleado, número de empresa y fecha de ingreso,* como se muestra en la figura siguiente:

número de empleado	*número de empresa*	*fecha de ingreso*
9300010	05	17 junio 1993
9300011	01	25 junio 1993
9300012	02	13 agosto 1993
9300013	04	9 enero 1993
9400010	06	12 febrero 1994
9400011	07	21 marzo 1994
9400012	07	10 mayo 1994
9400013	08	5 abril 1994

La tabla *pertenece a empresa*

I.2.3. Modelo relacional

El modelo relacional ha revolucionado el mundo de las bases de datos. Fue desarrollado en 1970 por el Dr. E. F. Codd de los laboratorios IBM de San José. Los primeros sistemas de base de datos estaban basados en el modelo jerárquico o en el de red. Estos están más ligados a la implantación física de la base de datos que el modelo relacional.

El modelo de datos relacional representa la base de datos como un conjunto de tablas. Aunque las tablas son un concepto simple e intuitivo, existe una correspondencia directa entre el concepto de una tabla y el concepto matemático de una relación. La principal ventaja del modelo relacional es que este no necesita de mezclar apuntadores y datos en tablas. En cambio los registros son ligados por relaciones entre los atributos; una relación consiste de una liga entre registros de dos tablas que tienen idénticos atributos.

El nombre formal para una tabla es de una *relación* (de ahí el nombre de *manejador de bases de datos relacionales*) sin embargo el uso de "tabla" es más común. Las ventajas de usar tablas para modelar datos son:

- El formato de renglones/columnas de una tabla es una forma familiar de visualizar datos.
- Los apuntadores físicos no se necesitan para representar relaciones. Los mismos datos son utilizados para representar relaciones entre ellos. Consecuentemente este tipo de relaciones entre datos es más fácil de entender.
- Las tablas y las operaciones sobre tablas son bien definidas de acuerdo a que la teoría relacional esta basada en la teoría de conjuntos, álgebra relacional y cálculo relacional.

El orden interno de las columnas y renglones dentro de una tabla no es trascendente. De hecho, no es necesario nunca saber el orden físico de las columnas y renglones. La preocupación debe de concentrarse en como se representarán las columnas y renglones dentro de un reporte o en una búsqueda.

El número de columnas en una tabla típicamente permanece siempre constante. Las columnas de una tabla permanecen constantes debido a que la entidad o relación que se desean modelar han sido bien definidas y se ha decidido qué atributos se desean monitorear.

Los renglones almacenan los datos de una tabla. Cada columna representa una ocurrencia de la entidad o de la relación dentro de la tabla. En la tabla de empleados, cada renglón representa a un empleado.

Mientras que el número y los nombres de las columnas en una tabla permanecen relativamente estáticos con el tiempo, el número de renglones en una tabla es un tanto dinámico. Por ejemplo, la tabla de *empleado* representa personas que trabajan en una empresa, entonces cada renglón dentro de la tabla de *empleados* representa a un empleado de la empresa. Los renglones de las tabla de *empleados* se reducirán y se incrementaran al mismo ritmo que se despiden y contratan empleados.

Un *valor* es el dato referenciado por la intersección de un renglón y una columna dados. En valor asume el tipo de dato de su columna y puede ser nulo.

I.2.3.1. Estructura de las bases de datos relacionales

Una base de datos relacional consiste en un conjunto de tablas, que tienen asignado un nombre único. Una columna de una tabla representa una relación entre un conjunto de valores. Puesto que una tabla es un conjunto de estas relaciones, existe una correspondencia entre el concepto de tabla y el concepto matemático de relación. del cual recibe su nombre el modelo de datos relacional.

I.2.3.2. Lenguajes de consulta formales

Álgebra relacional. El álgebra relacional es un lenguaje de consulta de procedimientos. Existen cinco operaciones fundamentales en el álgebra relacional que son: *elegir, proyectar, producto-cartesiano, unión* y *diferencia* de conjuntos. Todas ellas producen como resultado una nueva relación.

Los cinco operadores permiten hacer una definición completa de una expresión en el álgebra relacional . Una expresión básica en el álgebra relacional consta de cualquiera de las siguientes:

- Una relación en la base de datos.
- Una relación constante.

Cálculo relacional. El álgebra relacional es un lenguaje de procedimientos, porque cuando se escribe una expresión en álgebra relacional, se proporciona una secuencia de operaciones que genera la respuesta a la consulta. Por otro lado, el cálculo relacional es un lenguaje sin procedimientos, donde se da una descripción formal de la información deseada sin especificar como obtenerla.

Existen dos formas de cálculo relacional: una en la que las variables representan tuplas, y otra en la que las variables representan valores de dominios . Estas variantes se denominan cálculo relacional de tuplas y cálculo relacional de dominios

I.2.3.3. Lenguajes de consulta comerciales

Los lenguajes formales que se mencionaron anteriormente permiten representar las consulta en forma concisa. Sin embargo los sistemas comerciales de base de datos requieren un lenguaje de consulta más "amable con el usuario". Como lenguajes comerciales podemos mencionar: SQL, Quel, QBE, los cuales representan diversos estilos: QBE está basado en el cálculo relacional de dominios, Quel se basa en el cálculo relacional de tuplas, y SQL utiliza una combinación de álgebra relacional y construcciones de cálculo relacional. Los tres lenguajes son importantes no sólo en los sistemas de bases de datos para la investigación sino también en sistemas distribuidos comercialmente.

Aunque se les da el nombre de "lenguajes de consulta", esto no es del todo correcto. SQL, Quel, QBE realizan otras funciones además de consultar bases de datos. Entre éstas pueden mencionarse la capacidad de definir la estructura de los datos, para modificar los de la base, y para especificar las limitantes de seguridad

I.2.3.3.1. SQL *(Structured Query Language).*

En este trabajo profundizaremos sobre SQL por ser el lenguaje de nuestro interés.

El lenguaje SQL fue desarrollado de un prototipo de manejador de base de datos relacional, el Sistema R, por *IBM* a mediados de los 70's. El Sistema R fue descrito por E. F. Codd en la edición de Noviembre del 1976 del *Journal of Research & Development* de *IBM*. En 1979, *Oracle Corporation* introdujo la primera implementación comercial disponible de SQL. En la actualidad SQL ha sido incluido como lenguaje de consulta en la mayoría de sistemas manejadores de base de datos.

La estandarización del SQL. El Instituto Nacional Americano de Estándares (ANSI) adopto a SQL como el lenguaje estándar para sistemas de manejadores de base de datos relacionales en Octubre de 1986 (ANSI X3.135-1986). El estándar de SQL ha sido también adoptado por la Organización Nacional de Estándares (ISO Standard 9075), así como el Gobierno Federal de los Estados Unidos, Estándar de Procesamiento de Información Federal (FIPS) número 127. Recientemente, la Característica Mejorada de Integración (*Integrity Enhanncement Feature* descrito por ANSI SQL Addendum I ha sido combinada con el SQL original y ha sido recatalogado como X3.135-1989 y 9075-1989. Todos los sistemas manejadores de bases de datos relacionales grandes soportan alguna forma de SQL y la mayoría trata de concordar con el estándar ANSI SQL89.

SQL continua evolucionando como un lenguaje y varios aspectos de este lenguaje no han sido definidos. Por ejemplo, la definición completa de la integridad referencial no ha sido todavía finalizada por ninguna de las organizaciones de estándares antes descritas.

Beneficios de SQL. Los siguientes párrafos ilustran algunas de las razones por las que el SQL ha tenido una amplia aceptación por los vendedores de bases de datos relacionales y por los usuarios finales. Sus beneficios abarcan rangos de usuarios incluyendo programadores de aplicaciones, administradores de bases de datos, gerencia y usuarios finales.

Lenguaje no procedural. SQL no es un lenguaje procedural porque:

- procesa conjuntos de registros en vez de uno a la vez.
- provee navegación automática de los datos.

SQL permite trabajar con estructuras de datos de alto nivel. En vez de manipular registros unitarios, se manejan conjuntos de registros. La más común forma de conjuntos de registros es una tabla. Todas las instrucciones de SQL aceptan conjuntos como entrada y todas las instrucciones de SQL regresar conjuntos a la salida. La propiedad de los conjuntos de SQL permite que el conjunto de salida de una instrucción sea tomado como entrada para otra instrucción.

SQL no requiere que se especifique el método de acceso hacia los datos. Esta cualidad hace más fácil el concentrarse en obtener los resultados requeridos. Todas las instrucciones de SQL utilizan el optimizador de *query*, una porción del *RDBMS*, el cual determina la forma más rápido de acceder a los datos seleccionados. El optimizador de *query* conoce que ligas existen y utiliza estas donde es necesario; cuando accesa a una tabla, no se necesita nunca saber si una tabla tiene ligas o que tipo de estas tiene la tabla.

Un lenguaje para todos los usuarios. SQL es usado para todo tipo de actividades de bases de datos por todos los rangos de usuarios incluyendo:

- administradores de sistemas.
- administradores de bases de datos.
- programadores de aplicaciones.
- personal administrativo.
- personal de soporte de decisiones en sistemas.
- muchos otros tipos de usuarios finales.

SQL provee una variedad de comandos de fácil aprendizaje que son tanto consistente como aplicables para todos los usuarios. Las instrucciones básicas de SQL pueden ser aprendidas en unas cuantas horas y las más avanzadas pueden ser dominadas en unos cuantos días.

Lenguaje Unificado. SQL provee instrucciones para una gran variedad de tareas incluyendo

- búsqueda de datos.
- inserción, actualización, y borrado de líneas en una tabla.
- creación, modificación, y borrado de objetos de base de datos.
- control de acceso a la base de datos y a sus objetos.
- garantías de la consistencia de la base de datos.

Sistemas previos de manejadores de bases de datos ofrecen un lenguaje separado para cada una de las categorías descritas arriba. SQL unifica todas estas tareas en un lenguaje consistente.

Lenguaje común para todas las bases de datos relacionales. Debido a que los sistemas manejadores de bases de datos relacionales mas grandes soportan SQL, se pueden transferir todas las habilidades ganadas con SQL de un RDBMS a otro. Además, todos los programas que se han escrito en SQL son transportables estos pueden ser movidos desde un RDBMS a otro con muy pocas modificaciones.

Comandos principales del SQL. Enseguida se presenta la descripción de los principales comandos del SQL:

SELECT

Propósito: Despliega renglones y columnas de una o más tablas. Puede ser usado como una instrucción, o (con ciertas restricciones en cláusulas) como una búsqueda o subbúsqueda en otra instrucción.

Sintaxis: SELECT [ALL|DISTINCT] { * | table. * | expr [c_alias] }
[, {table.*} | expr [c_alias] }]
FROM [user.] table [t_alias] [, [user.] table [t_alias]]
[WHERE condition]
[CONNECT BY condition [START WITH condition]]
[GROUP BY expr [, expr] [HAVING condition]]
[{UNION | INTERSECT | MINUS} SELECT]
[ORDER BY {expr | position} [ASC | DESC]
[, {expr | position} [ASC | DESC]]]
[FOR UPDATE OF column [, column] [NOWAIT]]

INSERT

Propósito: Adiciona renglones a una tabla específica o a una tabla referenciada en una vista.

Sintaxis: INSERT INTO [user.] table [(column [, column] ...)]
{ VALUES (value [, value] . . .) | query }

UPDATE

Propósito: Cambia los datos en una tabla específica

Sintaxis: UPDATE [user.] table [alias]
SET column = expr [, column = expr] . . .
[WHERE condition]

DELETE

Propósito: Remueve renglones de una tabla o de una vista.

Sintaxis: DELETE [FROM] [user.] table [alias] [WHERE condition]

ALTER TABLE

Propósito: Altera la definición de la tabla de acuerdo a:

- adicionar columnas o constantes.
- modificar definiciones de columnas.
- modificar asignación de almacenamiento futuro.
- recordar que un BACKUP ha tomado efecto sobre la tabla.

Sintaxis: ALTER TABLE [user.] table
[ADD ({ column_element | table_constriant}
[, { column_element | table_constraint}] . . .)]
[MODIFY (column_element [, column_element] . . .)]
[DROP CONSTRAINT constraint] . . .
[PCTFREE integer] [PCTUSED integer]
[INITRANS integer] [MAXTRANS integer]
[STORAGE storage]
[BACKUP]

Ejemplo: Para adicionar una columna llamada THRIFTPLAN a una tabla llamada EMP, con un máximo de siete dígitos y dos posiciones decimales, así como una columna de un caracter llamada LOANCODE, se tiene:

```
ALTER TABLE EMP
  ADD (THRIFTPLAN NUMBER(7,2), LOANCODE CHAR(1))
```

Para incrementar el tamaño de THRIFTPLAN a nueve dígitos se tiene:

```
ALTER TABLE EMP
  MODIFY (THRIFTPLAN NUMBER(9,2) )
```

CREATE TABLE

Propósito: Crear una tabla, la estructura básica para almacenar datos de usuario, con opciones para:

- determinar el espacio de una base de datos.
- especificar el tamaño de la tabla.
- crear clusters en la tabla.
- cargar la tabla con resultados de búsquedas arbitrarias.

Sintaxis:

```
CREATE TABLA [user.] table
    ( { column_element | table_constraint}
    [, {column_element | table_constraint} ] . . . )
    [PCTFREE n] [PCTUSED n]
    [INITRANS n] [MAXTRANS n]
    [TABLESPACE tablespace]
    [STORAGE storage ]
    [CLUSTER cluster (column [, column] . . .]
    [AS QUERY}
```

I.2.3.4. Vistas

Una *vista* es una representación lógica de otra tabla o combinación de tablas. Una vista deriva sus datos desde las tablas en las que se basa. Estas tablas son llamadas *tablas base*. Las tablas base pueden cambiar a ser tablas actuales o bien vistas.

Se pueden usar vistas en la mayoría de los casos posibles de las tablas. Las vistas pueden ser buscadas, actualizadas, insertadas en, y borradas de, justo como las tablas estándares. Las vista actualmente no contienen datos; sin embargo, estas derivan sus datos de las tablas base. Todas las operaciones realizadas en una vista actualmente afectan la tabla base de la vista.

Las vistas representan medios para presentar una diferente representación (como subconjuntos o superconjuntos) de los datos que residen en otras tablas o vistas. Las vistas son muy poderosas porque permiten el manejar diferentes presentaciones de los datos a diferentes usuarios.

Las vistas son usadas para los siguientes propósitos:

1. Para proveer un nivel adicional de seguridad de las tablas al restringir acceso a predeterminados conjuntos de renglones y/o columnas o una tabla. Por ejemplo, una vista sencilla puede ser creada la cual representará datos sobre ventas del este al gerente regional del este, y los datos de las ventas del oeste se muestran al gerente de ventas del oeste, y todo el conjunto de datos de ventas son representados al gerente de ventas nacional de la misma tabla de datos.

2. Para ocultar complejidad de los datos. Por ejemplo, una sola vista puede ser usado para regresar las columnas de múltiples tablas, por lo tanto los usuarios solo tienen que recordar el nombre de la vista.

3. Para reducir complejidad sintáctica. Por ejemplo, las vistas permiten a los usuarios seleccionar información de múltiples tablas sin saber como realizar la operación de juntura.

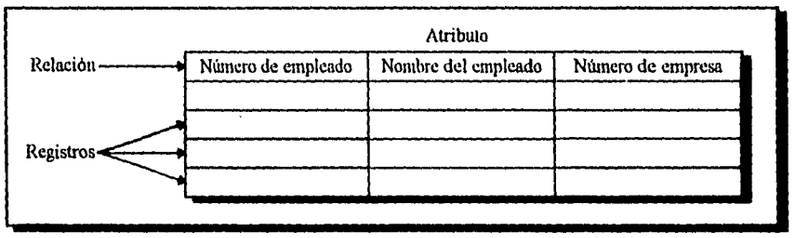
4. Para presentar los datos desde otra perspectiva. Por ejemplo, las vista proveen medios para renombrar columnas sin afectar las tablas en las cuales la vista se basa.

5. Para proporcionar un nivel de integridad referencial.

I.2.4. Diseño de bases de datos relacionales

I.2.4.1. Introducción

El *modelo relacional* es en la actualidad el más popular en los sistemas de manejo de bases de datos, puesto que es conceptualmente sencillo y comprensible por los profesionistas de los sistemas de información y muchos otros usuarios finales; puede evolucionar, ya que las relaciones entre los datos no necesitan estar predefinidas, además utiliza valores de éstos para implicar las relaciones. El modelo relacional, desarrollado en 1970 por E. F. Codd, se basa en una *relación:* una tabla bidimensional. Los renglones de la tabla representan los registros y las columnas muestran los atributos de una entidad. Las *bases de datos relacionales* utilizan un modelo para mostrar cómo se relacionan lógicamente los datos de un registro.



El orden de los datos en la tabla no es significativo y tampoco implica un orden cuando los registros están incluidos en la relación. Análogamente, los detalles físicos de almacenamiento (ya sea una organización aleatoria, indexada o secuencial) no son de interés para el analista. Las tablas relacionales muestran las relaciones lógicas, no físicas.

Al hacer una solicitud de información, el sistema produce una tabla que contenga la información.

Lo atractivo de las bases de datos relacionales es que el modelo relacional se puede comprender rápidamente. El objetivo del diseño de una base de datos relacional es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información. Una

de las técnicas para lograrlo consiste en diseñar esquemas que tengan una *forma normal* adecuada. Para determinar si un esquema de relaciones tiene una de las formas normales se requiere mayor información sobre la empresa del "mundo real" que se intenta modelar con la base de datos. La información adicional la proporciona una serie de limitantes que se denominan dependencias de datos.

Los defectos más comunes que puede tener una base de datos mal diseñada son:

- Repetición de la información
- Incapacidad para representar cierta información
- Pérdida de información

I.2.4.2. Normalización

Al planear la organización de los datos que se van a almacenar, el analista debe preveer la necesidad de accesar los datos para cumplir con requerimientos inesperados, objetivo que se puede alcanzar mediante la normalización de los datos.

El proceso de transformar datos existentes dentro de una forma relacional es llamado *normalización*. La normalización de datos se basa en la suposición de que se tienen organizados los datos dentro de una estructura tabular en donde una tabla contiene solo entidades del mismo tipo.

La *normalización* es el proceso de simplificar la relación entre los campos de un registro. Por medio de la normalización, un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. La normalización se lleva acabo por las siguientes razones:

- Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
- Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consulta y reportes.

- Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
- Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.
- Eliminar duplicado de información en cada tabla.
- Minimizar los impactos de cambios a la estructura de la base de datos, a las aplicaciones front-end (de usuario final) que procesan los datos.

Se han realizado investigaciones extensas con el fin de desarrollar métodos para la normalización.

Los analistas de sistemas deben familiarizarse con los pasos de la normalización, ya que este proceso puede mejorar la calidad del diseño de una aplicación.

1. Descomponer todos los grupos de datos en registros bidimensionales.
2. Eliminar todas las relaciones en las que los datos no dependen completamente de la llave primaria del registro.
3. Eliminar todas las relaciones que contengan dependencias transitivas.

1.2.4.2.1. Primera Forma Normal

Una de las mejoras básicas que el analista puede hacer es diseñar la estructura de un registro de manera que todos los registros de un archivo tengan la misma longitud. Los registros de longitud variable crean problemas especiales, ya que el sistema debe verificar siempre en dónde se encuentran los extremos de un registro (por ejemplo, buscando marcas especiales o leyendo un indicador de la longitud). Al fijar la longitud del registro se elimina este problema.

La primera forma normal se alcanza cuando se quitan todos los grupos de repetición, de modo que un registro tenga longitud fija. Un grupo de repetición, es decir, la aparición repetida de un dato o grupo de datos dentro de un registro, es en realidad otra relación. Por

lo tanto, se quita del registro y se le considera como una parte del mismo o como una relación adicional.

La primera forma normal se alcanza cuando un registro se diseña de longitud fija. Esto se lleva a cabo quitando el grupo de repetición y creando un archivo o relación aparte que contenga el grupo de repetición. El registro original y el nuevo se interrelacionan mediante un punto común de los datos.

1.2.4.2.2. Segunda Forma Normal:

La segunda forma normal se alcanza cuando un registro está en la primera forma normal y cada campo depende totalmente de la llave del registro (en el almacenamiento y recuperación). En otras palabras el analista busca la *dependencia funcional*: un campo es funcionalmante dependiente si su valor está asociado de manera única con un campo específico. Aunque este concepto suena complejo, en realidad es muy sencillo, como lo mostrará este ejemplo.

Los departamentos estatales de control vehicular trabajan arduamente para garantizar que a un vehículo en el estado se le asigne un número de placas específico. el número de placa identifica de manera única a un vehículo específico; un número de serie de un vehículo está asociado con uno y sólo un número de placa. Así se conoce el número de serie de un vehículo, se puede determinar el número de placa. Esto es la dependencia funcional.

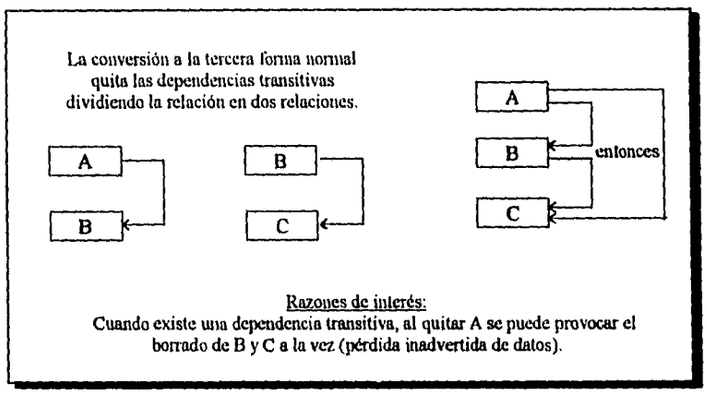
En contraste, si el registro de un vehículo contiene los nombres de todas las personas que lo manejan, se pierde la dependencia funcional (y el diseño del archivo no cumple el objetivo de la segunda forma normal) ¿Por qué? Si conocemos el número de licencia, no conocemos quién es el conductor (pueden ser muchos). Y si conocemos el nombre de un conductor, no conocemos el número de placa o de serie del vehículo específico ya que un conductor puede estar asociado con más de un vehículo en el archivo.

Para alcanzar la segunda forma normal, cada campo del registro que no dependa de la llave primaria del registro debe quitarse y utilizarse para formar una relación aparte.

1.2.4.2.3. Tercera Forma Normal

La tercera forma normal se alcanza cuando se quitan las dependencias transitivas de un diseño de registro. El caso general se muestra a continuación (Ver fig. siguiente.):

- A, B, y C son tres datos en un registro.
- Si C es funcionalmente dependiente de B y
- B es funcionalmente dependiente de A,
- entonces C es funcionalmente dependiente de A.
- Por lo tanto, existe una dependencia transitiva.



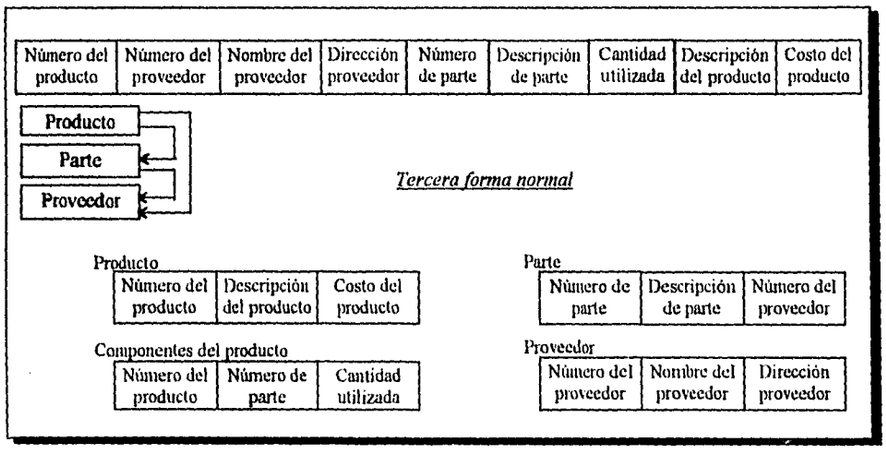
En el manejo de datos, la dependencia transitiva es una preocupación, ya que los datos pueden perderse de manera inadvertida cuando la relación está oculta. En el caso anterior, si se quita A, entonces también se quitan B y C, sea o no ésta la intención. Este problema se elimina diseñando el registro para la tercera forma normal. La conversión a la tercera forma normal quita la dependencia transitiva dividiendo la relación en dos relaciones separadas.

Consideremos otro ejemplo. Al utilizar la información sobre el proveedor de la figura siguiente, podemos ver qué PARTE depende de PRODUCTO, y qué PROVEEDOR depende de PARTE . Si quitamos un cierto producto de la base de datos, no sería adecuado eliminar las partes y los proveedores (los cuales podrían estar asociados con otros

productos). O bien, podríamos desear un registro de cada uno en la base de datos para un uso futuro.

Para quitar la dependencia transitiva en esta situación, se crean los registros PRODUCTO PARTE y PROVEEDOR (Fig. siguiente). Los registros adicionales dan mayor flexibilidad para cubrir los requerimientos futuros a la vez que eliminan las dificultades mencionadas arriba.

De lo anterior, se puede ver que la comprensión de cómo eliminar las dependencias transitivas requiere de un buen conocimiento de la relación entre los datos y las actividades empresariales en las que se utilizan. Pero ése es exactamente el punto. los diseños del archivo y la base de datos deben modelar la empresa a la que soportan. Estas decisiones no se pueden hacer en busca de satisfacciones técnicas.



La tabla siguiente resume las tres formas de normalización analizadas hasta ahora. Si la base de datos se diseña de acuerdo con los principios de normalización, la manipulación de los datos será más fácil.

Forma	Pasos
Primera forma normal	Cambiar todas las estructuras que no sean bidimensionales (es decir, grupos de repetición) en estructuras de registros bidimensionales.
Segunda forma normal	Eliminar los datos que no dependan totalmente de las llaves de registro
Tercera forma normal	Eliminar los datos que dependan transitivamente de las llaves primarias.

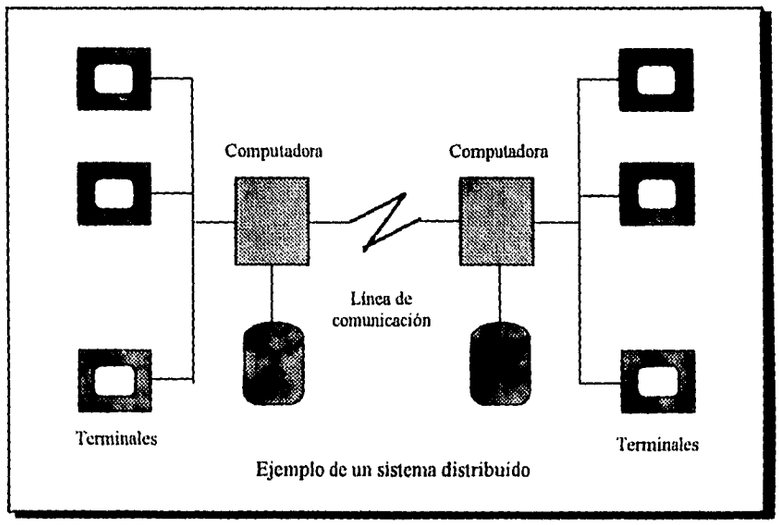
I.2.5. Bases de datos distribuidas.

I.2.5.1. Definición y características.

Una base de datos distribuida es aquella que no se encuentra físicamente en un sólo lugar, sino distribuida a lo largo de una red.

En general un sistema distribuido consiste en un conjunto de nodos conectados entre sí a través de una red. Cada nodo es en sí mismo un sistema de bases de datos. Consecuentemente, una base de datos distribuida puede concebirse como la unión de un conjunto de bases de datos centralizadas.

Un sistema distribuido de base de datos consiste en un conjunto de "localidades", cada una de las cuales puede participar en la ejecución de transacciones que accesen datos de una o más localidades.



Ejemplo de un sistema distribuido

I.2.5.2. Estructura de las bases de datos distribuidas.

Un sistema distribuido de base de datos consiste en un conjunto de localidades, cada una de las cuales mantiene un sistema de base de datos local. Cada localidad puede procesar transacciones locales, es decir, aquellas que sólo accesan información que reside en esa localidad. Además, una localidad puede participar en la ejecución de transacciones globales, es decir, aquellas que accesan información de varias localidades. La ejecución de transacciones globales requiere comunicación entre las localidades.

Este tipo de base de datos consiste en almacenar los datos en distintos lugares fisicos, e interconectarlos a través de red de computadoras separadas geográficamente. Es decir, los datos se almacenan el lugar donde se usan con mayor frecuencia, pero a través de la red de comunicaciones permanecen disponibles. Una de las ventajas de esta estructura es el procesamiento local de las operaciones y una percepción centralizada de los datos. De hecho la base de datos esta distribuida en el nivel bajo de la estructura de base de datos, pero no a nivel conceptual y de visión. Sin embargo, los costos de comunicación son elevados, además de ciertas dificultades técnicas para su implantación (la posibilidad de accesar información a pesar de las fallas en algunas localidades o líneas de comunicación).

Las computadoras pueden estar conectadas de acuerdo a las diversas topologías que existen en un sistemas de red. Las diferencias en las topologías se establecen de acuerdo a los siguientes criterios :

Costo de Instalación. Cuantos recursos económicos se contemplan para conectar fisicamente las localidades del sistema

Costo de Comunicaciones. Costo en tiempo y dinero que implica enviar un mensaje de un punto "A" al "B".

Confiabilidad. La frecuencia con que falla una línea de comunicación o una localidad.

Disponibilidad. La capacidad de accesar a cualquier punto de la red, aún cuando falle cualquier localidad.

Ventajas de una Base de Datos Distribuidas

Utilización compartida de los datos y distribución del control. Esta característica consiste en la capacidad que tiene cada punto para que pueda controlar hasta cierto punto los datos almacenados localmente. En el sistema distribuido, el encargado de todo el sistema es un administrador global. Sin embargo existen ciertas responsabilidades para los administradores locales. A pesar de cierto control indirecto de parte del administrador global, se tiene la posibilidad de contar con autonomla local.

Confiabilidad y disponibilidad. Para un sistema centralizado, la posibilidad de seguir trabajando cuando hay fallas en el sistema es completamente nula. En cambio, para un sistema distribuido es posible que las demás localidades puedan seguir trabajando. Además si los datos se repiten en otras localidades, la falla en cualquier punto de la red no será de gran importancia.

Agilización del procesamiento de consultas. Cuando se realiza una consulta que involucra datos de diversas localidades, es posible dividir la consulta en varias subconsultas que se ejecuten en forma paralela en distintas localidades.

Desventajas de una Base de Datos Distribuidas.

Costo de desarrollo de software. La estructuración de un sistema de base de datos distribuida es de mayor costo.

Mayor posibilidad de errores. Cuando se opera en paralelo las localidades del sistema, es más dificil garantizar que los algoritmos sean correctos. Aun es un campo de investigación el construir algoritmos para sistemas distribuidos.

Mayor tiempo extra de procesamiento. El intercambio de mensajes y los cálculos adicionales que se requieren para coordinar las localidades son en cierta forma tiempo extra que no existe en los sistemas centralizados.

Factores de Diseño para Bases de Datos Distribuidas.

A continuación se describen los factores importantes para el diseño de una base de datos distribuida.

Repetición de Datos. El sistema contiene varias copias idénticas de la relación. En caso extremo existe un a copia en todas los puntos del sistema.

Sin embargo esta característica tiene ventajas y desventajas que en seguida se describen:

- Disponibilidad. Si falla uno de los nodos del sistema, puede disponerse de la información en cualquier localidad. De esta forma el sistema puede seguir atendiendo las consultas de que impliquen una relación.
- Mayor paralelismo. La repetición de la información evita un mayor tráfico entre las localidades. Es decir , varios puntos de la red tiene la capacidad de procesar consultas que involucren a una relación repetida en diversos nodos.
- Mayor tiempo extra para actualizaciones. El sistema debe contemplar la consistencia e integridad de los datos, para evitar cálculos erróneos. Cada vez que se actualiza un dato en cualquier nodo del sistema, debe realizarse el cambio en todas las copias del sistema.

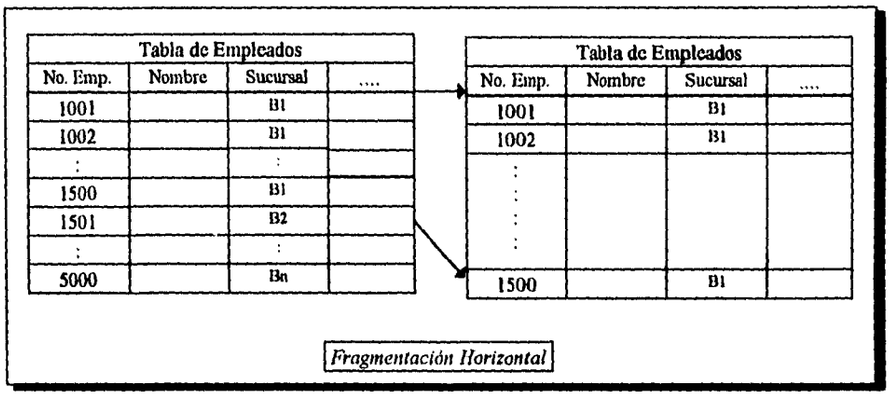
Fragmentación de la Información. En cada localidad se almacena un fragmento de la relación, la cual contiene la suficiente información para reconstruir la relación original. Existen dos esquemas para fragmentar una relación:

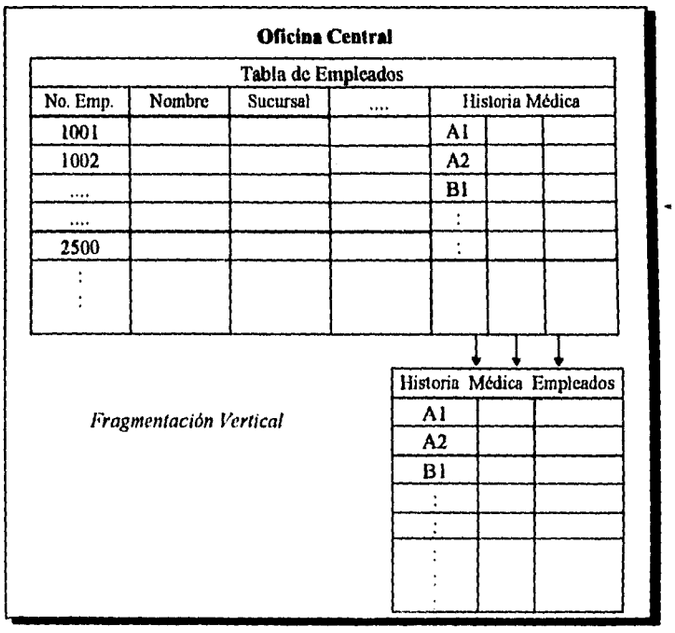
La Fragmentación Horizontal divide a la relación asignando a cada tupla de cualquier relación a uno o mas fragmentos.

Fragmentación Vertical. Esta fragmentación involucra la definición de varios subconjuntos. Dicho método consiste en agregar un atributo especial al esquema de la

relación. El atributo especial representa una dirección lógica o física para poder reconstruir la relación. Es importante que el usuario no pueda ver el atributo especial.

La repetición y fragmentación de la información puede aplicarse de forma recurrente, es decir, para cada relación y el subconjunto de fragmentos pueden repetirse en cada localidad y así sucesivamente.





La transparencia de la red es la capacidad del sistema de ocultar los detalles de la distribución de la información en la red. La asignación de nombres en los elementos de información es importante para la base de datos distribuida, para eliminar la duplicidad de nombres para diferentes conceptos. De esta forma surge la necesidad de un asignador central de nombres, perdiendo en cierta forma la autonomía de las localidades. Otra solución al problema es la posibilidad de que cada nodo ponga como prefijo su identificador de localidad a cualquier nombre que genere.

Una transacción se define como una unidad de programa atómica, de tal forma que se debe ejecutar todas las instrucciones que involucran esta acción o no se ejecuta ninguna. Para garantizar la atomicidad de dicha instrucción se establecen varios esquemas de recuperación y control de concurrencia para un sistema centralizado. Sin embargo para un sistema distribuido es mucho mas difícil garantizar la propiedad de atomicidad, debido a la posible participación de varias localidades. Para conservar esta función indispensable en el sistema es necesario una manejador de transacciones.

Para un sistema distribuido, los distintos manejadores de transacciones locales cooperan para llevar acabo las transacciones globales. Cada punto del sistema contiene dos subsistemas:

- Manejador de transacciones. Ejecuta las transacciones desarrolladas en su propia localidad. Cada transacción puede ser parte de una transacción global o solo una operación local.
- Coordinador de transacciones. Su función principal es administrar la ejecución de diversas transacciones que se iniciaron en dicho punto.

Este tipo de manejador de transacciones aún conserva funciones de un sistema centralizado. Por ejemplo, el mantener una bitácora de recuperación, coordinar la ejecución en paralelo de las transacciones a través de un esquema de control de concurrencia.

Es importante mencionar que un sistema distribuido de datos adolece de las mismas fallas que un sistema centralizado. Además es indispensable tener en cuenta otro tipo de fallas, tales como :

- Falla total de un punto de la red.
- Interrupción en una línea de comunicación.
- Pérdida de mensajes.
- Fragmentación de la red.

Para que el sistema distribuido funcione de la mejor forma es indispensable que sean detectados estos errores, se reconfigure el sistema en su totalidad, con el objetivo de aislar el problema, y una vez reparada la falla sea recuperado el nodo.

Las localidades del sistema pueden conectarse físicamente de diversas formas. Las distintas configuraciones se presentan como gráficas cuyos nodos corresponden a esas localidades. Una arista del nodo A al B corresponde a una conexión directa entre las dos localidades. Las diferencias principales entre estas configuraciones son:

- Costo de instalación. El costo de conectar físicamente las localidades del sistema.
- Costo de comunicaciones. El costo en tiempo y dinero que implica enviar un mensaje de la localidad A a la B.
- Confiabilidad. La frecuencia con que falla una línea de comunicación o una localidad.
- Disponibilidad. La posibilidad de accesar la información a pesar de fallas en algunas localidades o líneas de comunicación.

Concesiones al distribuir la base de datos:

Existen varias razones para construir sistemas distribuidos de bases de datos: compartir la información, mejorar la confiabilidad y la disponibilidad, y agilizar el procesamiento de las consultas. Sin embargo, estas ventajas vienen acompañadas de varias

desventajas: mayores costos de desarrollo del software, mayor posibilidad de errores y aumento en el costo extra del procesamiento.

Ventajas de la distribución de la información:

La principal ventaja de los sistemas distribuidos de bases de datos es la capacidad para compartir y accesar la información de manera confiable y eficiente.

Utilización compartida de los datos y distribución del control.

Si varias localidades diferentes están conectadas entre sí, entonces un usuario de una localidad puede accesar datos disponibles en otra localidad. Por ejemplo, ... un funcionario de las oficinas centrales puede accesar información de alguna de las sedes regionales ...

La ventaja principal de compartir los datos por medio de la distribución de la información es que cada localidad puede controlar hasta cierto punto los datos almacenados localmente. En un sistema centralizado, el administrador de base de datos de la localidad central controla la base de datos. En un sistema distribuido, existe un administrador global de la base de datos que se encarga de todo el sistema. Parte de estas responsabilidades se delegan al administrador de la base de datos de cada localidad. Dependiendo del diseño del sistema distribuido, cada administrador local podrá tener un grado de autonomía diferente, que se conoce como autonomía local. La posibilidad de contar con autonomía local es en muchos casos una ventaja importante de las bases de datos distribuidas.

Confiabilidad y disponibilidad

Si se presenta una falla en una localidad distribuida, es posible que las demás localidades puedan seguir trabajando. En particular, si los datos se repiten en varias localidades, una transacción que requiere un dato específico puede encontrarlo en más de una localidad. Así, la falla de una localidad no implica necesariamente la desactivación del sistema. El sistema debe detectar cuando falla una localidad y tomar las providencias necesarias para recuperarse de la falla.

I.2.5.3 Centralización, descentralización y distribución informática.

A lo largo de la década de los 70's el campo de las minicomputadoras aumentaba y empezaban a coexistir con la gran computadora central en una misma compañía. Al final de los 70's y principios de los 80's se concretó, asimiló y absorbió el fenómeno de la microcomputadora.

En la actualidad, cada usuario, o área funcional de una empresa, puede disponer a un costo razonable de sus propios recursos informáticos para resolver un problema, sin tener que recurrir a una computadora central.

Pero ésta no siempre es la mejor alternativa, los costos de las soluciones descentralizadas no son necesariamente más bajos que los de las soluciones centralizadas, además, en muchos casos no es posible obtener el mismo grado de servicio y nivel de información integrada como cuando se emplea una gran computadora.

Los argumentos sobre centralización versus descentralización, control de la oficina central versus control local, una gran computadora versus pequeñas computadoras, se han estado manejando desde la aparición de las minicomputadoras en la década de los 60.

La discusión adquirió diferentes matices con la proliferación de las minicomputadoras y el proceso distribuido. En primer lugar, la economía de escala de la gran computadora central fue perdiendo fuerza como argumento centralizado, ya que, en muchos casos es más barato tener un conjunto de minicomputadoras que una gran computadora. Por otra parte, el proceso distribuido introdujo una tercera alternativa entre la centralización total y la descentralización total. Se pueden utilizar varias computadoras locales unidas entre si y con la computadora central mediante líneas de comunicación. El trabajo se puede así dividir entre todas las computadoras que forman parte de la red en multitud de variantes, permitiendo así compartir todos los recursos de la red entre los usuarios.

Los argumentos de lo que debe ser centralizado y lo que debe ser distribuido se aplican a otras áreas de la vida además de a las computadoras.

En la mayoría de los casos es mejor una mezcla de facilidades locales y centrales. En el caso de las computadoras y las empresas, los procesos centrales deben ser aquellos que se beneficien de la iniciativa local, conocimiento de los problemas locales y control y autoridad local. La razón última de la tendencia a diseñar soluciones distribuidas es que posibilitan una mejor adecuación entre recursos y necesidades, permitiendo acercar los recursos al punto donde está el usuario que lo necesita.

El sistema informático de una organización es centralizado si toda la capacidad de proceso y el banco de datos están localizados en un sólo centro, y asimismo, todo el trabajo de desarrollo de nuevas aplicaciones lo lleva a cabo y está dirigido por ese mismo centro. Los usuarios remotos pueden ser servidos a base de un transporte físico de las entradas y salidas de información o estar ligados al centro mediante un conjunto de terminales conectados a él por una red de comunicaciones.

En el extremo opuesto, un sistema es descentralizado si cada área funcional de la compañía, o grupo de ellas, dispone de una computadora propia e independiente. Cada unidad es responsable, con sus propios recursos, del desarrollo y operación de la parte del sistema informático total de la compañía que la unidad precisa para sí misma.

Ahora bien, hay que tener en cuenta que existen varios elementos objetos de una posible descentralización. Estos son los equipos informáticos, el personal de explotación de la computadora, el personal de desarrollo de nuevas aplicaciones, los datos de la compañía y las aplicaciones o procesos. se pueden descentralizar unos elementos y otros no, y algunos de ellos en su totalidad o en parte.

Un sistema informático es distribuido si se emplean múltiples computadoras con capacidad de proceso propio, que están interconectados entre sí y, sobre todo, se da una utilización compartida entre ellos de los recursos disgregados por toda la red. Ello conlleva a una coordinación considerablemente centralizada en el diseño y operación de los recursos informáticos por los usuarios, los técnicos, las computadoras y las aplicaciones.

Así, por ejemplo, en un sistema distribuido la simple aplicación de nómina será centralizada si las computadoras locales recogen los datos y los transmiten a la computadora central sin hacer proceso sobre ellos, y es en está computadora donde se realizan todos los procesos requeridos. Al contrario, será descentralizada si cada computadora local utiliza y procesa los datos del personal de un área y realiza la nómina sin intervención de ninguna otra computadora, aunque posteriormente envíe algunos datos a la computadora central para otros procesos adicionales. Finalmente, se entenderá que es una aplicación distribuida si las computadoras locales realizan algún proceso de los datos que les llegan, como validación o cálculo de las primas locales , y envían estos resultados a la computadora central en donde se termina el proceso de la nómina.

Los conceptos descentralización y distribución están a menudo identificados, sin embargo, en la concepción más generalmente admitida, la distribución se diferencia fundamentalmente en que implica una cooperación y compartición de los recursos dispersos para realizar alguno de los procesos del sistema.

I.2.5.4. Nivel de descentralización y distribución.

Todo lo que forma parte de los recursos de un sistema puede ser objeto de reparto, adoptándose distintos niveles de distribución autonomía e independencia, según el grado de descentralización concreta de cada caso. No se debe plantear solamente el reparto del equipo informático sino, también, el del personal, la red de comunicaciones, la base de datos y las aplicaciones, en el grado suficiente para alcanzar el nivel de distribución deseado.

Es posible diferenciar tres niveles de descentralización más comúnmente utilizados.

- Captura de datos descentralizada.
- Computadoras autónomas.
- Computadoras interconectadas.

Captura de datos descentralizada.

En este nivel todo el sistema es fuertemente centralizado, exceptuando la captura remota de los datos, que se realiza mediante terminales inteligentes o microcomputadoras,

que suministran facilidades de validación de datos, detección de errores y ayudas automatizadas a la entrada. Estas terminales inteligentes pueden estar conectadas al ordenador central que controla todo el sistema. Las estaciones remotas no son conscientes de la existencia de las demás estaciones.

Computadoras autónomas.

En este segundo nivel se procesan remotamente las transacciones, no sólo la captura de datos, se obtienen determinados informes administrativos.

La configuración más utilizada para este nivel es una computadora central conectada a microcomputadoras con capacidad suficiente para procesar autónomamente parte de sus aplicaciones.

Computadoras interconectadas.

En este nivel se aprovechan las ventajas de una red de computadoras, los recursos de ésta pueden ser compartidos por todos los equipos interconectados en cuanto a líneas, datos, programas, almacenadores y procesos.

Este nivel consiste en la verdadera red de proceso distribuido. Comúnmente se tiene un coordinador central de la red aunque no es técnicamente imprescindible.

En la resolución de un problema práctico se debe empezar con elegir el nivel de descentralización que se quiere adoptar y, a continuación, se definirá el número de computadoras que se desean instalar, las conexiones entre ellos y su capacidad de transferencia de datos, las aplicaciones que residirán en cada uno de ellos, las datos a almacenar en cada punto, etc.

I.2.5.5. Criterios a favor y en contra de la centralización.

La polénica de la centralización frente a la descentralización informática, se viene discutiendo desde los años sesenta sin que haya quedado aclarada.

La solución mejor para cada caso particular no es evidente ni única, ya que los criterios a los que hay que dar respuesta incluyen factores políticos y técnicos. Muchas veces estos criterios tienen elementos contrapuestos y no existe una solución que cumpla todos los requisitos, lo que obliga a adoptar una alternativa de compromiso.

En seguida se presenta un esquema con los criterios que se manejan a la hora de elegir entre un sistema centralizado o descentralizado. Estos criterios se han agrupado en la siguiente cuatro categorías.

- Consideraciones de costo.
- Consideraciones técnicas.
- Desarrollo de aplicaciones.
- Consideraciones políticas

Consideraciones de costo

A favor de la centralización	*A favor de la descentralización*
Costo del Procesador	
Se dan economías de escala en el hardware de una gran computadora central, frente a múltiples computadoras de potencia total equivalente.	Los precios de las microcomputadoras bajan más rápidamente en proporción. Las microcomputadoras utilizan tecnología más avanzada que abarata costos, al renovarse con mayor rapidez.
Costo de almacenamiento	
El costo por bit almacenado es mucho menor en las grandes unidades centrales. La descentralización origina múltiples copias de los mismos datos encareciendo el costo del almacenamiento.	La centralización origina costos de transmisión para recoger los datos lejanos. Estos costos no se producirían en muchos casos si se almacenasen los datos localmente.
Costos de comunicación	
	La descentralización disminuye el número de datos transmitidos. Los avances en la tecnología de las computadoras vienen siendo mayores que los avances de la tecnología de comunicaciones. Es más caro transmitir que procesar localmente.

A favor de la centralización	*A favor de la descentralización*
Recursos compartidos	
Para un costo dado, la centralización permite disponer de mejores recursos.	Si el objetivo final es llegar a una red de proceso distribuido, es favorable tener los recursos descentralizados previamente.
Costos de instalación	
	Las grandes computadoras necesitan acondicionamientos especiales y personal especializado muy costosos, tanto en la instalación inicial como en el mantenimiento posterior.
Costos de personal	
El costo total de personal informático es menor en los sistemas centralizados al evitar redundancias.	
Planificación de costos	
La planificación centralizada puede minimizar mejor los costos totales.	La planificación y seguimiento de costos es difícil de hacer en un sistema central que barca a toda una corporación. La descentralización permite fijar y perseguir objetivos con responsabilidades repartidas más fáciles de alcanzar.
Escalada de costos	
En las instalaciones descentralizadas los costos informáticos se han incrementado, muy por encima de las previsiones, debido a la tendencia a mejorar continuamente el equipo y aumentar los recursos humanos dedicados.	

Consideraciones técnicas

A favor de la centralización	*A favor de la descentralización*
	Seguridad
En un sistema central es más fácil diseñar y controlar las medidas de seguridad y privacidad adecuadas.	Existe un alto riesgo estratégico de averías o catástrofes en una organización central, aunque exista duplicidad. En una estructura descentralizada el daño de una avería o fallo de los mecanismos de seguridad queda reducido a un daño local.
	Entrada de datos
Se consigue la ausencia de errores en un sistema centralizado mediante terminales adecuados.	En una descentralización los usuarios remotos son responsables de la calidad y rapidez de su propia entrada de datos. Reduciéndose así a los errores.
	Carga de trabajo
	El volumen de trabajo en muchas compañías es demasiado elevado para un sistema centralizado, esto es cada vez más cierto en la actualidad por la permanente ampliación del campo de utilización de computadoras dentro de las compañías.
	Cambios y ampliaciones
Las grandes computadoras centrales incluyen utilidades para facilitar el cambio. La centralización evita la proliferación de sistemas, aplicaciones y datos incompatibles, que son extremadamente difíciles e integrarse o modificarse con posterioridad.	En los sistemas locales el cambio es más sencillo por estar involucrados menor número de personas y programas. Las ampliaciones se puede realizar allí donde se necesiten en una descentralización; en un sistema central las ampliaciones presentan dificultades y suelen hacerse después de lo aconsejado.
	Software
Los grandes sistemas centrales permiten utilizar software más potente.	El tiempo de respuesta y el rendimiento de las grandes computadoras son, a menudo, pobres para algunas aplicaciones on-line Normalmente se necesitan sistemas de alta especialización y el software de las microcomputadoras progresa a gran velocidad.

A favor de la centralización	*A favor de la descentralización*
	Programación
Los grandes sistemas ofrecen ayudas más potentes para el diseño, conversión y mantenimiento de programas Es más fácil para un grupo central de analistas y programadores generar programas mejor estructurados y documentados.	En un sistema descentralizado las aplicaciones sólo tienen que cumplir las necesidades de una localidad, ello permite que: a) El programa pueda ser realizado con los propios medios locales. b) Se aproxime el problema al diseñador, por lo que la solución será más adecuada. c) Los cambios sean más sencillos porque involucran a menos usuarios.
	Información de gestión
Una estructura informática centralizada tienen los datos integrados, disponibles para un sistema de información gerencial.	
	Previsión
Los problemas graves de una mala descentralización no pueden aparecer hasta transcurrido un largo plazo, en este aspecto, descentralizar requiere una mayor previsión de los posibles problemas.	Los sistemas descentralizados necesitan menor previsión al poder detectar y resolver los problemas rápidamente.
	Control y auditabilidad.
Un sistema descentralizado requiere mayores esfuerzos de coordinación supervisión y control del sistema total para evitar la aparición de aplicaciones o datos incompatibles. Una descentralización puede, con mayor facilidad, empeorar de modo progresivo hasta hacerse difícil o imposible auditar, por deficiencias en los mecanismos de seguridad o documentación.	

Desarrollo de aplicaciones

A favor de la centralización	*A favor de la descentralización*
Cualificación	
La centralización rentabiliza, en mayor medida disponer de especialista en cada área.	
Rendimiento	
Los especialistas en una organización centralizada pueden aumentar la eficacia aplicando técnicas más avanzadas.	La descentralización acerca el diseñador al problema, con lo que aumenta el rendimiento del desarrollo de aplicaciones.
Planificación del desarrollo	
En un sistema centralizado se puede planificar mejor valorando adecuadamente las necesidades globales de la corporación.	Las necesidades informáticas son mejor valoradas, justificadas y planificadas cuando se hacen localmente. La supervisión del cumplimiento de una planificación es difícil hacerla en un sistema centralizado que abarca toda corporación; la división que se hace en una descentralización ayuda a la supervisión local independiente. La capacidad de adaptación a situaciones no previstas es mayor en estructuras descentralizadas, por lo que requieren menor planificación.
Capacidad de reacción	
	Los usuarios remotos con recursos propios reaccionan más rápidamente a sus necesidades; la mayoría de los sistemas centrales tienen aplicaciones atrasadas en espera de ser implementadas en algún momento.
Recursos humanos	
La organización centralizada evita la duplicidad de esfuerzos y recursos.	En general, el grupo central no consigue un dimensionamiento suficiente para responder con prontitud a todas las necesidades de los usuarios remotos.

Consideraciones políticas

A favor de la centralización	*A favor de la descentralización*
	Autoridad
La estructura centralizada o descentralizada de una compañía es un factor clave a favor de que la informática esté organizada de la misma manera que la propia compañía.	Los directores de los centros remotos se resisten de un control central y que les resta eficacia y pide mayor autoridad y autonomía local. La autonomía refuerza las responsabilidades locales, aumentado la eficacia.
	Influencia
Una configuración centralizada puede dar más poder a la alta dirección, el argumento se aplica según sea la política de la compañía de concentrar o dispersar el poder de la información.	La influencia que posee el personal informático es muy fuerte en un sistema central, pudiendo un grupo pequeño de personas parar toda la capacidad informática de la compañía.
	Presupuestos
La planificación de costos informáticos puede seguir mejor la política económica de la compañía en un sistema central.	En una descentralización se permite que los usuarios remotos sean responsables de sus propios gastos, consiguiéndose una inversión más controlable del presupuesto informático.

I.2.5.6 Componentes de un sistema distribuido.

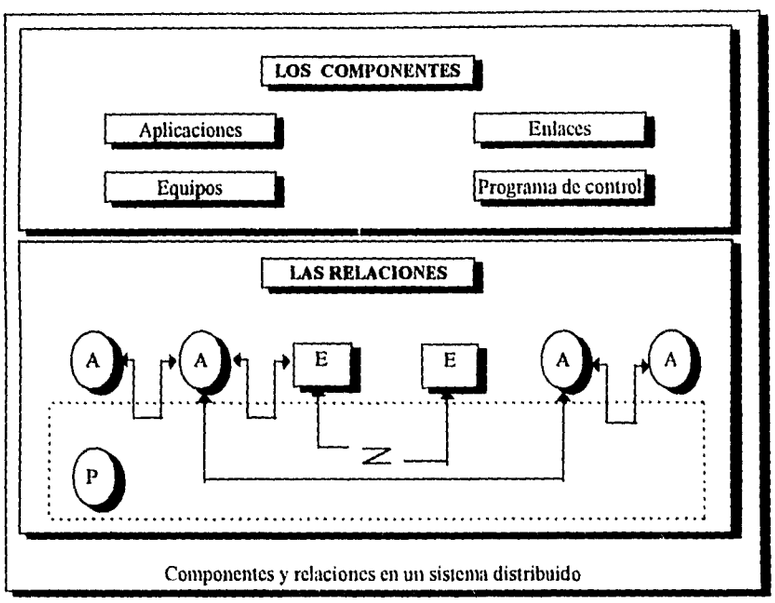
La característica fundamental de un sistema distribuido es la participación de varias computadoras distantes entre sí para realizar cooperativamente una determinada función.

En un sistema distribuido se pueden identificar los siguientes componentes:

- Equipo de cómputo.
- Aplicaciones (programas de usuario).
- Enlaces (comunicación entre los equipos distantes).
- Programas de control.

Se pueden establecer determinadas relaciones entre los componentes antes mencionados, entre equipos y aplicaciones, para que éstas utilicen los recursos pertenecientes a cualquiera de los equipos distribuidos, entre aplicaciones entre sí, para el

intercambio de información; y entre las aplicaciones y los enlaces, para que ellas utilicen los elementos de comunicaciones como un recurso más. Los programas de control se encargan del establecimiento de dichas relaciones.



Componentes y relaciones en un sistema distribuido

En el sistema centralizado, la responsabilidad del establecimiento de las relaciones corresponde al sistema operativo y programas de control asociados, quienes proporcionan el servicio de comunicaciones entre las aplicaciones entre sí y entre las aplicaciones y los recursos informáticos.

- Desde el punto de vista del usuario un sistema distribuido no se diferencia en esto de uno centralizado, porque las relaciones que se establecen son las mismas, si se prescinde de los recursos de comunicaciones. Las diferencias aparecen en los siguientes puntos:
- En un sistema distribuido, algunas aplicaciones de usuario deberán ser conscientes de que el sistema es distribuido y de que determinadas funciones se realizarán cooperativamente con otra aplicación residente en otra computadora.

- Los programas de control están distribuidos entre los diferentes equipos que constituyen el sistema, por lo que no se reducen a un sistema operativo.
- En un sistema distribuido se requiere de enlaces de comunicaciones entre las computadoras, soportados por un conjunto de elementos hardware y software.

Es común utilizar el término "*Arquitectura de un Sistema Distribuido*" para designar el conjunto de sistema operativo distribuido y enlaces de comunicaciones, por lo que las diferentes arquitecturas con que se cuenta están en base a las variaciones tanto en uno como en otro elemento.

Si lo que se distribuye son únicamente los datos, un sistema de base de datos distribuidos puede proporcionar, para las aplicaciones, la búsqueda del dato solicitado en la computadora donde se encuentre, sin que la aplicación tenga que ser consciente de ello. En este caso las aplicaciones son independientes de la ubicación de los datos y no hay diferencia para el usuario respecto a un sistema convencional.

La comunicación entre los componentes de un sistema distribuido se realiza mediante el establecimiento de un camino lógico por los programas de control entre dos componentes. Un camino lógico es una vía genérica de comunicación, está definida por diferentes parámetros de velocidad, volumen de información, componentes, origen, destino, bidireccional, mecanismos de control del intercambio de información, etc.

El establecimiento físico de un camino lógico se hace mediante los enlaces materiales de que se dispone en el sistema distribuido.

Para establecer comunicación entre dos componentes se requiere un camino lógico único y distinto de los demás establecidos. Un camino físico puede aportar varios caminos lógicos simultáneos, si su capacidad de transmisión y los programas de control lo permiten.

I.2.5.7. Arquitectura de un sistema distribuido.

Empezaremos con definir que son las redes de computadoras.

Son varias las definiciones aceptadas por la industria; la más sencilla de todas es probablemente, la siguiente: un grupo de computadoras (y terminales, en general) interconectados a través de uno o varios caminos o medios de transmisión, la mayoría de las veces, este medio de transmisión es la linea telefónica , debido a su fácil accesibilidad.

Las redes tienen una finalidad concreta. transferir e intercambiar datos entre computadoras y terminales. en el intercambio de datos lo que permite funcionar a los múltiples servicios a distancia que ya consideramos parte de nuestras vidas: cajeros automáticos, terminales punto de venta, etc.

Las redes de computadoras presentan varias ventajas importantes de cara a los usuarios, ya sean empresas o particulares.

1.- Mediante una red puede conseguirse que una organización que tenga empresas dispersas en diferentes partes del país intercambie información mediante computadoras, y que los programas y datos necesarios estén al alcance de todos los miembros de la organización.

2.- La interconexión de computadoras permite que varias máquinas compartan los mismos recursos. Así, por ejemplo, si una computadora se satura por estar sometida a una carga de trabajo excesiva, podemos utilizar la red para que otra computadora se ocupe de ese trabajo, consiguiendo así ese mismo aprovechamiento de los recursos.

3.- Las redes pueden resolver también un problema de especial importancia: la tolerancia ante fallos. En caso de que una computadora falle otro puede asumir sus funciones y su carga de trabajo, algo de particular importancia en los sistemas de control de tráfico aéreo. Si una computadora falla, las computadoras de reserva entrarán en funcionamiento rápidamente y tomarán el mando de todas las operaciones de control, sin que en ningún momento llegue a existir peligro para los pasajeros.

4.- El empleo de las redes confiere una gran flexibilidad a los entornos laborales. Los empleados pueden trabajar desde sus casas, utilizando terminales conectadas con la computadora de la oficina.

Funciones básicas de una red.

Las redes de computadoras fueron creadas para producir comunicación entre máquinas inteligentes. El análisis del sistema telefónico, que es una red que permite la comunicación entre personas es un buen ejemplo para comprender las funciones básicas de una red de computadoras.

- Antes de establecer la comunicación es necesario conocer el número de personas con quien deseamos comunicarnos.
- Después de marcar el número es imprescindible saber si se estableció correctamente la comunicación.
- De no ser así, es preciso conocer cuál es la razón de este falla que puede ser debido a que:
  - El teléfono receptor está ocupado.
  - La red se encuentra saturada.
  - El número se marcó erróneamente.
  - Existe una falla de equipo, etc.
- Dependiendo del problema, se tomará la solución necesaria para volver a intentar la comunicación.

Una vez que la comunicación eléctrica se establece correctamente, existe un protocolo (informal) de comunicación entre los interlocutores que tiene como objetivos:

1.- Saber si la persona requerida se encuentra conectada a la red.

2.- Identificación de la persona que llama.

3.- Una vez que las partes están de acuerdo comienza la sesión de conversación.

Todas las etapas que aquí se han mencionado son las mismas que se necesitan para establecer una comunicación hacia una red de datos. Desde luego muchas de esta etapas se realizan automáticamente, los protocolos para las transmisiones dejan de ser informales para convertirse en formales y, son en general, muy complejos.

Si deseamos iniciar una comunicación entre computadoras es necesario seguir un protocolo.

Después de realizar el protocolo necesario, las máquinas están en condiciones de ejecutar la transmisión de datos en una sola dirección o bien un dialogo. En este contexto podemos encontrar una computadora que dentro del conjunto de posibilidades puede cambiar el uso de una base de datos que reside en otra computadora. Un operador puede ingresar datos en una terminal, un usuario con una microcomputadora puede dialogar con una computadora que se encuentra a gran distancia, etc.

Hay tres fases principales para la comunicación: establecer la comunicación, la conducción de la sesión y finalización de la comunicación.

Por lo tanto las funciones básicas que deben existir en una red de computadoras la podemos separar en cinco procesos.

1.- Conexión del camino de transmisión.	Proceso físico
2.- Establecimiento de la sesión.	
3.- Conducción de la sesión.	Proceso lógico.
4.- Finalización de la sesión	
5.- Desconectar el camino de transmisión	Proceso físico.

El concepto que caracteriza a los sistemas distribuidos, es la participación en el tratamiento de información de varias computadoras, distantes entre sí, unidas mediante enlaces de comunicación se ha venido utilizando para establecer las arquitecturas de los sistemas distribuidos, llegando a la siguiente clasificación:

- Con base a la distancia entre las computadoras.

  - Sistema multicomputadora (mismo local).
  - Redes locales (mismo complejo industrial).
  - Redes de computadoras (grandes distancias).

- En base a la organización de los enlaces:

  - Red de anillo.
  - Red de estrella.
  - Red tipo bus.
  - Red irregular.

- En base a la transferencia de información en la red:

  - Red conmutada:
    Conmutación de circuitos.
    Conmutación de paquetes.
  - Red no conmutada:
    Punto a punto.
    Multipunto.

A continuación se describen las características de algunas de éstas arquitecturas.

Sistema multicomputadora.

Consiste en varias computadoras normalmente especializadas en funciones distintas e interconectadas entre sí mediante enlaces especiales para estar muy cercanas entre ellas. En general se corresponden, con unidades especializadas en el control de periféricos, gestión de comunicaciones, etc. Físicamente suelen estar ubicados en la misma sala a distancias inferiores a los diez metros.

Redes locales

Para las distancias limitadas, en torno a los diez kilómetros, se han consolidado desde los años 80 estas redes que son fruto de trasladar la experiencia obtenida en las redes de computadoras mediante enlaces con circuitos telefónicos, a redes configuradas con enlaces de transmisión más sencillas, cable coaxial o por hilo trenzado, que requieren equipos

electrónicos de comunicación más baratos. por otra parte al ser distancias limitadas y estar los equipos conectados físicamente con un cable continuo exclusivamente dedicado a ello, permiten aprovechar toda la capacidad de transmisión del cable, por lo que la velocidad y volumen de transferencia de información es muy superior a las redes convencionales.

En estas redes locales se consigue de manera eficiente compartir equipos especializados de impresión, almacenamiento y proceso entre todos los elementos conectados a la red, constituyendo la solución más completa para sistemas distribuidos localizados en un mismo edificio o en un complejo industrial limitado geográficamente.

Redes de Computadoras

Las redes de computadoras surgieron en los años 60 como una solución para conectar computadoras situadas en lugares remotos, con el objetivo inicial de transferir datos de un lugar a otro utilizando medios de comunicación preexistentes, como eran los circuitos telefónicos.

Posteriormente, al consolidarse la comunicación de datos y aumentar la velocidad de transferencia de información se utilizaron estas redes para compartir los recursos de los elementos conectados a la red, dando lugar a los sistemas distribuidos.

La utilización de enlaces telefónicos permite conectar equipos situados a centenares de kilómetros. El desarrollo posterior de las redes públicas de transmisión de datos han ampliado las posibilidades de interconexión de equipos, permitiendo incluso la conexión internacional a través de ellas, al estar la redes públicas nacionales también conectadas entre sí.

El hecho es que hoy día los fabricantes de equipos informáticos y los suministradores de enlaces de comunicaciones ofrecen un amplio abanico de alternativas de conexión para construir un red privada propia, una red basada en servicios públicos o redes mixtas para la conexión entre equipos informáticos, con lo que las redes de computadoras son actualmente una realidad madura y flexible para adaptarse a cada necesidad.

Redes de Anillo.

Se compone de una línea unidireccional que conecta a todos los nodos de la red. De un nodo llega y de él sale una única línea de comunicación. Debido a la configuración de esta red, es preciso que los nodos sean relativamente dependientes (microcomputadoras).

La interface con el anillo es simple y requiere poca memoria. Para evitar el almacenamiento de todo un bloque de datos la información se transmite de inmediato. Todo mensaje debe regresar a su punto de partida.

El destinatario solo lo copia, porque es necesario que el mensaje vuelva al emisor.

Si dos equipos empiezan a transmitir simultáneamente existe un choque de los mensajes, pues los anillos poseen una elevada velocidad de transporte, debido a que no hay memoria suficiente para demorar la retransmisión de un bloque, por lo tanto existe el problema de la asignación del anillo.

Métodos para administrar el anillo.

Vamos a llamar ficha o token a un mensaje que va circulando por la red. Cuando un nodo recibe la ficha le indica que la vía está libre y puede transmitir. Al terminar la transmisión el nodo debe incorporar la ficha al final de su mensaje, para que los demás nodos puedan saber que a partir de ellos pueden transmitir. Siempre debe haber exactamente una ficha en la red.

Carros de tamaño fijo.

Se considera el anillo como un tren con carros; éstos se encuentran ocupados si va un mensaje y desocupados si no es así. Cuando un nodo inserta un mensaje en un carro, también activa una señal para que los demás puedan leer que está ocupado. Una vez que el mensaje se transmite correctamente, se desactiva la señal para que otro nodo pueda utilizarlo. La dificultad que se produce es la aparición de un carro ocupado que ningún nodo va a eliminar, por ejemplo porque la dirección del nodo destino está mal puesta. Este

problema se resuelve mediante una estación de servicio, que tiene la función de verificar que no existan anomalías como la anterior mencionada. Así, si pasa por la estación de servicio dos veces exactamente el mismo mensaje, la estación lo elimina pues quiere decir que ningún nodo se ha hecho cargo de él.

Inserción de Registro

Se inserta un registro de desplazamiento con el mensaje. La inserción sólo puede realizarse si no está transmitiendo, o directamente al final de un mensaje.

El registro se saca del nodo.

En este método no hay que esperar fichas o carros (casos en el que el anillo está ocioso). Sólo se producen en el peor caso demoras por el tamaño del mensaje que se está mandando.

Ventajas del anillo

- Cada interface de la red, genera la señal.
- El retorno del mensaje permite verificar errores y recibir confirmaciones.
- Los límites de la longitud máxima de cada línea de transmisión se aplican a cada tramo.

Desventajas del anillo

- El trazado de cables debe regresar al punto de partida.
- Todos los equipos deben estar activos en la línea.
- Si falla un nodo el anillo no funciona.

Redes de Estrella.

Esquemáticamente pueden representarse por un árbol cuya raiz es el equipo (nodo) central. Se caracterizan por su simplicidad y porque hay una sola ruta entre dos nodos de la red.

Se pueden ahorrar líneas agregando <<concentradores>>; pero provocando simultáneamente un carga extra en la línea; normalmente el concentrador es un computadora en la memoria necesaria para almacenar los mensajes en el camino.

Un problema importante es la configuración de esta arquitectura:

1.¿Donde colocar los concentradores?
2.¿Cómo trazar las líneas de comunicación?

Las soluciones a estos problemas no son fáciles, por lo general, lo que lleva a resolverlos con estrategias de teoría de colas, que recurren a modelos de localización y búsqueda de caminos.

Uno de los problemas más típicos de esta organización se produce cuando alguno de los nodos falla porque no existen caminos alternativos para los mensajes.

Redes de Conmutación de Paquetes

Corresponde al primer tipo de redes digitales. La red almacena paquetes en una ruta y los va retransmitiendo hacia su destino. Por tanto no es necesario que exista una conexión directa entre nodos los que se van a comunicar. Este sistema se caracteriza porque los usuarios se conectan a estas computadoras por medio de líneas de comunicación telefónica.

Ventajas de la estructura de esta red.

- Multiplexión en los recursos de esta red.
- Sólo se gastan recursos durante la retransmisión del paquete.
- Si un camino de comunicación falla, existen caminos alternativos.
- Mejor adaptación al tipo de uso de computación: en computación puede superarse fácilmente el problema de que los paquetes lleguen en forma desordenada.

Desventajas de la estructura de esta red.

- Demora variable en la entrega de los paquetes.

- Este tipo de redes presenta muchos problemas de diseño.
  - Como interconectar los nodos.
  - Selección de rutas.
  - Adaptación en fallos de la red.
  - Adaptación a cambios en el tráfico.

Red Multipunto (Multi-Drop)

Este sistema fue creado con la idea de tener más de dos terminales en línea. Las primeras aplicaciones persiguieron siempre el objetivo de conectar varias terminales a una computadora. La principal ventaja que ofrece este sistema es el ahorro de líneas de comunicación para lograr que todos los equipos estén interconectados, la gran desventaja se produce cuando la capacidad de la línea se satura, lo que conduce a que aumente el tiempo de respuesta de cada nodo de la red.

En la actualidad la tendencia más común es contar con muchas microcomputadoras que se comuniquen entre sí. La idea central es interconectarlas todas con un solo cable. Sin embargo surge una dificultad: la imposibilidad de repartir adecuadamente en frecuencias la capacidad de la línea, lo que obliga a controlar el uso de la línea para transmisiones simultáneas.

Para solucionar este problema existen dos familias de mecanismos de solución. Veamos en qué consisten cada uno de estos mecanismos.

Asignación de turnos explícitos (Polling).

Este mecanismos se realiza mediante una consulta cíclica a todos los nodos de la red (este método se utiliza frecuentemente en redes que tienen todos los equipos en una distribución centralizada).

Un esquema básico de esta estructura es la siguiente:

El tráfico de información se produce de computadora a terminal y viceversa.

Generalmente la transmisión es síncrona porque se necesitan velocidades de comunicación muy elevadas, de hecho los protocolos de transmisión síncrona incluyen control de una línea multipunto. Las transmisiones del computador deben identificar la terminal al cual va dirigido el mensaje, esto necesariamente obliga a que las terminales de éste deban tener direcciones únicas. Porque es necesario que la computadora tenga la dirección de cada terminal y así les haga llegar los mensajes correctamente a cada uno.

La computadora debe enviar a través de las líneas un mensaje especial a cada terminal; el mensaje es de la forma: << si tiene algo que decir, dígalo >> . Ante esto la terminal debe transmitir, ya sea un bloque de datos o al menos una confirmación de que se recibió el mensaje.

Un punto importante que debemos considerar ahora, es el tiempo necesario para obtener acceso a la línea. Este tiempo a menudo es esencial al dimensionar un sistema como el que aquí descrito. Es posible dar preferencia a algunas terminales, preguntándose varias veces por cada ciclo de consulta (lo que se denomina consulta por lista o roll- call polling).

Se puede lograr una mejora del funcionamiento de este sistema haciendo que la terminal que no transmite genere la pregunta a la siguiente terminal. La última terminal del ciclo pasa el control a la computadora (este método es llamado consulta con paso automático o nubpolling), lo que se logra con esta mejora es evitar que la computadora espere respuestas de terminales que no deseaban transmitir, sólo transmite mensajes hacia la computadora o la terminal que deseaba hacerlo. Por lo tanto la computadora no pierde tiempo haciendo preguntas inútiles. De esta forma el usuarios debe esperar menos para que la computadora le atienda sus pedidos.

Este sistema presenta los siguientes inconvenientes:

Cada terminal debe saber la dirección del siguiente. Los problemas se generan cuando una terminal de la red deja de funcionar, esto produce que se interrumpa el ciclo de consultas, y la computadora no puede determinar cuál fue la terminal que lo interrumpió. Se

puede apreciar que en el sistema de consulta anterior no pueden ocurrir este tipo de problemas.

Transmitir en cuanto se quiera.

Como su nombre lo indica permite que se transmita en cuanto se quiera. Un problema típico es que dos o más nodos transmitan al mismo tiempo (transmisión simultánea) lo que provoca una colisión, pero como vamos a ver esto se resuelve mediante protocolos de recuperación.

Un ejemplo clásico de este tipo de solución es la red de Hawai ALOHA. En Hawai se deseaba instalar una red de computación, pero no existían líneas de comunicación para conectar los nodos de la red por problemas debido a la geografía del lugar. Como sabemos, Hawai es una región insular y las islas no tienen cables telefónicos que las comuniquen. Para solucionar esta dificultad se debió utilizar un sistema de transmisión mediante ondas de radio (comunicación entre líneas).

Los inconvenientes que acarrea esta solución son la aparición de colisiones y los códigos de redundancia. A veces el transmisor es capaz de detectar y detener la transmisión. Los transmisores deben ser informados de la colisión (explícitamente o por no recibir respuesta). Para evitar una nueva colisión, se introduce una demora artificial en los nodos. Esta demora puede ser de dos tipos :

- Que cada terminal espere un lapso fijo de tiempo, pero distinto para cada uno de ellos.
- Que cada terminal espere un lapso de tiempo aleatorio en cierto rango.

En consecuencia, una mejora importante es escuchar antes de transmitir. Si hay una transmisión se espera antes de que ésta termine.

Los bloques enviados desde la terminal hacia la computadora contienen códigos de redundancia. Si hay colisión, la computadora recibe un bloque deteriorado. Todo bloque

correcto debe ser confirmado como tal por la computadora mediante un mensaje: si hay problemas con un bloque de la computadora no envía mensajes de confirmación. Las terminales retransmiten si transcurrido cierto lapso de tiempo no reciben mensajes de confirmación; el tiempo de espera es aleatorio en un cierto rango. Si la frecuencia de enviar mensajes aumenta es probable que aumenten también las colisiones. Para que se comporte en forma aceptable, este sistema no debe exceder el 20 por ciento de la capacidad total de la línea de transmisión.

I.3 Desarrollo de sistemas informáticos.

La labor del diseñador de sistemas es altamente creativa. El diseño de un nuevo sistema y por tanto de la planificación de importantes cambios dentro de la organización, es competencia exclusiva del diseñador de sistemas.

Puesto que un funcionamiento eficaz es imprescindible para la vida de cualquier empresa, gran parte de su esfuerzo se invierte en labores de diseño. Se persigue alcanzar sistemas óptimos para el tratamiento de datos mejora de los existentes y buenos cauces para suministrar la información.

Evidentemente la tarea de un diseñador no puede aislarse; debe trabajar en estrecha relación con el resto de los miembros de la organización.

Su trabajo abarca muchas y muy diferentes etapas. Comienza con la identificación clara de los objetivos y una visión global primaria del nuevo sistema. De ahí surgen varios proyectos que se someten a profundas revisiones y, tras ser modificados adecuadamente, se escoge una versión.

Ciclo de desarrollo: Cualquier sistema de información basado en el uso de computadoras tiene un ciclo de vida característico.

La necesidad de crear un sistema nuevo surge siempre del deseo de mejorar el *procesamiento de la información.*

Se empieza por una versión preliminar con el fin de decidir si es posible o no desarrollar un sistema que resuelva los problemas detectados. En caso afirmativo, se acometerá la siguiente etapa; un estudio de viabilidad más concreto, con base al cual se decide continuar o abandonar el diseño propuesto.

Cuando el proceso continúa debe elegirse una de las alternativas con el fin de proceder a su desarrollo.

Se inicia entonces una fase de estudio detallado de los procedimientos existentes en la que se deben definir en detalle las partes afectadas o responsables de las dificultades descubiertas.

Se llega a continuación al punto más creativo: el diseño de un nuevo sistema.

La fase de especificación que sigue es muy laboriosa y debe desarrollarse con detalle.

- Descripción de las funciones fundamentales a ejecutar por el sistema (relaciones entre ellas y con el resto de los componentes de la organización).
- Descripción exacta de registros y elementos de datos (magnitud, tipo, validación, etc.)
- Selección de dispositivos de entrada y salida.
- Requisitos de programación.

En la etapa de programación se escriben los procedimientos que han de seguir los operadores y las máquinas. Este trabajo es desarrollado por programadores o por analistas. Deben someterse estos programas a una minuciosa etapa de pruebas.

Ya que el desarrollo de un nuevo sistema supone un cambio sustancial respecto al ya existente, su éxito posterior depende en gran medida de una buena formación de los usuarios.

Terminada la fase de instalación se inicia la denominada fase operacional, básicamente rutinaria. Pero esta calificación no significa en ninguna medida que no esté sujeta a cambios. Por el contrario, existe aquí una continua necesidad de mantenimiento, modificaciones y ampliación.

Mantenimiento porque es inevitable que los programas tengan errores. Además, el uso cotidiano hará surgir mejoras que le hagan ganar en eficiencia y lo ajusten de forma creciente a la realidad.

I.3.1 Revisión Preliminar.

Se analizan conjuntamente los procedimientos actuales a modificar o sustituir. Para el entendimiento entre usuarios y diseñadores se realizan diagramas de flujo, organigramas y tablas de decisiones.

I.3.2 Análisis del sistema.

Hoy en día, la idea más generalizada respecto a la metodología de análisis de sistemas sugiere que, en el trabajo de un analista el que toma la decisión de incluir o no en su desarrollo la sugerencias propuestas.

Sea cual fuere el motivo que impulse el desarrollo de un sistema nuevo, siempre habrá presentes dos circunstancias: los procedimientos con que se cuenta no son todo lo satisfactorios que se precisa o no serán los suficientes para responder a las demandas previstas.

Es fundamental una buena relación de cooperación entre diseñadores y usuarios.

Etapas de modelo general de cambio de una organización:

- Exploración
- Principio
- Diagnosis
- Planeamiento
- Acción
- Evaluación
- Finalización

La etapa inicial de exploración no presupone compromiso alguno entre analista y usuario. Cada uno de ellos desde su campo, estudia los motivos de iniciar el análisis.

El segundo paso a considerar lleva a un acuerdo informal entre ambas partes que debe basarse en un acuerdo inicial sobre los objetivos a alcanzar.

Durante la diagnosis, el papel del usuario es ayudar a definir en detalle el sistema. Algunos momentos de la etapa de diagnosis deben transcurrir en paralelo con el estudio de viabilidad.

Se deben delimitar claramente:

- La naturaleza del problema procesado
- Las partes de la organización afectadas por los cambios y los usuarios
- Los recursos con los que se puede contar
- La disposición de la organización frente a los campos necesarios

La etapa de planeamiento supone el desarrollo de detalles técnicos del sistema. Pero el desarrollo real del proyecto tiene lugar en la etapa de acción. En esta etapa, los usuarios comienzan a utilizar nuevos formularios y equipos. Su sistema de trabajo empieza a modificarse. Durante la etapa de pruebas el usuario utiliza por primera vez el nuevo sistema.

Un final con éxito será aquel en el que se comprueba que el usuario ha asumido el nuevo sistema y se consigue transferir la posesión del diseñador al usuario.

I.3.3 Estudio de viabilidad.

Los diseñadores analizan los costos y los usuarios participan en la evaluación de los beneficios.

Esta es la primera etapa donde se realiza un contacto entre analistas y usuarios; es una etapa fundamental para el logro de los objetivos finales.

El analista estudia en la revisión preliminar lo que se espera del nuevo sistema basándose en los conocimientos proporcionados por el usuario. A partir de aquí se desarrollan varias alternativas que podrían encajar como modelo solicitado.

Es muy importante que el analista ayude a escoger entre todas las alternativas y no cree en el usuario la necesidad de aceptar el sistema propuesto en su conjunto o abandonarlo y continuar con el existente.

¿Qué diferencia hay entre revisión preliminar y estudio de viabilidad? Simplemente el grado de detalle que alcanza en cada una de ellas.

Una vez conocidas globalmente las necesidades a cubrir, la revisión preliminar permite confeccionar una serie de sistemas que podrían satisfacerlas.

Posteriormente el estudio de viabilidad abordará el suficiente número de detalles como para que se pueda elegir una *única* alternativa en cuyo desarrollo se centrará el resto del proceso.

Ahora analizaremos someramente los puntos que deben formar parte de la revisión previa y de del estudio de viabilidad. En ambas etapas habrá que tener en cuenta el sistema actual y la nueva alternativa.

Siempre se realiza un estudio del riesgo que suponga esta nueva alternativa; en la mayoría de los casos habrá organizaciones que no desearán aparecer como abanderadas de sistemas innovadores.

El análisis de viabilidad operacional debe centrarse en si será o no posible operar con el nuevo sistema: El ajuste de los programas a la realidad, la posibilidad de reunir o no los datos precisos, etc.

En el estudio de la relación costo/beneficio, los costos a considerar serán los de desarrollo (tiempos de procesador, analista, programador, usuario) y de operaciones (costo de equipos de cómputo, comunicaciones, mantenimiento, etc.)

Como es de suponer no todos los ahorros son fácilmente cuantificables. Citamos algunos de ellos:

1. Posibilidad a partir de la innovación de obtener más información.
2. Recepción de la información más oportuna.
3. Mejoras operacionales.
4. Capacidad de cálculo superior a la existente.
5. Reducción de la burocracia.
6. Mantenimiento de una posición competitiva.
7. Mayor eficiencia en la toma de decisiones.
8. Mejor imagen de la empresa, servicio al cliente, etc.

I.3.4 La evaluación y adquisición de Hardware y Software.

En el proceso de selección de hardware y software pueden darse varias situaciones diferentes:

- La organización no dispone aún de hardware ni software y es necesario elegir el equipo completo.
- La organización dispone de Hardware pero está en estudio la sustitución por otro equipo.
- La organización tiene el hardware y es necesaria la adaptación del sistema a éste.

La selección de hardware y software es un punto crucial en el análisis y creación de sistemas ya que el tipo y características del equipo influyen notablemente en el desarrollo y diseño del nuevo sistema.

La conveniencia de un equipo dado solo puede determinarse en el contexto de una aplicación determinada, diseñada para resolver alguna necesidad concreta.

Durante el desarrollo de un sistema nuevo la evaluación de necesidades se lleva a cabo en varias etapas.

En las primeras etapas del diseño se realiza una preparación de estas necesidades. En esta fase se trata de especificar una configuración de la computadora adecuada para poner a punto el nuevo sistema. Todavía no se dispone de los detalles necesarios para la valoración de todas las posibilidades.

Enfocaremos el problema, en esta primera fase, suponiendo que se debe seleccionar un nuevo hardware. Como durante las primeras etapas del diseño no están especificados aún los procesos, la conveniencia de las distintas configuraciones puede y debe fijarse en función de necesidades generales.

Habrá que estudiar la configuración de una computadora basándonos en el diseño de mayor amplitud dentro de una serie, haciendo hincapié especial en la flexibilidad y compatibilidad del equipo. Esto significa estudiar características generales como:

UNIDAD CENTRAL:

RAM

- Capacidades posibles.
- Organización
- Tipo de protección
- Bits por acceso

Equipo central:

- Organización del direccionamiento.
- Registros de índice
- Disposición de forma flotante
- Tipos de interrupciones
- Tiempos de ejecución y representación de instrucciones

Canales:

- Tipos de canales disponibles
- Velocidad máxima promedio del tiempo total del proceso
- Medida en que los canales del procesador son independientes
- Número de canales que pueden conectarse a un equipo central
- Número de unidades de control que pueden conectarse por canal

UNIDADES DE CONTROL DE ENTRADA-SALIDA:

Impresora:

- Líneas por minuto
- Caracteres por línea
- Conjuntos de caracteres
- Máximo número de copias
- Formato de formularios diseñados
- Espaciados de líneas

Consola:

- Teclado
- Velocidad de escritura
- Posibilidades de consulta
- Representaciones visuales

Terminal:

- Teclado numérico o alfanumérico

- E/S de tarjetas
- E/S de cinta magnética
- E/S de discos magnéticos
- Impresora
- Velocidad de transmisión
- Número de unidades de E/S por terminal
- Combinaciones de E/S
- Memorias intermedias
- Comprobación y corrección de errores

Explorador óptico:

- Resolución
- Tamaño de documentos
- Velocidad
- Método de exploración
- Posibilidades
- De uso remoto
- En línea

Trazadores:

- Velocidad horizontal y vertical.
- Movimiento bidireccional
- Tamaño de papel y pasos
- Control en línea y fuera de línea
- Posibilidad de uso remoto

Manejo de Gráficos:

- Memoria intermedia
- Número de caracteres por línea
- Número de líneas
- Velocidad
- Características de E/S
- Posibilidad de uso remoto
- Pantalla de color

MEMORIAS EXTERNAS:

Unidades de cintas magnéticas:

- Densidad de caracteres por pulgada
- Velocidad de transferencia
- Tiempos de acceso: Máximo , mínimo y promedio

- Número máximo de unidades que se pueden conectar a la misma unidad de control.
- Número de mecanismos de acceso de independiente disponibles por unidad

En cuanto al software, habría que estudiar los niveles disponibles y sus principales especificaciones para cada modelo. Es importante tener información sobre los programas de apoyo que se disponen y de los lenguajes existentes.

Resultará de gran ayuda saber si hay utilerías de clasificación, intercalado, clasificación, programas de transcripción de datos y entrada de datos.

Esta primera preparación de necesidades es importante y debe hacerse aún encontrándonos en etapas muy previas del ciclo de vida del nuevo sistema.

Una vez terminado el desarrollo del sistema es hora de proporcionar los equipamientos de hardware y software. En tramos ahora en una etapa de revisión. Puesto que el primer estudio se hizo considerando el sistema en su conjunto, serán pocos los cambios que durante la revisión haya que hacer. De cualquier manera habrá que hacerla para prever la posibilidad de que fuese imposible ejecutar un buen desarrollo de el sistema o alguna de sus partes integrantes. Si esta posibilidad se diera, habría que decidir si las especificaciones previas de hardware y/o software deberían o no ser alteradas, considerando siempre el impacto que podría originarse sobre el resto del sistema.

El mayor problema surge cuando es necesario introducir cambios en el software. Estos cambios llevan aparejados dificultades de reconocimiento y descripción:

Cada archivo podría definirse de la forma siguiente:

- Número de archivo: cada archivo se asocia con un número
- Número de registro lógico por archivo
- Número de campos por registro
- Número de registros por archivo
- Número de caracteres por registro
- Categoría del archivo
- Soporte del archivo
- Nombre del archivo
- Etiquetas

En cuanto a la definición de sistema debe especificarse el diseño del mismo y sus necesidades de proceso.

Cada ejecución de la computadora se define en términos de:

- Identidad de proceso
- Identidad de archivo y parámetros totales que manejan
- Especificaciones de las necesidades de proceso interno estableciendo las categorías siguientes:

Operaciones de tratamiento de datos (actualización, calculo de sumas y restas, validación, cálculo de productos y divisiones...)

- Operaciones matemáticas.
- Operaciones de matrices y tablas.
- Operaciones de básicas de cómputo.
- Operaciones de programación uniformes.

I.3.5 Desarrollo del sistema (programación).

La programación es el arte y la ciencia de "conducir" al procesador. Esto es, disponemos de unas herramientas y proceso y debemos de decidir cuáles se han de utilizar y en que orden para conseguir los resultados deseados con el menor número de recursos (de todo tipo) posibles.

Un programa es una secuencia de procesos descritos de una manera inteligible para una computadora, que contiene las instrucciones de las operaciones a realizar por la computadora para alcanzar la solución del problema propuesto. Cada instrucción es una orden que se le da al procesador para que realice una operación simple. Las operaciones complejas no son más que combinaciones de las más simples. Los programas escritos en las computadoras, se denominan *software*.

Consideramos dos etapas fundamentales en programación que son:

1. Determinar la secuencia de procesos necesarios para conseguir el objetivo.
2. Escribir esa secuencia de forma que la computadora pueda entenderlo.

La primera etapa corresponde con el análisis y diseño del problema y la segunda con la codificación de la solución a un lenguaje de programación.

La necesidad de comunicarse con las computadoras en un lenguaje más preciso que el humano ha dado lugar a los diversos lenguajes de programación, es pues, un medio de comunicación entre el programador y la computadora.

El único lenguaje que la computadora es capaz de interpretar es el de *máquina*.

Hacer programas en lenguaje de máquina es muy complicado y por lo cual se diseñaron los lenguajes de programación de los cuales podemos mencionar dos grandes grupos:

De alto nivel

De bajo nivel

Los lenguajes de alto nivel se caracterizan porque el vocabulario utilizado es mucho más cercano al lenguaje humano que al de máquina, mientras que los de bajo nivel, denominados generalmente, *ensambladores*, son mucho más similares al lenguaje de máquina.

Con un lenguaje de alto nivel se gana tiempo de programación pero se pierde tiempo de ejecución.

La codificación de un programa a un lenguaje es la ultima fase a efectuar cuando se esta en proceso de programación. Las fases previas, que se describen a continuación las consideramos indispensables:

- Puntualizar objetivos
- Decisión del formato de entradas y salidas
- Especificación de los requisitos lógicos
- Selección del lenguaje de programación
- Concepción de la lógica del programa
- Codificación

Una vez codificado un programa, todavía quedan algunas fases antes de su instalación definitiva:

- La compilación
- Pruebas y puesta en servicio

Puntualizar objetivos. Los objetivos los fijan los analistas junto con los usuarios de los programas. Frecuentemente, esta etapa de análisis es larga y decisiva, Si los objetivos se puntualizan con exactitud, el programa a desarrollar no creará mayores problemas en fases posteriores.

Formato de las entradas y salidas. Antes de escribir un programa hay que decidir qué datos debe contener y cómo resolver la cuestión de las entradas y las salidas.

Especificaciones de los requisitos lógicos. Una vez claros los objetivos a conseguir, hay que definir cómo conseguirlos. Hay que elegir o diseñar el método de trabajo; en definitiva hay que elegir un algoritmo.

Selección del lenguaje de programación. Cualquier nuevo lenguaje de programación se diseña para resolver una cuestión concreta o agilizar algún procedimiento. El analista decidirá que lenguaje se adapta mejor a la aplicación a realizar. Es necesario evaluar las características, ventajas e inconvenientes que poseen cada uno de ellos y escoger el óptimo de acuerdo a las circunstancias específicas del caso.

Concepción de la lógica del programa. Una etapa muy importante, previa a la codificación, es la estructuración del programa.

La idea básica de la estructuración es la división. El programador escribe cada parte por separado. Cuando éstas funcionan correctamente, también lo hará el programa completo si las conexiones se realizan adecuadamente. Cada programa es, pues, un procedimiento principal que consta de módulos.

El objetivo básico de la programación estructurada es reducir el riesgo de introducción de errores y facilitar su aislamiento y corrección cuando éstos aparecen.

Codificación: Es el hecho de introducir a la computadora el código para crear el programa fuente.

Compilación: Para que una computadora pueda ejecutar un programa, éste debe figurar en código de máquina. La misión de la etapa de compilación es convertir el programa fuente en objeto. Como consecuencia de la compilación se pondrán de manifiesto, si existieran algunos errores.

Prueba y puesta en servicio: Los errores sintáctico o los que producen por un mal uso del lenguaje se detectan como consecuencia de la compilación, y son fáciles de corregir.

El programador debe de ocuparse de depurar el programa por si existieran errores en la lógica, casi siempre poco obvios. Algunos errores lógicos quizás aparezcan hasta el uso.

I.3.6 Documentación.

Con el término documentación hacemos referencia a todo lo que se escribe acerca de un sistema de información. Sirve para diversos fines:

En la etapa de diseño es el producto desarrollado por diseñadores y usuarios.

Tras la instalación sirve como base para realizar modificaciones al sistema.

La calidad de la documentación está en relación directa con la capacidad que el departamento de cómputo tiene para responder eficazmente a las sugerencias de los usuarios.

Se puede hablar de cuatro grandes grupos de documentación:

En primer lugar la *documentación de diseño* que surge como resultado del trabajo del equipo de diseño. Es parte del proceso de diseño en sí, y facilita la comunicación entre los componentes del equipo. Esta documentación cumple, también, una labor de control: registra todo el proceso de desarrollo de cambios efectuados.

Disponer de la documentación de diseño supondrá, en un futuro, la posibilidad de estimación de la duración del desarrollo de un programa semejante al ya diseñado.

La sección más importante de la documentación de diseños es quizás el índice de asuntos contenidos en los documentos. Éste índice (que debe ser jerárquico) sirve como directorio de los archivos manuales que contienen toda la información que describe el sistema.

La documentación de diseño debe incluir los estudios de revisión y viabilidad y las especificaciones detalladas del nuevo sistema. Otros de sus componentes principales deben ser los programas actuales. Es fundamental mantener copias de entrada, salida, formatos de registro y lista de módulos.

En síntesis debe contener lo siguiente:

- Índice
- Revisión
- Estudio de viabilidad
- Documentación de sistemas ya existentes
- Especificaciones del nuevo sistema: Salida y entrada, versiones del archivo, lógica, módulos de programas, programas actuales, pruebas.
- Procedimientos manuales
- Planes de trabajo
- Informes de progresos

Documentación para el aprendizaje del usuario, se prepara para la puesta en servicio del sistema. Casi toda la información válida para la enseñanza puede extraerse de la documentación de diseño vista anteriormente.

La misión de la documentación del usuario es salvar la diferencia entre el antiguo sistema y el nuevo.

La práctica ha demostrado que resulta más positivo empezar informando sobre la salida del sistema o producto final del mismo. A continuación se informa sobre la entrada y los registros necesarios para lograr esa salida. Por último se estudia la lógica del proceso de la computadora.

Muy importante resulta informar en la documentación sobre las condiciones de error y las acciones a tomar frente a ellos.

El aprendizaje de los usuarios no resultará difícil si el diseño se hizo contando con su participación. En estos casos la enseñanza debe ser organizada por los usuarios que participan en el desarrollo del sistema.

Documentación de operaciones. El departamento de informática, para manejar el sistema una vez puesto en servicio, necesita la información sobre procedimientos de operación y formas de actuación frente a los errores.

La mayor parte de la documentación de operaciones se obtiene de la documentación de diseño.

Documentación para referencia del usuario. Por último debe producirse una documentación para uso del usuario, después de que el sistema ya esté en servicio.

Todo usuario que tenga una duda consultará la documentación que si es de calidad suficiente evitará consultas al departamento de informática.

La mayor parte del material para la documentación de referencia puede obtenerse de la documentación del usuario para su aprendizaje. Su contenido mínimo debe ser :

- Índice
- Documento de entrada
- Informes de salida
- Lógica del proceso
- Condiciones de error
- Procedimientos manuales
- Personal de programación y mantenimiento
- Representantes del usuario

Uno de los componentes más importantes de esta documentación es una lista de condiciones error y corrección.

I.3.7 Pruebas y puesta en servicio.

Las fases finales del desarrollo de un sistema incluyen pruebas y la instalación. Si a lo largo del desarrollo la participación del usuario ha sido la adecuada, esta fase resultará sencilla y eficaz.

Las pruebas incluyen la verificación de cada programa y del buen funcionamiento del sistema. Es obvio que este tipo de pruebas difícilmente serán exhaustivas y que son muchos los caminos posibles. Analizaremos en este apartado los caminos más frecuentemente empleados. Por supuesto siempre hay que trabajar con la idea de que ningún programa está libre de errores por completo. Es aquí donde comprobaremos la importancia de que en el desarrollo del sistema se haya tenido en consideración la necesidad del control de errores.

A lo largo del proceso de diseño del sistema se hacen pruebas de los módulos básicos del programa. Estas pruebas las llevan a cabo los programadores, quienes construyen sus propios datos de pruebas o utilizan un programa generador de tales datos.

Una vez probados los módulos individualmente se realizan las pruebas de módulos combinados, pruebas relacionadas con los programas mismos.

Simultáneamente con el desarrollo de los programas tiene lugar una actividad importante: El diseño y comprobación de los problemas manuales (no debe olvidarse que los procedimientos manuales pueden determinar el éxito de un sistema).

En muchos casos se combina el nuevo sistema con el uso paralelo del antiguo método de proceso. De esta manera los resultados de los sistemas se comparan como una prueba del sistema nuevo antes de desechar el viejo. Estas pruebas en paralelo son muy ventajosas a pesar de la incomodidad adicional que supone para el al usuario trabajar simultáneamente con los dos sistemas.

En ocasiones se tiende a una conversión gradual al nuevo sistema realizando sustituciones por departamentos, regiones geográficas o cualquier otro método viable de división.

I.4 Arquitectura Cliente/Servidor.

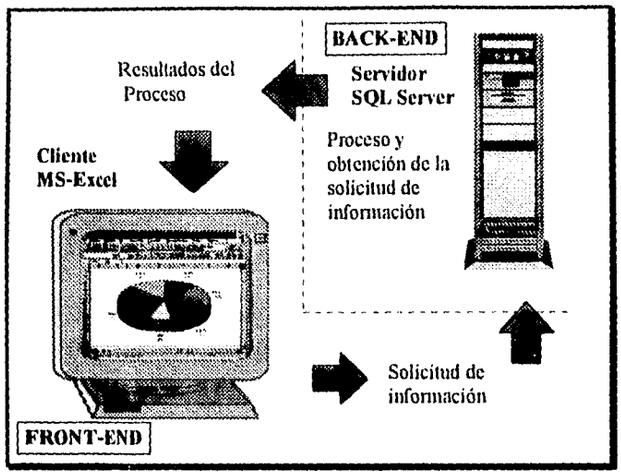
Inspirada en lo mejor de los mainframes y aplicando las ventajas de las redes de área local, se ha desarrollado la arquitectura Cliente/Servidor basada en un principio relativamente simple, pero sumamente poderoso de ejecutar las aplicaciones tomando la seguridad y eficiencia de un mainframe mientras se toman las ventajas y la simplicidad del poder de las estaciones de trabajo en PC.

Esencialmente la arquitectura Cliente/Servidor parte en dos una aplicación y asigna el proceso de cada parte de la misma a los mejores recursos situados a su alcance para su manejo.

El componente "front-end"[1] puede ser diseño propio o bien, comercialmente disponible en aplicaciones para PC. La presentación y manipulación de los datos de la estación de trabajo la realiza la propia PC. El manejo de "back-end"[2] del almacenamiento de la información, atributos, derechos y protección de los datos, como los realizan como en un mainframe, pero generalmente residen en un poderoso servidor tipo PC. Las conexiones tipo LAN (Local Area Network), los componentes del cliente y los del servidor son los que definen el término "Cliente/Servidor".

Un elemento clave de la tecnología Cliente/Servidor es el servidor inteligente. En el pasado los servidores han sido dispositivos muy caros de almacenamiento de datos. Con las maquinas disponibles en la actualidad, es posible adicionarle recursos a la red, mejorando el proceso de tareas, control de acceso a la información y manejo de periféricos como en minicomputadoras y mainframes.

[1] Cliente (front-end): Computadora personal en la que se ejecuta una aplicación de Presentación y Manipulación de Datos al Usuario.
[2] Servidor (back-end): Equipo mini o macro en que se lleva a cabo el Almacenamiento, la Recuperación y la Protección de los Datos.



La evolución de las estaciones de trabajo inteligentes (IWS), el tremendo desarrollo del poder de proceso y el tremendo apogeo de los modos gráficos están cambiando la dirección de los sistemas de cómputo de hoy. Una de las principales causas es la implementación y el desarrollo más rápido de aplicaciones basadas en estaciones de trabajo. Otra causa es el incremento en el costo-beneficio en el desarrollo de aplicaciones para un usuario o grupos de usuarios. Esos grupos pueden compartir recursos tales como información, dispositivos, servicios y aplicaciones.

Compartir recursos donde se optimiza la cooperatividad así como la interacción entre *clientes* (requiriendo recursos) y *servidores* (proporcionando recursos) puede ser alcanzada empleando un modelo de computación *CLIENTE/SERVIDOR.*

El modelo de computación Cliente/Servidor representa una instancia específica del procesamiento distribuido cooperativo, donde la relación entre los Clientes y los Servidores es un conjunto de componentes de *hardware* y *software*. La arquitectura Cliente/Servidor son componentes de hardware y software y su interrelación es el objetivo de éste capítulo.

I.4.1 Introducción a los sistemas distribuidos abiertos y al modelo Cliente/Servidor.

Actualmente es muy común escuchar que la tecnología Cliente/Servidor es el camino de los noventas. Se puede afirmar que reduce los costos de mantenimiento de software incrementa la portabilidad, mejora el desempeño dentro de las redes y elimina la necesidad de mainframes y minicomputadoras. Sin embargo es necesario revisar de que manera puede lograrse esto con base en los modelos de la evolución de los sistemas de cómputo, para poder tener elementos y ratificar la anterior aseveración.

Proceso basado en Host (Amo).

El modelo de computación Cliente/Servidor implica un proceso cooperativo de peticiones solicitadas por un cliente que depende de un servidor, el cual, procesa las requisiciones y devuelve los resultados al cliente.

Este proceso cooperativo es realmente una forma especial de proceso distribuido, pero el más elemental sistema de proceso distribuido es el del modelo de *tiempo compartido* utilizado por sistemas de minicomputadoras y mainframes. Las aplicaciones y datos son almacenados en una computadora central llamada host ("amo"). El host proporcionaba la inteligencia para ejecutar las aplicaciones y procesar los datos. Las terminales no inteligentes son utilizadas para accesar al host, este procesa la información y la almacena. Este modelo tuvo ventajas definitivamente: El poder de proceso del host proporciona una alta respuesta; los datos se almacenan y controlan en un punto centralizado, así se protege la integridad; además de que los mainframes pueden utilizar programas sofisticados y procedimientos que aseguran una gran confiabilidad.

El tradicional modelo de computación basada en *Host* (amo) está disminuyendo en ciertas áreas también: esas grandes y poderosas máquinas con complicadas interfaces, son sumamente difíciles de operar; cuando la empresa requiere de incrementar su sistema de cómputo necesita invertir grandes cantidades de capital para aumentar el poder de proceso;

para adicionar o actualizar aplicaciones usualmente se requiere de mucho tiempo y un nuevo código. Desde el punto de vista de proceso, el ambiente Host es totalmente no-distribuido.

Proceso Maestro-Esclavo

En respuesta de esas necesidades, la primera generación de LAN's fueron desarrolladas con base en el modelo de *recursos-compartidos*. A diferencia de un Host que hacía todo el proceso, éste modelo utilizaba estaciones de trabajo inteligentes para ejecutar las aplicaciones. El servidor únicamente permitía a los usuarios de la red compartir archivos y periféricos.

El modelo de *recursos-compartidos* asimismo, se ha fortalecido: La familiaridad, la facilidad de utilizar PC's como servidores de las estaciones de trabajo; el costo de una LAN es considerablemente más bajo que el de un sistema con mainframes o minicomputadoras; y porque el poder de proceso está en la estación de trabajo, es posible adicionar fácilmente éste aumentándolas.

Proceso Cliente/Servidor

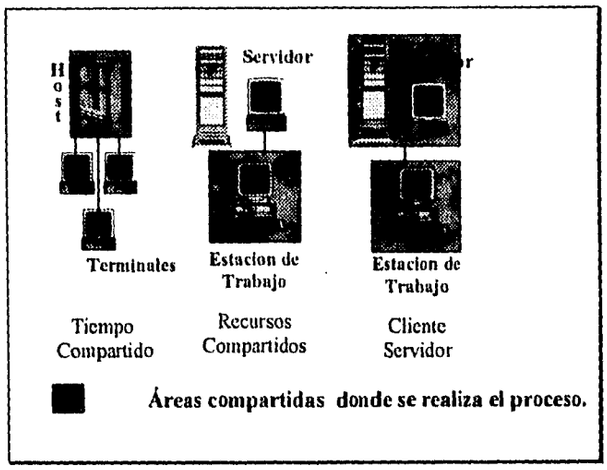
El modelo de proceso Cliente/Servidor ha emergido como el más alto nivel de compartición de recursos típicamente encontrados en una red de área local (Local Area Network). En un ambiente de proceso compartido tipo LAN, las computadoras personales son conectadas a un sistema de dispositivos que pertenecen a ese conjunto como recursos comunes; un ejemplo típico es un archivo en un disco duro, una impresora, una unidad de respaldo, etc. En la terminología LAN, en la cual los dispositivos compartidos son llamados *SERVIDORES*. El nombre servidor es apropiado, ya que son las máquinas que se utilizan para recibir los requerimientos de servicio de las demás PC's, que generalmente tienen recursos más limitados. El modelo de proceso Cliente/Servidor es una extensión natural del proceso de recursos compartidos. Un ejemplo de este tipo de proceso dentro de una red de área local son el NetWare de Novell y el LAN Manager de Microsoft. Como en una red de área local era mayor el número de estaciones de trabajo el desarrollo se enfocó a la compartición de archivos y de impresoras, gradualmente se convirtieron en los actuales

servidores que le daban servicio a un gran número de estaciones de trabajo ya que la evolución permitió aumentar el poder y capacidad de proceso.

A su vez, el rol de las estaciones de trabajo fue cambiando convirtiéndose en clientes de los servidores . La principal razón para éste cambio fue que a lo largo del ambiente de compartición de los servicios de impresión y archivos entre las estaciones de trabajo en un grupo de la LAN representaba solamente una fracción de una aplicación específica. La parte significativa de la aplicación que proporcionaba funcionalidad a este grupo fue la mejor candidata para que se compartiera dentro de los usuarios que la requerían dentro de la red completa. Así pues, parte del proceso de la aplicación fue distribuido a un nuevo servidor - éste recibía la petición del proceso especial dentro de la aplicación y devolvía el resultado a sus *clientes.*

En este modelo, el proceso de la aplicación es dividida (no necesariamente siempre) entre el cliente y el servidor. El proceso es actualmente iniciado y parcialmente controlado por quien lo requiere -*cliente*- pero no en una modalidad maestro-esclavo. Aquí, en cambio, un conjunto de clientes y servidores cooperan para ejecutar satisfactoriamente una aplicación completa. Un manejador de base de datos, tal como Sybase o SQL Server de Microsoft son ejemplos de ambiente de proceso Cliente/Servidor.

Los sistemas de red actuales, y disponibles a nivel comercial, soportan el desarrollo de aplicaciones Cliente/Servidor realizando una comunicación interproceso (InterProcess Comunication) fácilmente, que permite al cliente y al servidor coordinarse y comunicarse dentro de su red. Puede utilizarse el NetBIOS (Network Control Blocks o control de bloques de red), correo electrónico, o los nombrados pipes (tuberías) para transmitir la información entre dos o más computadoras



- **Tiempo Compartido**: Aplicaciones y datos almacenados en una máquina central. Tiene terminales tontas para accesar al host.

- **Recursos Compartidos**: Usa la inteligencia de una Estación de Trabajo para correr aplicaciones y el servidor comparte archivos y periféricos.

- **Cliente-Servidor**: Combina las ventajas de los modelos anteriores y minimiza sus desventajas. Trata como inteligentes tanto al Servidor como a las Estaciones de Trabajo y los explota al máximo.

Sistemas de Archivos distribuidos.

Los sistemas de archivos distribuidos operan para permitir que un usuario en un sistema que está conectado a una red accese y modifique datos almacenados en archivos de otro sistema. Desde el punto de vista de la arquitectura Cliente/Servidor, los sistemas en los cuales el usuario está trabajando son del *cliente*, mientras que los sistemas en los cuales están almacenados los datos son el *servidor*.

Cuando el dato es seleccionado en el servidor de archivos, una copia de ellos es colocada en memoria *caché* en el sistema del cliente, así pues, el cliente puede leerlos y modificarlos. Cuando se han modificado son devueltos y reescritos en el servidor.

El ambiente de computación distribuida y los sistemas de archivos distribuidos proporcionan los siguientes avances y rasgos distintivos dentro de una arquitectura Cliente/Servidor:

•*Protección y seguridad de acceso.* Generalmente UNIX proporciona un sólido sistema de seguridad pero además de eso se han creado procedimientos de autenticidad de acceso, encriptamiento y mecanismos de acceso controlado a clientes.

•*Consistencia de la información.* Anteriormente cuando en un sistema distribuido se llevaba a cabo un proceso se requería de un servidor que no se encontraba disponible éste no se consumaba y provocaba que la información estuviera incompleta o desequilibrada. Este problema se ha resuelto haciendo una réplica de los servicios de red en uno de los clientes, que en el momento en que uno de los servidores se encuentra indispuesto se hace un cambio inmediatamente asignándole sus funciones a otra máquina.

•*Disponibilidad de la información.* Los administradores actuales de red han mejorado las ventajas que presentaban los sistemas de archivos distribuidos anteriores, permitiendo por ejemplo, realizar copias de respaldo sin necesidad de bloquear los archivos o dar de baja algún servidor.

•*Desempeño* Se han mejorado los procedimientos del manejo de caché haciendo mas eficientes las comunicaciones y disminuyendo las sobrecargas en la red.

•*Estándares* En lo general la arquitectura cumple con la semántica del sistema de archivos del IEEE POSIX.

Los sistemas de computación distribuida en arquitectura Cliente/Servidor representa una valiosa contribución al desarrollo de los sistemas distribuidos abiertos.

1.4.2 ¿Qué es la arquitectura Cliente/Servidor?

En los últimos tres años, la mayor parte de los proveedores de tecnología de información y consultoría, han hecho de las tendencias de reducción (downsizing)[3] y de Cliente/Servidor, los elementos fundamentales de sus estrategias.

Pero ¿qué es la arquitectura Cliente/Servidor? Para decirlo en pocas palabras, en un ambiente computacional con Cliente/Servidor los empleados utilizan sus PC's, conocidas como clientes, para ejecutar software de productividad personal, como hojas de cálculo. Si necesitan datos sobre la empresa en que laboran, solicitan acceso al servidor. Podrían necesitar estos datos y la hoja de cálculo para analizar qué zonas tienen mayores ventas de la empresa.

El software de productividad se da en grupo tanto en servidores como en máquinas de escritorio. Un fabricante podría tener software especial para que los empleados que solicitan suministros tengan acceso a los sistemas de facturación e inventario que corren en el servidor.

Otro software de productividad en grupo hace posible que los empleados intercambien información y documentos. Por ejemplo, los compañeros de trabajo podrían obtener copias electrónicas de los análisis de ventas por regiones y un informe sobre inventario. Antes, el software para realizar labores relacionadas con la contabilidad, el presupuesto y la administración de proyectos se encontraba todo en un mainframe enorme y costoso; ahora esas mismas aplicaciones están a disposición de los usuarios de la LAN en una configuración Cliente/Servidor.

En la computación Cliente/Servidor una computadora poderosa actúa como la central de conexiones de un grupo de usuarios que necesitan trabajar conjuntamente. En el nivel más bajo de la arquitectura Cliente/Servidor podemos usar hasta una computadora Compaq

[3] Downsizing: Transferencia de aplicaciones en Mainframes a redes de PCs con servidores poderosos.

ProLiant en una red NetWare para que grupos de trabajo de hasta cincuenta usuarios compartan archivos e impresoras.

A diferencia del software para un solo usuario, el software para un ambiente Cliente/Servidor debe dar servicio tanto a personas en lo individual como en grupo. Asimismo, debe tener gráficas y características que simplifiquen su uso, como las que la gente espera encontrar en los programas para PC.

Los sistemas cliente/servidor son la más reciente tendencia en el desarrollo de sistemas avanzados de información interconectados por medio de una tecnología de red de área local. Como las redes LAN han encontrado una amplia aceptación dentro de la mayoría de las empresas de negocios los usuarios han encontrado una nueva y diferente manera de usar y compartir recursos simultáneamente.

Debido a la necesidad de mas eficientes maneras de compartir y accesar información, los sistemas cliente /servidor han parecido ser "una idea a la que le ha llegado su hora". Los sistemas cliente servidor alojan aplicaciones para almacenar y accesar datos en servidores de base de datos utilizando la tecnología LAN.

Con respecto al ambiente de oficinas inteligentes, el modelo cliente/servidor proporciona un número mayor de ventajas. Principalmente el hecho de que la tecnología Cliente/Servidor centraliza las aplicaciones para optimar el uso de hardware y de software. La arquitectura cliente servidor mejora los accesos a los datos dentro de una oficina que comparte sus recursos por medio de una red LAN y de esta manera más otras oficinas pueden accesar a los mismos datos utilizando el software de su propia PC.

En un modo genérico tienen la siguiente estructura:

1.- Consisten de nodos, los cuales, están interconectados a través de un medio continuo, tal como un cable.

2.- Es privada. El usuario se administra personalmente y no está sujeto a otras regulaciones.

3.- Se puede utilizar tanto en canales de comunicación de alta como de baja velocidad.

4.- Es comercialmente disponible en paquetes. No requiere de programación adicional.

5.- Se utiliza en forma de nodos dedicados, los cuales proporcionan servicio a otros usuarios.

Por lo anterior hoy en día cada vez más corporaciones utilizan sistemas de cómputo distribuidas que incluyen PC's compatibles con IBM, Macintoshes Apple, estaciones de trabajo UNIX, y otros tipos de sistemas de cómputo.

Como resultado, el usuario tiene la oportunidad de accesar la información de las bases de datos de uno o más sitios remotos. Es decir, que la responsabilidad de presentación y acceso de datos es dividida entre el cliente y el servidor. Un estación de trabajo cliente (generalmente una computadora personal) se enfoca a la presentación de los datos al usuario. La máquina servidor, usualmente un poderoso procesador con una vasta capacidad de almacenamiento, se concentra en el almacenamiento y recuperación de los datos.

Cuando el usuario requiere de la base de datos, la máquina cliente presenta una petición a la máquina servidor. La base de datos residente en el servidor acepta la petición, la compila y selecciona la mejor estrategia de optimización. Entonces el manejador de la base de datos colecta el resultado de la petición y transfiere el resultado a la máquina cliente.

Cada vez más sistemas de bases de datos proporcionan aplicaciones programadas para utilizar interfaces, la cual, es utilizada por las aplicaciones de la máquina cliente para accesar a la base de datos y que se encuentra en el servidor. Este tipo de protocolo incluye algunas acciones:

1.- Establece una sesión: Para establecer una sesión con el servidor, es necesario que el usuario proporcione su nombre de usuario y su password. El servidor coteja la validez de los datos y si éstos son correctos establece la sesión. A partir de este momento toda la comunicación entre el cliente y el servidor es manejada dentro del contexto de sesión.

2.- Prepara un petición (Query): El cliente transfiere una petición a la base de datos, tal como un SELECT, UPDATE o un DELETE al servidor. El sistema de base de datos del lado del servidor realiza los siguientes pasos:

- Compila la petición, revisando asimismo, que todos los campos y tablas sean válidos.

- Optimiza la petición compilada para determinar el método de obtención de los resultados más idóneo. El proceso de optimización toma ventaja de índices definidos en las tablas dentro de la base de datos y los reordena para mininizar el acceso a cada tabla.

3.- Acepta el resultado y procesa: Si se requiere de visualizar los datos desde el cliente, éste puede preguntar por cada una de las tuplas del resultado que procesó el servidor. De esta manera el cliente puede recibirlas una a una, o procesarlas individualmente.

Si el cliente requiere de esas tuplas individualmente, la base de datos del servidor no transfiere todas las tuplas a través de la conexión individual. En su lugar, para evitar un exceso de tráfico innecesario dentro de la red proporciona un caché al cliente. Cuando se requiere de un conjunto de resultados, se toman del caché y se le transfieren al cliente en un "bloque de resultados" a través de la red.

4.- Termina la sesión: Cuando un cliente ha procesado todas sus peticiones de la base de datos del servidor termina la sesión enviando un "fin de sesión".

Sin embargo, todas las ventajas señaladas anteriormente se ven entorpecidas por dos razones:

La primera es la falta de habilidad para manejarlas. Como una red LAN crece continuamente en tamaño y en funcionalidad, nuestra capacidad para poder controlarla se ve entorpecida. Los esfuerzos en el área de la estandarización de los protocolos se ve beneficiada, pero debemos continuar trabajando antes sobre eso en áreas donde el problema no se ha resuelto.

La segunda es la falta de aplicaciones diseñadas para las redes. El valor de la productividad que puede ser ganada por la utilización de tecnología LAN sin un software lo suficientemente bueno que lo respalde es limitada.

Las oficinas inteligentes son un intento para resolver estos problemas. De la total y correcta utilización de las redes, la compartición de ideas e información, la integración de las aplicaciones existentes dentro de un ambiente de red , las oficinas inteligentes pueden proporcionar un marco de trabajo mediante el cual seguiremos utilizando computadoras así dispuestas durante los años venideros.

I.4.3 Componentes de los sistemas Cliente/Servidor.

La arquitectura Cliente/Servidor permite a las organizaciones lograr la transición con resultados palpables a muy corto plazo. El objetivo de este modelo consiste en dividir las funciones de la aplicación en tres componentes básicos:

Presentación: Este componente se encarga de la interacción hombre-máquina a través del teclado, ratón, monitor (que puede ser touch-screen), o bien mediante reconocedores de voz o escritura.

Middleware (Servidores): Con el tremendo auge que ha tenido el llamado downsizing corporativo, las compañías enfrentan el dilema de lograr que las aplicaciones trabajen en un ambiente Cliente/Servidor heterogéneo, que cuente con protocolos de red, sistemas de operativos y bases de datos diferentes. La solución a este problema parece ser el llamado middleware, una capa que se encuentra entre las aplicaciones y los sistemas operativos, las bases de datos y los protocolos de red. El middleware hace que los sistemas operativos y los protocolos dispares sean invisibles para el desarrollador de aplicaciones y la aplicación resultante. El middleware es transparente para el usuario final, y está diseñado fundamentalmente para desarrolladores de aplicaciones y programadores. Son dos las áreas en que el middleware está teniendo una gran influencia. En primer lugar, en el área de redes, pues es posible que sistemas con protocolos distintos trabajen conjuntamente. En segundo lugar, gracias al middleware cualquier base de datos frontal es capaz de trabajar con cualquier base de datos de fondo. Compuesto por varios servidores o componentes de software localizados en una de las plataformas que se encargan de:

- Conectar a los sistemas existentes
- Conectar a las bases de datos
- Procesar y consultar información
- Interactuar con las interfaces del usuario
- Mantener la seguridad, la auditoría, y controlar las versiones.
- Proveer de mecanismos que permitan un nombrado independiente de los diferentes protocolos, sistemas operativos y equipos involucrados.

Información: En este componente se incluye la información, sistemas y aplicaciones existentes, que habrán de ser encapsulados para aprovechar las ventajas de esta arquitectura y minimizar los esfuerzos de programación transicional.

El modelo anterior, al separar la aplicación en tres componentes proporciona:

- Libertad para seleccionar cualquier manejador de base de datos (DBMS)
- Libertad para elegir la interface del usuario.
- Flexibilidad para integrar nuevas tecnologías cuando se requieran
- Acceso en línea a todo tipo de información
- Libertad para elegir las características del hardware
- Desarrollar en paralelo las aplicaciones
- Seguridad y tolerancia a fallas
- Conservar las versiones actuales de hardware y software
- Diversas posibilidades de visualizar la información

I.4.3.1 Interfaces Gráficas de Usuarios (GUI).

El medio lógico que se permite al usuario final comunicarse con las aplicaciones es llamado *Interfase Gráfica de Usuario* (o Graphical User Interfaces) y están diseñadas para presentar la información a los usuarios de una manera gráfica. Los gráficos no necesariamente significan dibujos, un procesador de texto o un escritorio de trabajo requiere de múltiples tipos de letras, tamaños, presentaciones y estilos, por tanto, requiere de un manejo gráfico de presentación.

El diseño de una interface gráfica de usuario tiene limitaciones de plataforma, por ejemplo, las diseñadas para Macintosh no son compatibles con Microsoft o con OS/2. Así pues, una GUI estándar con una aplicación sencilla y programada en un lenguaje común que permitiera la potabilidad tendría un impacto benéfico sumamente significativo en los desarrolladores de aplicaciones y en la productividad.

Requerimientos generales de una GUI:

•*Portabilidad.* Que le permita a los usuarios saltar de una plataforma a otra.

Que cumpla con los estándares. Es necesario que cumpla con los estándares comunes a los sistemas abiertos como el ANSI, IEEE, NIST (National Institute of Scince and Tecnogy), X Window System para servidores, etc. Estas especificaciones se encuentran listadas en el Interclient Communications Convention Manual (ICCCM) y son de importancia para permitir un alto grado de la interoperatividad de las aplicaciones.

•*Flexibilidad.* Un GUI estándar debe ser flexible para adaptarse a las distintas memorias, dispositivos de entrada/salida tanto actuales como a los futuros.

•*Internacionalización.* Hoy en día el mercado es global y la internacionalización es una especie de portabilidad incluyendo otros lenguajes, números, monedas, unidades, fechas, horas, etc.

•*Independencia de Plataformas.* Para ser verdaderamente abierta y estándar, una GUI debe ser diseñada para operar independientemente del sistema operativo o de la plataforma de hardware donde se ejecute. Similarmente en un ambiente de red una GUI debe operar sin importar el protocolo de comunicación que se utilice.

En general, una GUI presenta la información en un área rectangular en la pantalla llamadas ventanas (o windows). Las ventanas pueden colocarse unas sobre otras. Los usuarios pueden manipular las ventanas ya sea moviéndolas de un lugar a otro, variarles el tamaño, etc. Las ventanas contienen *objetos* que pueden ser seleccionados por medio del ratón, éstos generalmente son dibujos llamados *iconos.* Una ventana entera puede ser minimizada completamente de tamaño en un icono y el usuario puede devolverlo a su tamaño natural posteriormente.

Hasta hace unos años, el usuario tenía que trabajar en ambientes poco amigables. Son dos tipos de usuarios que más sufren en este tipo de situación: el usuario final que demanda facilidad de uso y el desarrollador de aplicaciones que tiene que atender estas demandas.

Desde hace varios años, la industria de cómputo ha estado en un constante debate acerca de las plataformas ideales de desarrollo de sistemas.

Alrededor de 1983, surge un nuevo concepto, GUI (Graphic User Interface), una interface gráfica para el usuario. El usuario ve en su pantalla el mismo sistema pero representado con gráficos.

Poco a poco, los GUI's se convirtieron en la plataforma ideal para el desarrollo de aplicaciones. Sin ir muy lejos, el principio de las aplicaciones basadas en la arquitectura Cliente/Servidor, es la entrega de la información directamente en el escritorio del usuario final.

Esta entrega se lleva a cabo a través de una red corporativa hacia una interface gráfica en donde los datos son preparados para presentarlos al usuario.

Las interfaces gráficas están diseñadas para que el usuario pueda trabajar con varias aplicaciones "a la vez", por medio de ventanas e iconos, los cuales sustituyen a los comandos de línea y estandarizan el manejo de las aplicaciones. El teclado es sustituido por un mouse. La pantalla del usuario se ve en general más bonita y más ordenada.

Características principales:

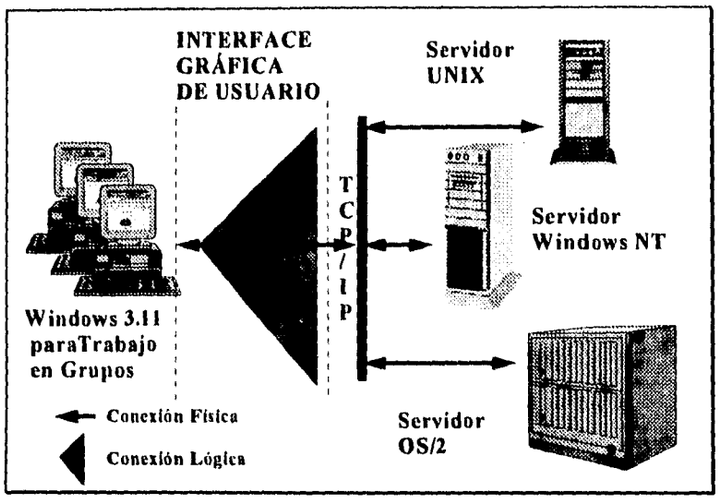
CONSISTENCIA. Las GUI's brindan mayor consistencia a las aplicaciones desde el punto de vista de estandarización de las mismas.

INTERCAMBIO ENTRE TAREAS. En diversos sistemas se pueden tener varias sesiones abiertas, aunque no todas se vean a la vez, por lo cual, se crearon los intercambiadores de tareas.

MULTITAREA. En la mayoría de los casos, el poder trabajar con varias aplicaciones a la vez, resulta más atractivo que el intercambio de manera simultánea. El concepto de multitarea convierte a una PC en una herramienta de trabajo muy poderosa y atractiva.

OBJETOS. Un objeto es una representación gráfica de varios elementos, generalmente iconos, dentro de esta imagen el programador puede incluir la aplicación misma y varios procedimientos.

Quizás pueda pensarse que la arquitectura Cliente/Servidor ya era utilizada desde hace tiempo por medio del sistema X Window, en cierta manera esto es correcto, sin embargo hay que señalar que el sistema X Window (también llamado X Window, aunque erróneamente X Windows), se ha convertido en una herramienta de software estándar utilizada en el desarrollo de interfaces gráficas de usuario sobre estaciones de trabajo. Debido a la independencia de dispositivo, las interfaces gráficas de usuario desarrolladas en una computadora, pueden ser ejecutadas sin problema en otras distintas. Pero existen importantes diferencias entre las GUI's y los sistemas X Window basadas en microcomputadoras, aunque tienen el mismo objetivo básico: Facilitar el uso de la computadora. Todos proporcionan elementos gráficos, por ejemplo, iconos y barras de desplazamiento, para comunicarse con el usuario. Los Sistemas X Window se limitan a la utilización de las librerías Xlib (ésta es una biblioteca de funciones para generar un protocolo llamado X similar al lenguaje de máquina) y Xt Intrinsics (en una herramienta que permite crear barras, botones, cuadros de diálogo, etc.) creando una interface con apariencia propia, en lugar de trabajar con la idea que otra persona tiene de una interface ideal como lo son las GUI's. Quizás lo mas importante es señalar que las GUI's a diferencia de los sistemas X Window no se centran en el uso de terminales tontas y sobre todo no saturan la red en términos de carga ya que todas sus instrucciones tienen que viajar por ella para ser procesadas, las terminales no colaboran con el trabajo y la red se aglomera de una manera considerable.



I.4.3.2 Sistemas Operativos de Redes (NOS).

Dentro de lo que es la arquitectura Cliente/Servidor la parte total es en primer término el diseño de la RED LAN, ya que es mediante la cual se va llevar a cabo todo el flujo de información. Generalmente cuando se piensa en una arquitectura de este tipo es porque se necesita de un Sistema de Información Administrativa.

Para ello se requiere de dispositivos interredes que acentúen la confiabilidad, disponibilidad y el servicio además de la conectividad. Las herramientas del oficio que tiene el sistema de administración para la formación de redes tales como boletas de rastreo de problemas, bases de datos para la resolución de problema y capacidades de contabilidad del tráfico necesitan aplicarse a las redes de área local (LAN's) y a los dispositivos interredes de las mismas.

Podríamos decir que la esencia de los sistemas operativos de redes son los protocolos de comunicación. Muchos de los protocolos desarrollados para redes de área local tienen problemas en las grandes redes empresariales. Este es precisamente el caso de IBM. Dos protocolos populares de la IBM, el NetBIOS y el SNA/DLC, no poseen un manejador de rutas de acceso (routing). No utilizan el direccionamiento de red por capas (Network Layer

Addressing) en la que se basan muchos manejadores de ruta que trabajan con múltiples protocolos.

Para encaminar a estos protocolos tales dispositivos los encapsulan, en la misma forma que manejan la arquitectura del sistema de red (SNA). Esto causa sobrecarga innecesaria, degradación en el rendimiento y costos WAN más altos.

El NetBIOS y la SNA/DLC utilizan un ruteo fuente, de manera que pueden enlazarse utilizando puentes de ruta fuente. Pero esta solución es menos óptima en las redes grandes que tienen múltiples conexiones WAN.

Por otro lado muchas de las redes de área local utilizan protocolos que son ruteables, incluyendo el TCP/IP, XNS, DECnet y al Novell IPX (en intercambio de paquetes interredes de Novell). Estos protocolos con frecuencia se encuentran en las redes Ethernet en donde está mejor desarrollada la conexión interredes.

En empresas que utilizan de unos y otros se requiere de una solución que pueda manejar a todos los protocolos de la LAN, los ruteables y los no ruteables, los que son ruteados por fuente y los que no lo son. Esto requiere de un protocolo independiente de la formación interred, uno que defina las rutas de todos los protocolos.

Este tipo de protocolos da al sistema de información administrativa las herramientas necesarias para manejar varios protocolos LAN. Aunque cada protocolo tiene la misma misión (la comunicación entre computadoras), cada uno de ellos implanta políticas administrativas diferentes para el control de acceso, la administración de dirección, la asignación de prioridades de tráfico, la seguridad y la recuperación de errores.

En algunos casos, los protocolos no tienen ninguno de estos requerimientos, de manera que tienen que venir de otra parte. La elección obvia es el dispositivo interred y el dispositivo de protocolo independiente para la formación de interredes que incluyan la herramienta para la administración de todos los protocolos de la red.

I.5 Creación de la plataforma Cliente/Servidor.

La creación de la plataforma Cliente/Servidor es relativamente sencilla en cuanto a conexión y hardware, en su gran mayoría, las aplicaciones corre en una pequeña PC y su comunicación con otras se lleva a cabo sobre el administrador de Windows NT. En su gran mayoría, y el objetivo primordial de programar en herramientas gráficas es para que se ejecuten en un ambiente tipo Windows, ya que esto permite que cualquier usuario neófito lo utilice rápidamente.

Para la creación de la plataforma es necesario considerar los siguientes puntos:

Seleccionar la topología y el equipo físico, esto es, diseñar la arquitectura física que tendrá la red.

Instalar el equipo físico y el sistema operativo de la red. Deben hacerse las conexiones por medio del cableado, las tarjetas y configurar las máquinas de acuerdo a su naturaleza, ya sean clientes o servidores.

Cargar las aplicaciones necesarias para llevar a cabo el manejo de la información del sistema.

Definir los atributos de cada usuario así como sus claves de acceso.

Establecer una administración de la red y los procedimientos de su soporte.

I.5.1. El Cliente.

El manejo de cliente que ofrece la arquitectura C/S es sumamente versátil ya que la estación de trabajo o cliente puede fungir como una máquina aislada, o bien, cuando sea necesario utilizar los recursos del servidor.

La plataforma de los clientes suelen ser computadoras personales que el único requisito que deben tener es soportar un software como el del ambiente windows. Cada uno de los clientes pertenecerá a un grupo de trabajo definido dentro de la administración de la red, además, cada grupo de trabajo o área formará parte a su vez de un dominio de red y se le instalará el software necesario para llevar a cabo los procesos, consultas y comunicaciones con el servidor.

La mayor función de un cliente en un sistema con ambiente Cliente/Servidor es la presentación de la las funciones y algunos asuntos lógicos (business logic). La interacción con usuarios finales con una aplicación está mejorada por medio de la presentación lógica. La presentación lógica es la capa de la aplicación que por medio de un manejo lógico interactúa de una manera más natural con el usuario. Todo ello se ha mejorado por los dispositivos físicos más recientes como el mouse, dispositivos de entrada y salida (monitores, teclados, lápices ópticos, etc.)

Las funciones de una presentación tradicional están basadas en el manejo de los caracteres recibidos desde el teclado y vislumbrados por el monitor donde el microprocesador hace un manejo secuencial de ellos para interpretarlos y procesarlos.

La evolución continúa de las funciones de presentación han estado estrechamente ligados con el alto desempeño que ofrecen las estaciones de trabajo que dan un manejo gráfico de la información.

Este manejo lo realiza el microprocesador por medio del control individual de los pixeles en la pantalla construyendo un mapa especial de bits o "bitmap". Cada bit del mapa corresponde a un pixel (elemento del gráfico) en la pantalla del monitor, así el procesador

puede pintar un gráfico en la pantalla ya sean caracteres, símbolos o dibujos. La resolución de la pantalla define el número de pixeles disponibles. Así el color, el tamaño y la posición de los objetos de la pantalla no están limitados por el formato de algún caracter en especial o por el número de renglones o columnas. Gracias a esta posibilidad los diseñadores de software han desarrollado interfaces gráficas intuitivas, efectivas y mnemónicas para las aplicaciones, enriqueciéndolos con gráficos estéticos y agradables dándole imaginación a la computación y facilitando la utilización de dispositivos de entrada y salida como audio y video. Imagínese una aplicación para oficina que ofrece un ambiente con gráficos (llamados iconos), un dibujo de un archivero define el almacenamiento de los archivos, un dibujo de una carpeta define un documento y un bote de basura la posibilidad de eliminar los documentos innecesarios.

El usuario final puede señalar con el ratón y efectuar un clic (oprimir uno de los botones) para correr alguna de las aplicaciones o seleccionar un documento, o bien, tomar uno de los documentos con el puntero del ratón (o mouse) arrastrándolo hasta el bote de basura y eliminarlo. Lo importante es que se utiliza el poder de las instrucciones basadas en caracteres por medio de gráficos.

I.5.2 El Servidor.

La parte toral de la tecnología Cliente/Servidor es la optimización de los servidores, es decir, es seleccionar las computadoras con la tecnología precisa en sus componentes electrónicos para adaptarse al software utilizado o viceversa. Uno de los beneficios de esta tecnología es el poder utilizar como servidores máquinas relativamente pequeñas, en este caso desde una 486DX a 66 Mhz, 16 Mb en RAM, hasta una minicomputadora que soporte OS/2. Es evidente que dentro de un ambiente gráfico la utilización de memoria extendida es extrema, con mayor razón para una computadora que funge como servidor de muchas otras por lo cual se recomienda utilizar máquinas que posean gran capacidad de ésta, también se agilizarían los cálculos con procesadores en paralelo y así podríamos terminar en una minicomputadora. Quizás lo más importante sea la facilidad con la que este tipo de tecnología permite la adición, sustracción, actualización o adecuación de cada uno de sus componentes tanto físicos como de software, la manera en que se pueden escalonar o distribuir aplicaciones, bases de datos, archivos, en síntesis, todos los recursos de la red.

Una de las características principales tras la tecnología Cliente/Servidor es la separación física de manejo de las presentaciones de los usuarios de otros servicios de las aplicaciones. En el complejo ambiente del sistema distribuido Cliente/Servidor ésta idea ha ayudado a que la evolución de los sistemas vaya hacia la especialización, que esté dirigida a la funcionalidad de una tarea en particular. Esta especialización facilita que tanto las funciones del servidor como las de funciones cliente-usuario se enfaticen en una presentación funcional.

La especialización de los servidores está mejor reflejada en la funcionalidad y diseño de los servidores de bases de datos. Básicamente, los servidores de bases de datos deben ser capaces de poder proporcionar una gran velocidad de almacenamiento en disco, poder significativo de proceso y la capacidad de correr muchas aplicaciones (clientes) simultáneamente. Un servidor es un proceso lógico que proporciona servicios a procesos que los soliciten. En la computación Cliente/Servidor, un cliente inicia la interacción Cliente/Servidor por el envío de una petición al servidor. Las funciones que un servidor

mejoraría son determinadas, en gran parte, por los tipos de peticiones que envían los clientes. Consecuentemente, si el servidor no está disponible para mejorar una función requerida por un cliente, entonces este servidor no participa en las interacciones cooperativas Cliente/Servidor.

En general, cuando un cliente y un servidor son interconectados en un sistema de red, el servidor puede proporcionar las siguientes funciones de los usuarios:

- *Compartición de archivos.*
- *Impresoras compartidas.*
- *Acceso a Bases de Datos.*
- *Servicios de Comunicación.*
- *Servicios de Facsímiles.*

Muchos de los servidores para proporcionar servicios específicos a sus usuarios deben cumplir con ciertas características, tales como:

•*Soporte Multiusuario.* La mayoría de los grupos de cliente necesitan desenvolverse en un ambiente multiusuario.

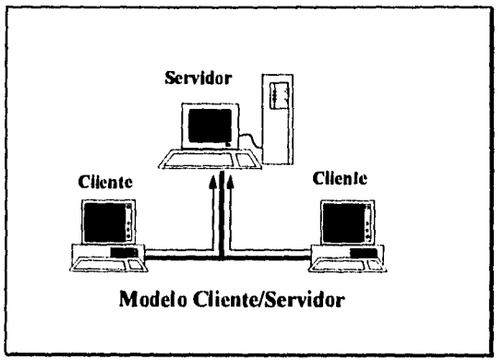
•*Escalabilidad.* Deben poder soportar la ampliación o disminución de los recursos dependiendo de la demanda del usuario. Es decir el sistema debe permitir la ampliación de los recursos de una manera sencilla y atractivamente económica en sus capacidades.

•*Desempeño.* Un servidor debe proporcionar altos niveles de desempeño para las necesidades de los negocios y los requerimientos de los usuarios sencillos y multiusuarios. Es decir, que el comportamiento de estímulos/respuesta se lleve a cabo en un mínimo de tiempo y con la precisión máxima.

•*Almacenamiento.* Debe permitir una gran velocidad en el manejo del almacenamiento de la información, su acceso a ella, depuración, consistencia y disponibilidad.

•*Multimedia.* Actualmente muchas de las aplicaciones tienen que manejar información en tecnología multimedia ya sean datos, imagen o sonido y el servidor debe ser capaz de manipularlo.

•*Interconexión.* Debe haber una perfecta compatibilidad en todo el trayecto de red optimizando interfaces y protocolos.



Modelo Cliente/Servidor

I.6 Ventajas y Desventajas de la Arquitectura Cliente-Servidor

Ventajas

- Permite el procesamiento distribuido entre varios servidores.
- Elimina algunos de los factores de costo asociados con el uso de Mainframes.
- Permite utilizar protocolos de red distintos con relativa facilidad.

Desventajas

- Es más difícil de Depurar, Probar y Afinar que un ambiente homogéneo.
- El ambiente de desarrollo debe ser idéntico al de producción, ya que se pueden exceder las capacidades que tenga del usuario.
- Dadas las plataformas heterogéneas en las que se implementa, para depurar un problema se tienen que hacer muchas consideraciones rápidamente.
- Se debe tener perfectamente bien controlado el aspecto de seguridad desde el principio del proyecto dado que el manejo de seguridad se pone en manos de distintas aplicaciones en lugar de una sola.

CAPITULO II

IDENTIFICACION DE LA PROBLEMATICA

II.1. Identificación de la problemática del usuario.

La función primordial del Area de Seguros es la contratación y negociación de pólizas de seguro de vehículos para la Institución y sus empleados.

Será obligatoria la contratación de este seguro, para todo el personal a quienes la Institución les otorgue crédito para la adquisición de un automóvil. Los empleados y funcionarios deberán tomar el seguro de cobertura amplia por el tiempo que permanezca vigente dicho crédito. Además la Institución facilitará la adquisición de este tipo de seguro a todos los funcionarios y empleados de planta para asegurar un máximo de dos vehículos, considerando que los mismos deberán ser de uso particular.

El período de vigencia y el importe de las primas de seguros se determinarán conforme al convenio que celebre la Institución con las compañías aseguradoras. La prima se descontará quincenalmente al empleado o funcionario vía nómina.

Las reclamaciones de indemnización se pueden realizar en el Area de Seguros, a través de Recursos Humanos o directamente en la compañía de seguros.

Si por cualquier motivo, concluye la relación laboral del empleado o funcionario con la Institución, el seguro se cancelará automáticamente.

Contratación, Control y Operación de Seguros:

1.- Para cualquier contratación de pólizas de seguro de personal de la Institución, deberá existir invariablemente la autorización expresa del Area de Recursos Humanos.

2.- La solicitud de alta, baja o modificación de seguro de automóvil, se podrá tramitar en el módulo de prestaciones de Recursos Humanos que corresponda.

3.- En aquéllos seguros donde sea opcional la forma de pago de una prima de seguro el empleado o funcionario deberá presentar la solicitud, autorizando a la Institución realizar los descuentos vía nómina, correspondientes al pago de la prima. Estos descuentos no se

consideran dentro de la capacidad de pago del empleado o funcionario por concepto de prestaciones económicas otorgadas por la Institución.

4.- El seguro de automóvil se renovará en forma automática al término de su vencimiento. El personal podrá solicitar su cancelación en cualquier momento, a través del formato establecido y anexando el original de la póliza correspondiente, excepto en el caso de que sea colateral a un préstamo para adquisición de automóvil, donde no podrá cancelarse hasta el término del plazo de vigencia del crédito.

5.- En la administración de los seguros contratados para la Institución, será indispensable contar con los servicios de un despacho de asesores de seguros (corredor), el cual fungirá como intermediario entre ésta y las compañías aseguradoras.

6.- La selección del despacho de asesores de seguros y de las compañías aseguradoras que trabajarán con la Institución en un determinado periodo de tiempo, se efectuará mediante concurso, el cual se llevará a cabo bajo los lineamientos que dicte la Dirección.

7.- El Area de Seguros vigilará que el servicio otorgado por el despacho de asesores de seguros y la compañía aseguradora sea adecuado y oportuno. Las quejas que pudieran presentarse por incumplimiento, se tramitarán directamente ante dicha Area.

Facultades y responsabilidades.

1.- Area de Seguros:

- Controlar centralizadamente los seguros, en la ciudad de México, D.F.
- Negociar la contratación, renovación y operación de los seguros, así como controlar los mismos contando siempre con la supervisión del Comité de Seguros. La Gerencia será la encargada de registrar las altas, bajas o notificación de siniestros de automóviles procedentes de empleados o bien propiedad del Grupo Financiero; posteriormente mediante el corredor de seguros solicitará las pólizas a la Compañía de Seguros. La Gerencia se encargará de hacer llegar a los empleados o a las Areas las pólizas.

- Vigilar el cumplimiento de las condiciones negociadas en las pólizas de seguro, controlar el pago oportuno de las primas y el cobro eficiente de las indemnizaciones, así como supervisar los servicios que prestan el despacho de asesores externos y las compañías aseguradoras. Verificar que el cobro sea el correcto y efectuar los pagos, para esto solicitará mediante fichas contables al Area de Contabilidad que se efectúe el pago a la compañía aseguradora.

- Evaluar, elegir y contratar a las compañías aseguradoras que en función de: servicio, cotizaciones o precio y amplitud de coberturas, sean las que brinden mejores condiciones y satisfagan eficientemente las necesidades del grupo.

2.- Corredor de Seguros:

- Asesoría técnica para coberturas, contratación de seguros, siniestros, ya sea manejados con su intermediación con las compañías aseguradoras o que estén en proceso de ajuste profesional.

- Tramitación administrativa de las pólizas de seguro y endosos, así como reportes periódicos a la Institución.

- Expedición y tramitación interna de pólizas de seguros, así como los endosos modificatorios que se requieran.

Pago de primas:

Dependiendo de cada contrato de seguro, el pago de primas deberá efectuarse por lo menos una vez al mes. Las primas deberán ser liquidadas en su totalidad, en un período máximo de 30 días a partir de su fecha de expedición o vigencia. Dichas primas deberán ser calculadas con base a un equilibrio racional entre la máxima protección o cobertura de los riesgos y el mínimo costo, considerando métodos tales como cláusulas de ajuste, deducibles y dividendos o bonificaciones.

Cobranza:

La conciliación de los saldos de primas pendientes de pagar, se efectuará periódicamente durante el ejercicio o vigencia de las mismas, a la discreción y requerimiento del Area de Seguros.

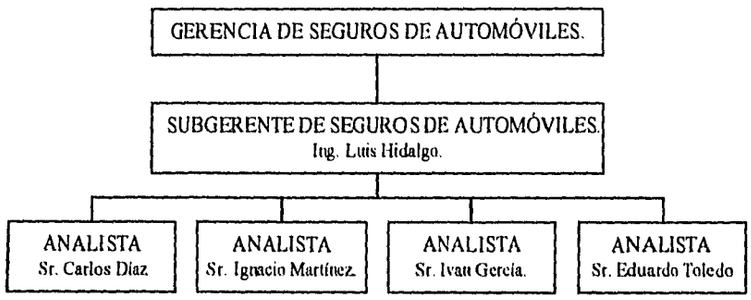
Problemática:

La problemática actual que vive la Gerencia de Seguros de Automóviles se puede resumir en:

1. Dado que es una área concentradora de información, la misma no les llega con la oportunidad ni presentación requerida.
2. No cuentan con el equipo de cómputo adecuado.
3. Funciona limitadamente en materia informática.
4. Bases de datos fraccionadas con información redundante e incompleta.
5. No es posible integrar las pólizas de todas los regiones y empresas que componen al grupo financiero.
6. La información fuente no es recibida en forma oportuna lo cual genera rezagos en procesos operativos.
7. Carga excesiva de trabajo, ya que el proceso operativo en su mayoría es manual.

II.1.1. Identificación de los departamentos usuarios e individuos a contactar.

La Gerencia de Seguros de Automóviles tiene el siguiente organigrama:



El Gerente de está Area el Lic. Miguel Angel Soledad es el encargado de llevar el control contable, así como analizar las alternativas que le presentan la diferentes compañías aseguradoras y tomar decisiones sobre la contratación de pólizas.

El SubGerente se encarga de coordinar a cada analista, de subir a la base de datos la información que le envía el corredor de seguros y controlar todos los movimientos administrativos para el control de las pólizas.

El control de altas y cancelaciones de pólizas son trabajo de dos de los analistas Carlos Díaz e Ignacio Martínez y el control de siniestros del Sr. Ivan García y el Sr. Eduardo Toledo se encarga de dar apoyo general a la Gerencia.

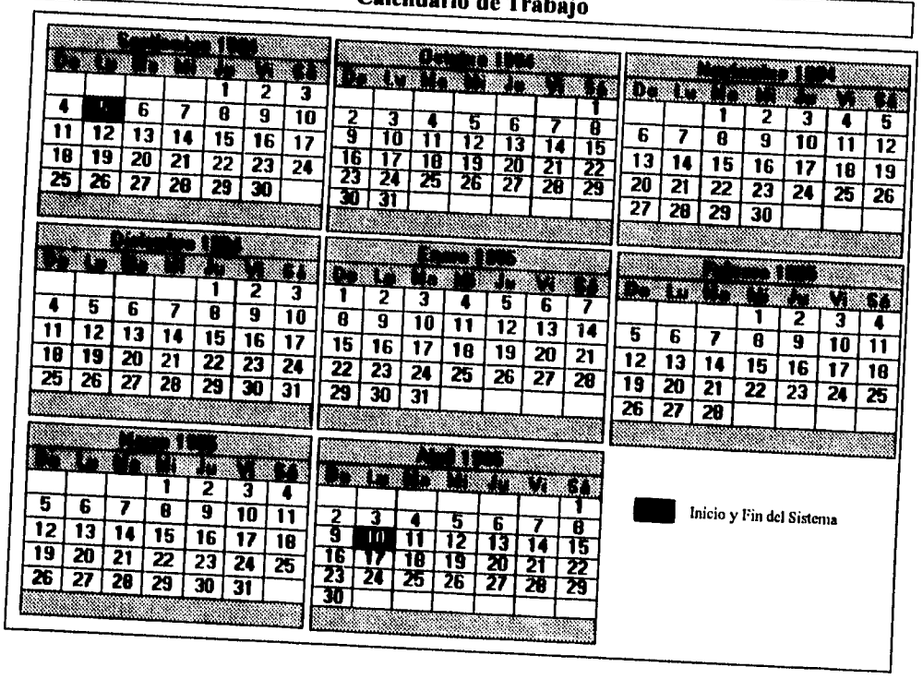
CAPITULO III

PLANEACION DEL PROYECTO

III.1. Planeación del Proyecto

En la primera reunión los usuarios plantearon la problemática de la Gerencia de Seguros en forma general, además se acordaron las fechas y horas en las que se haría el levantamiento de información del trabajo operativo y las personas responsables de proporcionarla. Con base en lo anterior se planteó el siguiente calendario de trabajo, posteriormente se muestra el plan de trabajo donde se ve el desarrollo del proyecto por etapas, así como la duración de cada una.

Calendario de Trabajo



Plan de Trabajo

	Etapa		Sub-Etapa	Fecha de Inicio	Fecha de Terminación
1.-	Análisis Preliminar (Levantamiento de información)			***05/09/1994***	***16/09/1994***
		1.1	Sub'Dirección	*05/09/1994*	*05/09/1994*
		1.2	Gerencia	*06/09/1994*	*06/09/1994*
		1.3	Sub'Gerencia	*06/09/1994*	*06/09/1994*
		1.4	Registro de pólizas	*07/09/1994*	*08/09/1994*
		1.5	Siniestros de vehículos	*09/09/1994*	*09/09/1994*
		1.6	Cancelación de pólizas	*12/09/1994*	*12/09/1994*
		1.7	Pagos y Bonificaciones	*13/09/1994*	*14/09/1994*
		1.8	Control Contable	*15/09/1994*	*16/09/1994*
2.-	Análisis			***19/09/1994***	***11/11/1994***
		2.1	Objetivos del Sistema Propuesto	*19/09/1994*	*20/09/1994*
		2.2	Análisis de Requerimientos	*21/09/1994*	*30/09/1994*
		2.3	Determinación de Información Relevante	*03/10/1994*	*14/10/1994*
		2.4	Identificación de Procesos y Productos	*17/10/1994*	*28/10/1994*
		2.5	Requerimientos de Software y Hardware	*31/10/1994*	*11/11/1994*
3.-	Diseño del Sistema			***14/11/1994***	***02/12/1994***
		3.1	Diagrama Entidad-Relación.	*14/11/1994*	*18/11/1994*
		3.2	Diagramas de Flujos de Datos	*21/11/1994*	*25/11/1994*
		3.3	Diseño de Pantallas	*28/11/1994*	*02/12/1994*
4.-	Desarrollo del Sistema			***05/12/1994***	***03/03/1995***
		4.1	Módulo de Catálogos	*05/12/1994*	*23/12/1994*
		4.2	Módulo de Mantenimiento a Pólizas	*19/12/1994*	*27/01/1995*
		4.3	Módulo de Pagos y Bonificaciones	*09/01/1995*	*10/02/1995*
		4.4	Módulo de Transferencias de Información	*06/02/1995*	*03/03/1995*
5.-	Pruebas			***06/03/1995***	***24/03/1995***
		5.1	Pruebas	*06/03/1995*	*24/03/1995*
6.-	Implantación			***20/03/1995***	***14/04/1995***
		6.1	Implantación	*20/03/1995*	*14/04/1995*

CAPITULO IV

ANALISIS DEL SISTEMA

IV.1 Objetivos del Sistema

Como resultado de charlas sostenidas con el personal de la Gerencia de Seguros de Automóviles, así como la observación en condiciones reales de las actividades de trabajo se definieron los siguientes objetivos para el Sistema de Seguros de Automóviles:

- Desarrollar un sistema de información para el control y administración de las operaciones de seguros de automóviles tales como altas, cancelaciones y siniestros.
- El sistema de Seguros de automóviles deberá llevar el control de pagos y bonificaciones en forma automática.
- Construir una base de datos que contenga información confiable y completa.
- Compartir información con otros sistemas para evitar duplicidad e incongruencia.
- Definir un sistema automatizado, que permita a cada usuario el acceso a la información en cualquier momento.
- Deberán de existir claves de acceso restringido, de tal manera que cada usuario pueda consultar y afectar solamente aquella información que le permita el cumplimiento de sus funciones.

IV.2. Determinación de la información utilizada.

IV.2.1. Fuentes originadoras de información.

- **Funcionarios y empleados**: Que tengan crédito para automóvil otorgado por la Institución deberán tomar el seguro de cobertura amplia por el tiempo que permanezca vigente dicho crédito.

- **Areas**: Todos los automóviles propiedad del grupo deberán de estar asegurados.

- **Nómina**: Es esta área la encargada de descontar en pagos quincenales la prima del seguro a los empleados, para lo cual, proporciona al Area de Seguros los datos generales de todos los mismos.

- **Recursos Humanos:** Es la encargada de recibir y autorizar las solicitudes de alta, baja y modificación del seguro, y enviarla al Area de Seguros.

- **Corredor de seguros:** Tramita administrativamente las pólizas de seguros y endosos, solicita los cobros al Area de Seguros y efectúa el pago de indemnizaciones a la misma, así como genera reportes periódicos a las distintas áreas.

- **Contabilidad:** Es el área encargada de generar los pagos a la compañía aseguradora a petición del Area de Seguros.

IV.2.2. Procesos

- **Integración de la información**:
  - Registro automático de las altas de pólizas y endosos.
  - Registro automático de cancelaciones de pólizas
  - Registro automático de siniestros .

- **Cálculo de primas**: Se debe calcular el importe de la prima a pagar dependiendo del automóvil, modelo y tipo de cobertura.

- **Pagos y bonificaciones:** Registro del pago de prima y de bonificaciones en caso de altas o cancelaciones.

- **Control contable:** Registrar y controlar los movimientos contables generados ya sea por altas (pagos) o bajas por cancelación (bonificaciones), y llevar un arrastre de saldos.

- **Renovaciones:** Respaldar los datos de las pólizas con vigencia a terminar e ingresar los datos de la siguiente.

- **Generación de disquetes:** Bajar la información de la base de datos de altas o bajas y grabarla en disquete para enviarla a la nónina y se efectúen los descuentos.

- **Transferencia de Información:** Subir a la base de datos la información que llega por parte del corredor de seguros o del Area de Nónina.

IV.2.3. Productos Generados.

- **Pólizas:** Expedición de pólizas de seguros.

- **Etiquetas:** Generar etiquetas con datos del empleado para enviarle pólizas y otros documentos.

- **Fichas contables:** Emitir fichas contables con movimientos de control contable.

- **Reportes para:**
  - Gerencia de Seguros de Automóviles.
  - Area de Nómina.
  - Areas del interior.
  - Corredor de Seguros

- **Comunicados para:**
  - Area de Nómina
  - Areas del interior
  - Area de Pagos
  - Corredor de seguros.
  - Area de Prestaciones.

IV.3. Funciones del Sistema Propuesto

Como producto de las entrevistas realizadas con el Area de Seguros de Automóviles, de las aplicaciones Micro con las que contaban y del análisis realizado se identificaron los siguientes módulos y funciones para el sistema propuesto:

MODULO 1: MANTENIMIENTO DE POLIZAS

Propósito: Permitirá registrar los datos de vehículos de las diferentes pólizas para los procesos de altas, cancelaciones y siniestros. Automáticamente afectará el control contable y el de pagos y bonificaciones.

MODULO 2: PAGOS Y BONIFICACIONES

Propósito: Controlará el registro de pagos de primas y de bonificaciones en caso de bajas. El registro se generará como consecuencia de dar una alta de pólizas o de registrar una baja. El registro que se llevará aquí debe ser congruente con la información en el control contable.

MODULO 3: CONTROL CONTABLE

Propósito: Controlará el registro de movimientos contables generados ya sea por altas (pagos) o bajas por cancelación (bonificaciones), y llevará un arrastre de saldos. Los registros se deben generar automáticamente, y deben ser congruentes con la información en el registro de pagos y bonificaciones.

MODULO 4: MANTENIMIENTO A CATALOGOS

Propósito: Servirá para actualizar los diferentes catálogos que se utilizan en el sistema.

MODULO 5: TRANSFERENCIA DE INFORMACION

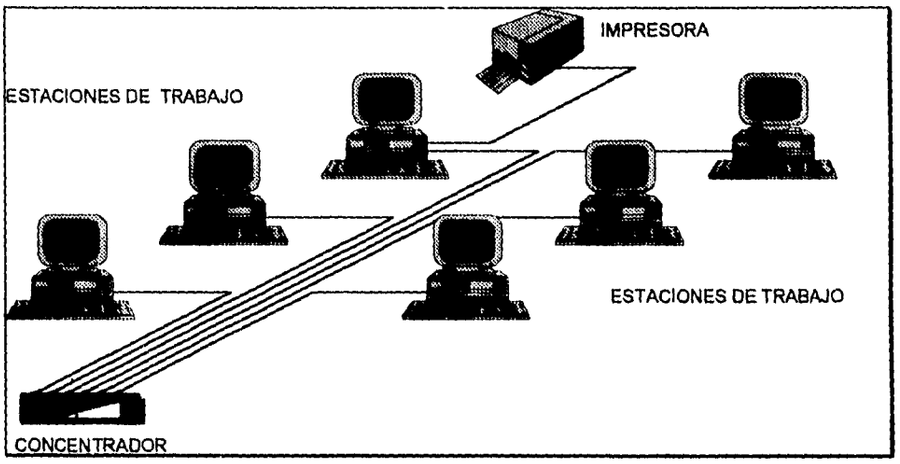
Propósito: Permitirá subir y bajar información a la base de datos.

Cada uno de los módulos anteriores exceptuando el de transferencia de información incluyen generación de reportes, comunicados y fichas contables para: el Area de Recursos Humanos, áreas del interior, empresas del grupo y la compañía de seguros .

IV.4 Requerimientos de Hardware y Software

Se analizaron los pros y contras de dos configuraciones de hardware, configuración de red con grupo de trabajo y configuración de red con servidor (cliente/servidor), dadas las necesidades del proyecto y gracias al apoyo que brindó la Subdirección de Seguros y Fianzas fue posible optar por la configuración de cliente/servidor, más poderosa y flexible que la de grupo de trabajo. Enseguida se presentan las caracteristicas y requerimientos de dichos esquemas.

IV.4.1. Configuración de red con Grupo de Trabajo



Estaciones de Trabajo:

- Procesador: 486 a 50 Mhz.
- Memoria: 8 Mb.
- Disco Duro: 200 Mb.
- Tarjeta de Red: Ethernet.

Software en cada estación de trabajo:

- *Windows for Workgroups*, para permitir la interconexión de las máquinas.
  - Compartición de los discos duros de todas las máquinas de la red.

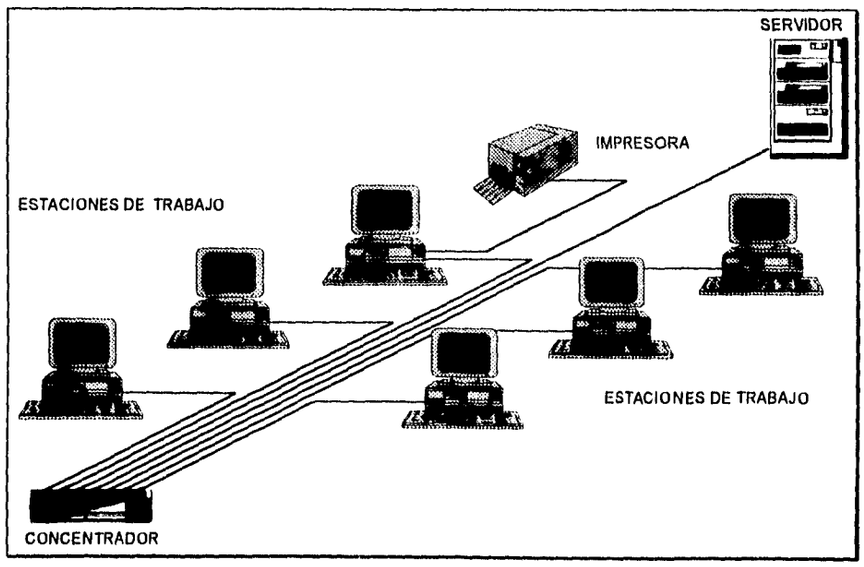
- Compartición de periféricos (impresoras, CD-ROMs, etc.).
- Manejo de correo electrónico.

- Base de Datos *Access*
  - Control de acceso a la información.
  - Seguridad de transacciones en caso de caídas del sistema.
  - Compatibilidad con otros sistemas de información (dBASE, SQL Server, etc.)
  - Compartición de información en la red.
  - Uso recomendado: para acceso local (en la misma máquina), ya que el acceso por red se puede hacer hasta 6 veces más lento.

Costo aproximado de la solución (dólares):

6 Computadoras	$ 12,000.00
6 Tarjetas de red	$ 720.00
1 Concentrador de 12 puertos	$ 1,700.00
Total	$ 14,420.00

IV.4.2. Configuración de red con Servidor.



Servidor:

- Procesador: Pentium a 90 Mhz.
- Memoria: 32 Mb.
- Disco Duro: 1 Gb.
- Unidad de respaldo DAT.

Software en el Servidor:

- Sistema operativo *Windows NT:*
  - Sistema real de multitareas: se puede atender a varios procesos simultáneamente.
  - Administración integral de toda la red: permite la administración de los usuarios, el control de entradas y salidas de usuarios y de quién usa información.
  - Alto nivel de seguridad: permite restringir el acceso a la información y realizar respaldos en línea.
  - Múltiples protocolos de red y conectividad abierta: permite interconectarse con otras redes e incluso con equipos macro.

- Servidor de Base de Datos *SQL Server:*
  - Arquitectura cliente/servidor: el procesamiento se lleva a cabo en el servidor en lugar de usar el tiempo de la estación de trabajo.
  - Integridad garantizada de la Base de Datos: Tiene controles automáticos para que la información mantenga su consistencia.
  - Alta seguridad: Controla el acceso a la base de datos, a sus tablas, registros y campos.
  - Procesamiento de transacciones: permite recuperar la información que se esté procesando, en caso de una falla del sistema o de la corriente.
  - Ofrece la posibilidad de conectarse con otros servidores para utilizar su información.
  - Uso recomendado: para acceso en red, ya que genera un mínimo de tráfico en la red y el tiempo de respuesta hacia cualquier estación de trabajo es muy alto, incluso en estaciones remotas (en otro edificio o ciudad).

Estaciones de Trabajo:

- Procesador: 486 a 50 Mhz.
- Memoria: 8 Mb.
- Disco Duro: 200 Mb.
- Tarjeta de Red: Ethernet.

Software en cada estación de trabajo:

- *Windows for Workgroups*, para permitir la interconexión de las máquinas.
  - Compartición de los discos duros de todas las máquinas de la red.
  - Compartición de periféricos (impresoras, CD-ROMs, etc.).
  - Manejo de Correo Electrónico.

Costo aproximado de la solución (dólares):

1 Servidor	$ 13,000.00
Windows NT Advanced Server	$ 1,500.00
SQL Server	$ 8,000.00
6 Computadoras	$ 12,000.00
6 Tarjetas de red	$ 720.00
1 Concentrador de 12 puertos	$ 1,700.00
Total	$ 36,920.00

CAPITULO V

DISEÑO Y DESARROLLO DEL SISTEMA

V.1. Consideraciones Generales.

El Sistema de Seguros de Automóviles se desarrolló bajo la filosofía Cliente/Servidor. Este contempla para la parte de servidor una computadora con sistema operativo Microsoft Windows NT Advanced Server V3.1 para el manejo de archivos e impresión y Microsoft SQL Server V4.21 para servicios de base de datos.

Para la parte del cliente se utilizaron MS-DOS y Microsoft Windows for WorkGroups V3.11 y la aplicación se desarrolló utilizando Microsoft Visual Basic V3.0 así como algunas extensiones a este ambiente de programación desarrolladas en Microsoft C++ V1.5 y OLE Automation V2.0.

Desarrollo parte del Servidor

Las metas de diseño de la parte Servidor incluye lo siguientes puntos:

- Seguridad en el manejo de usuarios e información
- Base de datos poderosa y flexible
- Servicios de red

Seguridad en el manejo de usuarios e información

El manejo de información, en cuanto a la seguridad que el sistema de Seguros de Automóviles provee, es acorde a las necesidades de el área usuaria. El desarrollo de las distintas aplicaciones que compone el sistema, únicamente obedece al conjunto de restricciones y permisos definidos en el dominio del servidor.

La parte del servidor del Sistema de Seguros de Automóviles se basa en el sistema operativo Microsoft Windows NT Advanced Server V3.11. El Sistema de Seguros de Automóviles aprovecha la facilidad que Windows NT AdVanced Server y SQL Server proveen a través del esquema Mixed Security Environment ofreciendo una sola clave de acceso para el servidor de red y para la Base de Datos.

Base de datos poderosa y flexible

La implementación de la Base de Datos del Sistema de Seguros de Automóviles se basa en el Manejador Relacional Cliente/Servidor Microsoft SQL Server para Windows NT V4.21. A lo largo del desarrollo se hizo uso intensivo de las facilidades que SQL Server proporciona a través de objetos como tablas, índices, stored procedures, triggers, etc., uso que obedece a las necesidades específicas de cada módulo de programa desarrollado.

Servicios de red

Como una característica inherente a cualquier servidor de red, el servidor proporciona al Sistema de Seguros de Automóviles el manejo compartido de archivos, aplicaciones e impresión y además, en caso necesario, provee a estaciones de trabajo con acceso remoto a través de línea telefónica.

La naturaleza del Sistema de Seguros de Automóviles (conforme a la filosofía Cliente/Servidor) ha sido desarrollado contemplando que para la realización de un proceso, es adecuado distribuir las tareas que conforman dicho proceso entre un especialista en el manejo y seguridad de grandes volúmenes de información y por otro lado un especialista en el manejo gráfico de la misma que proporcione un fácil acceso al usuario final.

Desarrollo parte Cliente

Las metas del diseño parte Cliente considera los siguientes puntos:

- Facilidad de uso por medio de un ambiente gráfico.
- Aplicaciones de rápido desarrollo, que optimen el uso de recursos y que utilicen tecnología actual en el manejo de información.
- Integración de la información con otras herramientas y aplicaciones.

Facilidades de uso por medio de un ambiente gráfico

La parte del Cliente se lleva a cabo mediante aplicaciones desarrolladas para el ambiente gráfico Microsoft Windows en su modalidad Windows for WorkGroups V3.11

para las facilidades que, para el procesamiento de información, provee este medio ambiente y aprovechando la familiaridad que los usuarios tienen con el mismo. Aunado a estas ventajas, Windows for WorkGroups se integra fácilmente a Windows NT AdVanced Server y permite lograr ambiente cooperativo a nivel grupo de trabajo o a nivel dominio.

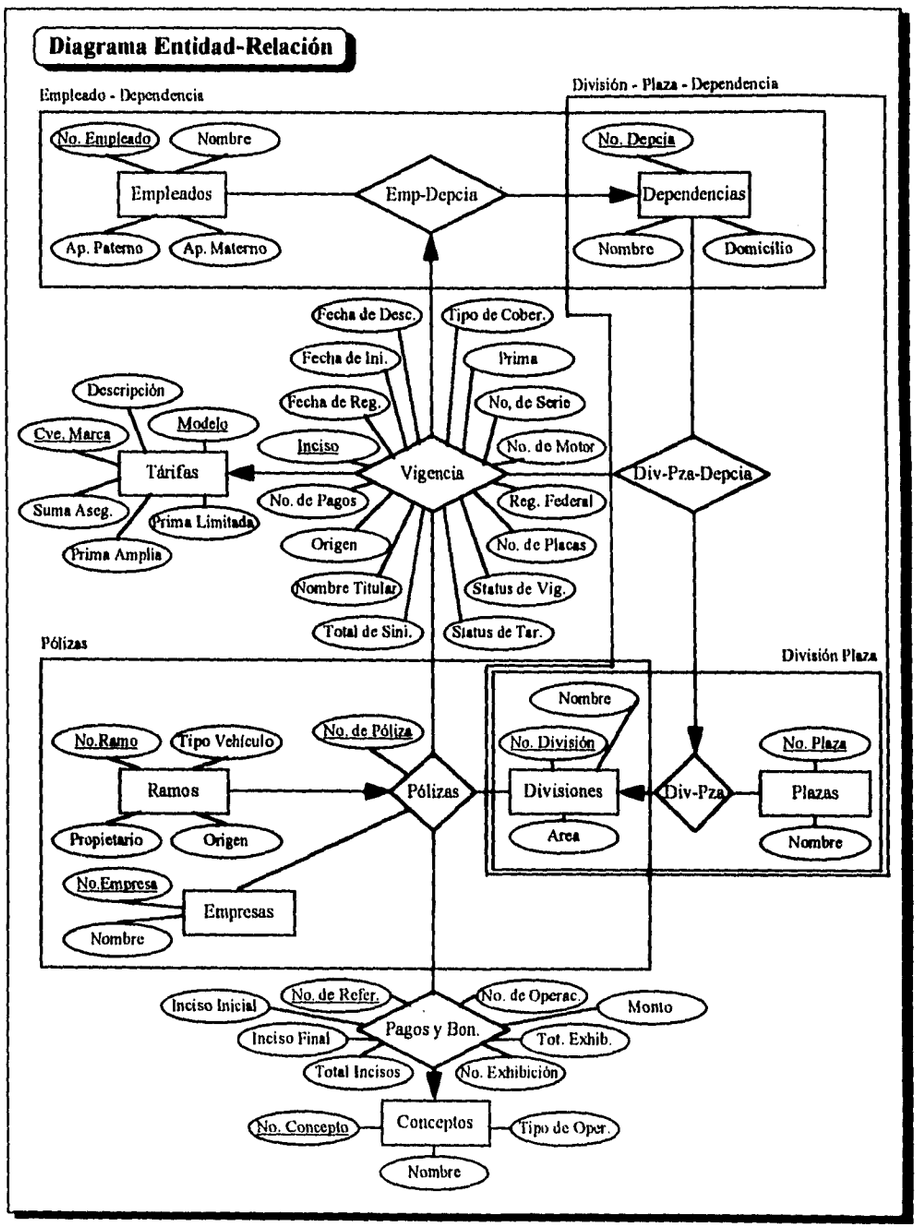
Aplicaciones de rápido desarrollo, que optimen el uso de recursos y utilicen tecnología actual en el manejo de Información

Para el desarrollo de los módulos de programa Cliente se utiliza como herramienta de programación Microsoft Visual Basic V3.0 y módulos y Dynamic-Link Libraries (DLLs) desarrolladas en Microsoft Visual C++ V1.5. En combinación estas herramientas permiten. por un lado, un rápido desarrollo de aplicaciones para Windows y por otro, aprovechar al máximo los recursos con que cuente la computadora en donde sean ejecutadas tales aplicaciones.

Integración de la información con otras herramientas y aplicaciones

Dentro de las principales características que Microsoft Visual Basic provee están la comunicación directa con manejadores de base de datos relacionales por medio del concepto Open Database Connectivity ODBC e integración con otras aplicaciones Windows a través de Dynamic Data Exchange y OLE Automation. De esta forma las aplicaciones extraen la información residente en las bases de datos y administradas por Microsoft SQL Server, la presenta en forma amigable al usuario, quien a su elección puede cambiarla, consultarla o llevarla, por ejemplo, a aplicaciones como Microsoft Excel V5.0 o Microsoft Word for Windows V6.0.

V.2. Diagrama Entidad-Relación



V.3. Diagramas de Flujo de Datos

Diagrama de contexto

Con el diagrama de contexto del sistema, se representa en forma gráfica y en un solo diagrama el proceso completo de la administración de seguros de automóviles.

Se muestran todos los catálogos y tablas que se utilizan como entrada de información y las salidas del sistema.

Además se señala la participación que tienen algunas entidades de la Institución en el proceso, y la comunicación que se tiene con otros sistemas ya existentes.

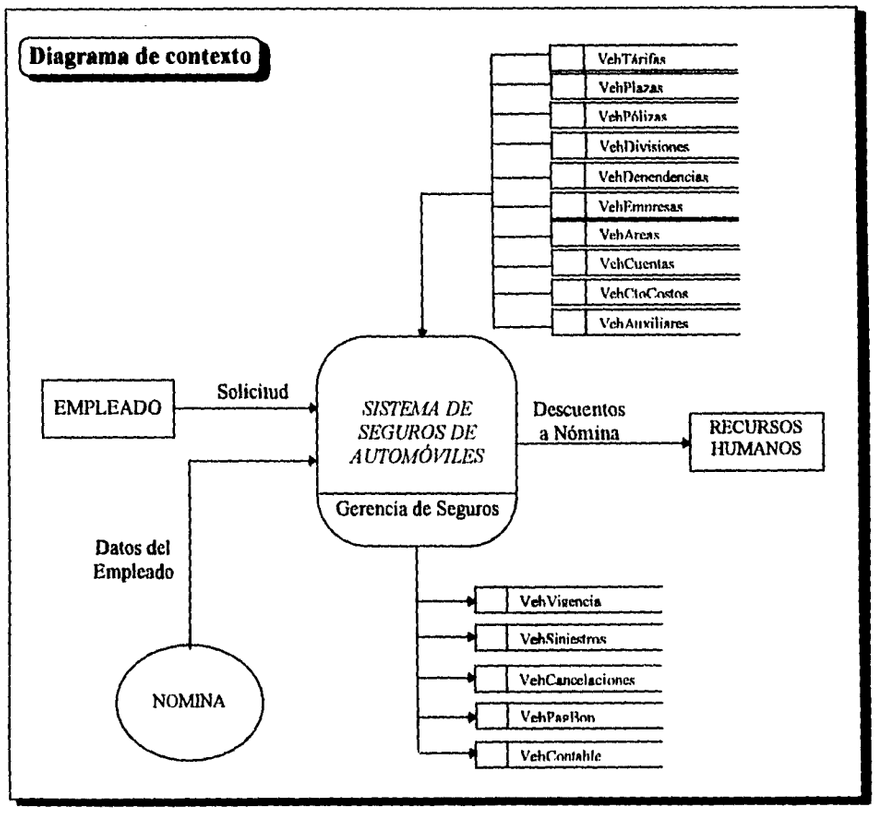
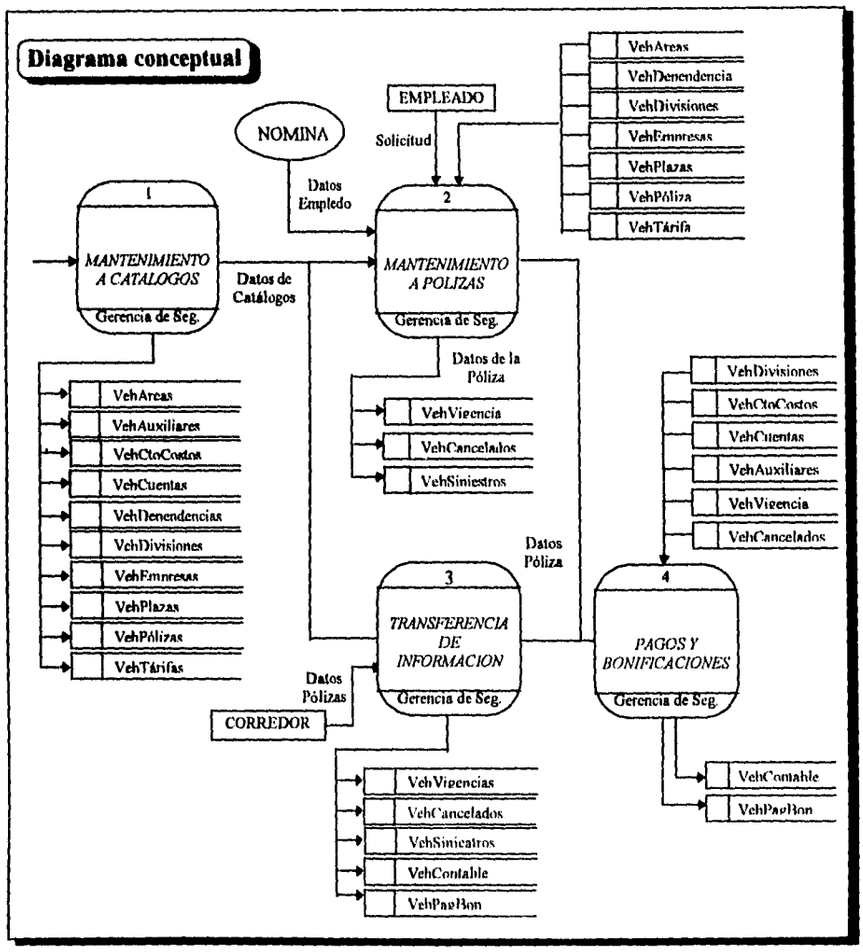


Diagrama conceptual o diagrama de nivel cero.

El Sistema de Seguros de automóviles está compuesto por 4 procesos principales, Mantenimiento a Catálogos (1), Mantenimiento a Pólizas (2), Transferencia de información (3) y Pagos y Bonificaciones (4). En el nivel 0 del DFD se representan estos 4 procesos y la interacción que tienen entre si.

Cada uno de los procesos se representan con entradas y salidas de datos.

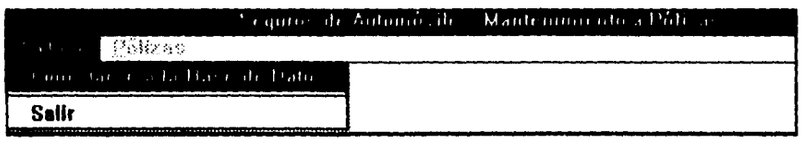


V.4. Diseño y Desarrollo de los Módulos

En el diseño del sistema se acordó que los módulos se presentarían de manera independiente uno de otro debido a la distribución de tareas dentro del área usuaria, es decir, cada usuario dependiendo de las actividades que realiza entrará solo al módulo con el que va a trabajar, evitando de esta manera cargar las pantallas que no va a usar.

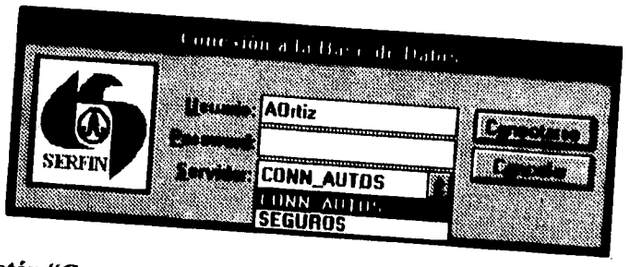
Para entrar al sistema desde el administrador de programas de *Windows* se creó un grupo llamado *"Seguros de Vehículos"*. El grupo muestra los módulos por los que está conformado el sistema, representado cada uno por un icono ("Pólizas", "Pagos y Bonificaciones", "Transferencia" y "Catálogos").

Una vez que se ha entrado al módulo deseado, aparece la ventana principal la cual tiene una lista de nombres de menús, todas tienen uno llamado *"Archivo"* que al entrar al módulo es el único que se encuentra habilitado. Para poder abrir los demás menús, se debe abrir el de *"Archivo"* y elegir la opción *"Conectarse a la Base de Datos"*.



Al elegir la opción antes mencionada aparecerá la pantalla de *"Conexión a la Base de Datos"*. Esta se diseñó como medida de protección para restringir el uso de la información del sistema, la forma solicita el nombre de un usuario, password y el nombre del servidor al que se desea conectar, en este último campo se muestra una lista con los servidores disponibles para la conexión. Una vez que se han proporcionado los datos mencionados

anteriormente se tiene un botón con el nombre de *"Conectarse"* este contiene el siguiente código para hacer la validación:



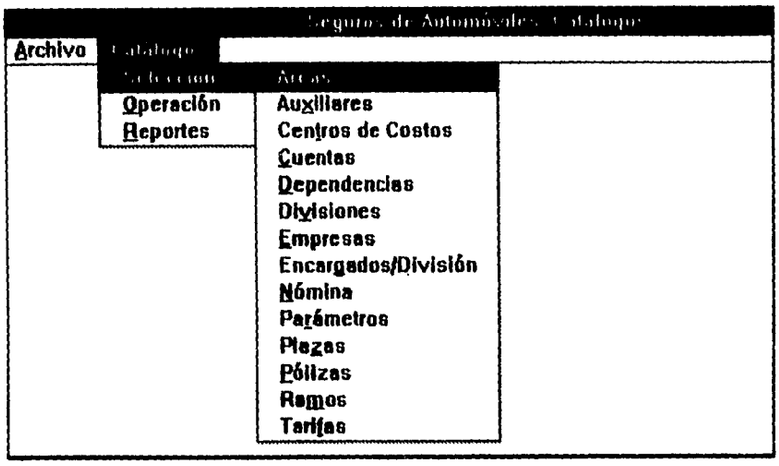
Código del botón "Conectarse" de la pantalla *"Conexión a la Base de Datos"*

```
Sub pbOK_Click ()
    Dim sUser As String
    Dim sPwd As String
    Dim sDSN As String
    Dim Connect As String
    sUser = txUserName
    sPwd = txPassword
    sDSN = cbDataSources
    If sUser = "" Then
        MsgBox "Usuario inválido", MB_ICONEXCLAMATION
        Exit Sub
    End If
    On Error Resume Next
    mEstatus = "Abriendo la base de datos en el Servidor. Favor de esperar..."
    relojmouse
    Connect = "ODBC;" + "UID=" + sUser + ";PWD=" + sPwd
    Set dbVehiculos = OpenDatabase(sDSN, False, False, Connect)
    resetmouse
    mEstatus = ""
    If Err = 0 Then
        Unload Me
    Else
        MsgBox "No puede conectarse a " & sDSN, MB_ICONSTOP
    End If
End Sub
```

Módulo de Mantenimiento a Catálogos

Este módulo se diseñó para actualizar, consultar y emitir listados de los diferentes catálogos que se utilizan en el Sistema. A través de éste el usuario puede controlar la información residente en la base de datos que se considera básica para las operaciones del sistema.

La pantalla principal tiene el menú de *"Catálogos"* con las opciones de: *"Selección"*, *"Operación"* y *"Reportes"*.



La opción de *"Selección"* permite indicar a cuál de los catálogos se le dará mantenimiento; al quedar seleccionado el catálogo aparecerá una marca de verificación (√) a la izquierda del nombre del catálogo la cual indica que está activo.

La opción de *"Operación"* permite poner al corriente el catálogo que se haya seleccionado. Las opciones son las siguientes: altas que permite capturar los datos del catálogo en cuestión; bajas sirve para eliminar registros erróneos; modificaciones que permite corregir los datos y consultas que se utiliza para revisar los datos en la ventana

Una vez que seleccione el catálogo y una acción de esté, el sistema mostrará la pantalla para darle mantenimiento al catálogo, ésta tendrá como titulo *"Mantenimiento a Catálogos:"* seguida del nombre del catálogo y entre paréntesis la acción a ejecutar sobre dicho catálogo. El diseño de las pantalla de catálogos fue mostrar los atributos para esa entidad de manera sencilla.

A continuación se presenta el código que carga la forma del catálogo correspondiente.

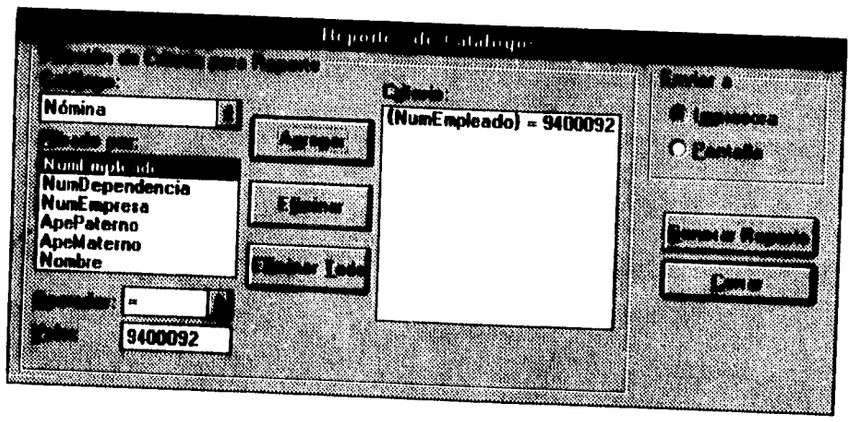
Carga la pantalla correspondiente para altas, bajas, modificaciones o consultas del catálogo seleccionado.

```
Sub cargaFmaCat (ByVal numCat As Integer, ByVal accion As Integer)
    Static aCap(15), aAcc(4) As String
    Dim capt As String
    Dim pos As Integer
```

```
aCap(1) = "          Mant. a Catálogos : Areas (*)      "
aCap(2) = " Mantenimiento a Catálogos : Auxiliares  (*)          "
aCap(3) = "    Mant. a Catálogos : Cto. Costos (*)          "
aCap(4) = "    Mant. a Catálogos : Conceptos (*)          "
aCap(5) = "          Mant. a Catálogos : Cuentas (*)      "
aCap(6) = "    Mant. a Cat.: Dependencias (*)     "
aCap(7) = "     Mant. a Cat.: Divisiones (*)     "
aCap(8) = "     Mant. a Cat.: Empresas (*)     "
aCap(9) = "   Mant. a Cat.: Encarg./Div. (*)     "
aCap(10) = "          Mant. a Catálogos : Empleados (*)      "
aCap(11) = "Mant. Cat.: Parámetros (*)  "
aCap(12) = "     Mant. a Cat.: Plazas (*)     "
aCap(13) = "          Mant. a Catálogos : Pólizas (*)      "
aCap(14) = "    Mant. a Catálogos : Ramos (*)      "
aCap(15) = "          Mant. a Catálogos : Tárifas (*)      "
aAcc(1) = "Altas"
aAcc(2) = "Bajas"
aAcc(3) = "Modificaciones"
aAcc(4) = "Consultas"
pos = InStr(aCap(numCat), "*")
capt = Mid(aCap(numCat), 1, pos - 1)+aAcc(accion)+Mid(aCap(numCat), pos +
1) +                    Format$(Now, "dd-mm-yyyy")
Select Case numCat
Case 1
    frmCatManAre.Caption = capt
    relojMouse
    frmCatManAre.Show 1
Case 2
    frmCatManAux.Caption = capt
    relojMouse
    frmCatManAux.Show 1
Case 3
    frmCatCtoCos.Caption = capt
    relojMouse
    frmCatCtoCos.Show 1
Case 4
    frmCatManCon.Caption = capt
    relojMouse
    frmCatManCon.Show 1
Case 5
    frmCatManCtas.Caption = capt
    relojMouse
    frmCatManCtas.Show 1
Case 6
    frmCatManDep.Caption = capt
    relojMouse
    frmCatManDep.Show 1
Case 7
    frmCatManDiv.Caption = capt
    relojMouse
    frmCatManDiv.Show 1
Case 8
    frmCatManEmpr.Caption = capt
    relojMouse
    frmCatManEmpr.Show 1
Case 9
    frmCatManEnc.Caption = capt
    relojMouse
    frmCatManEnc.Show 1
Case 10
    frmCatManEpd.Caption = capt
    relojMouse
    frmCatManEpd.Show 1
Case 11
    frmCatManPats.Caption = capt
    relojMouse
    frmCatManPats.Show 1
```

```
    Case 12
        frmCatManPla.Caption = capt
        relojMouse
        frmCatManPla.Show 1
    Case 13
        frmCatManPol.Caption = capt
        relojMouse
        frmCatManPol.Show 1
    Case 14
        frmCatManRam.Caption = capt
        relojMouse
        frmCatManRam.Show 1
    Case 15
        frmCatManTar.Caption = capt
        relojMouse
        frmCatManTar.Show 1
    End Select
End Sub
```

La opción de *"Reportes"* dentro de Catálogos se diseñó para que el usuario tuviera la flexibilidad en la construcción de criterios que le permitiera ver sólo aquellos datos que en un momento dado quisiera.



Una vez especificadas las condiciones de búsqueda, la aplicación construye un *query* que incluye las cláusulas *Select, From* y *Where* y que se pasa directamente a la base de datos para que, en caso de existir, regrese el conjunto de registros resultado en el reporte.

Código del botón *"Generar Reporte"*.

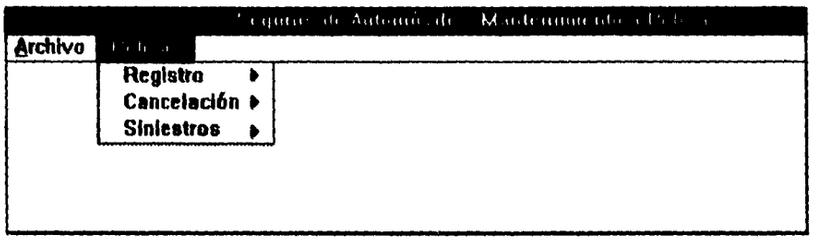
```
Sub cmdGenReport_Click ()
    Dim dbConnect As String,  formula As String
    Dim ruta As String, sql As String, sMsj As String, nl As String, Tabla As
String,
    Dim destino As Integer, RengAfec As Integer, n As Integer
    dbConnect = "ODBC;DSN=CONN_AUTOS;UID=;PWD="
```

```
nl = Chr(13) + Chr(10)
Select Case cboCatalogos.ListIndex
 Case 0
  ruta = app.Path + "\area.rpt"
  Tabla = "VehAreas"
 Case 1
  ruta = app.Path + "\auxili.rpt"
  Tabla = "VehAuxiliares"
 Case 2
  ruta = app.Path + "\ccostos.rpt"
  Tabla = "VehCtoCos"
 Case 3
  ruta = app.Path + "\cuenta.rpt"
  Tabla = "VehCuentas"
 Case 4
  ruta = app.Path + "\depend.rpt"
  Tabla = "VehDependencias"
 Case 5
  ruta = app.Path + "\divisi.rpt"
  Tabla = "VehDivisiones"
 Case 6
  ruta = app.Path + "\empresa.rpt"
  Tabla = "VehEmpresas"
 Case 7
  ruta = app.Path + "\encarga.rpt"
  Tabla = "VehEncargados"
 Case 8
  ruta = app.Path + "\empleado.rpt"
  Tabla = "VehEmpleados"
 Case 9
  ruta = app.Path + "\paramet.rpt"
  Tabla = "VehParametros"
 Case 10
  ruta = app.Path + "\plaza.rpt"
  Tabla = "VehPlazas"
 Case 11
  ruta = app.Path + "\polizas.rpt"
  Tabla = "VehPolizas"
 Case 12
  ruta = app.Path + "\ramos.rpt"
  Tabla = "VehRamos"
 Case 13
  ruta = app.Path + "\tarifa.rpt"
  Tabla = "VehTarifas"
End Select
If OptImp Then
    destino = 1
ElseIf OptPant Then
    destino = 0
End If
sMsj = "Trayendo los datos del Servidor. Favor de esperar..."
mEstatus = sMsj
DoEvents
relojMouse
On Error GoTo errMantCat
RptMantCat.ReportFileName = ruta
If lstCriterio.ListCount = 1 Then
   formula = "{" & Tabla & "." & Mid$(lstCriterio.List(0), 2,
   Len(lstCriterio.List(0)))
   RptMantCat.SelectionFormula = formula
ElseIf lstCriterio.ListCount > 1 Then
   formula = "{" & Tabla & "." & Mid$(lstCriterio.List(0), 2,
   Len(lstCriterio.List(0)))
   For n = 1 To (lstCriterio.ListCount - 1)
        formula = formula & " " & "and" & " " & "{" & Tabla & "." &
        Mid$(lstCriterio.List(n), 2, Len(lstCriterio.List(n)))
   Next
```

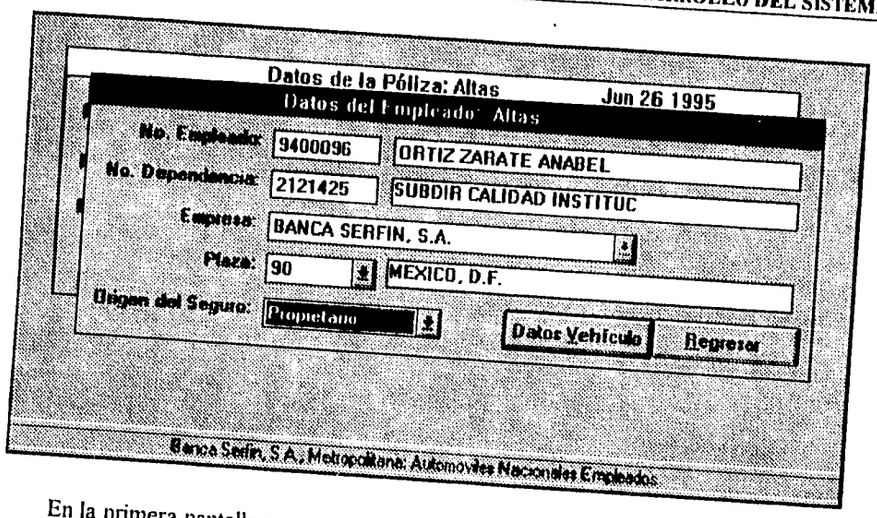
```
      RptMantCat.SelectionFormula = formula
    End If
    RptMantCat.Connect = dbConnect
    RptMantCat.Destination = destino
    If destino = 0 Then      'Ventana de impresión.
        frmVehCatMain.ScaleMode = 3
        RptMantCat.WindowWidth = frmVehCatMain.ScaleWidth
        RptMantCat.WindowHeight = frmVehCatMain.ScaleHeight
        RptMantCat.WindowTop = 0
        RptMantCat.WindowLeft = 0
        frmVehCatMain.ScaleMode = 1
    End If
    RptMantCat.Formulas(0) = "Hora= " + Chr(34) + Time$ + Chr(34)
    RptMantCat.Action = 1
    mEstatus = ""
    resetMouse
    GoTo endMantCat
errMantCat:
        resetMouse
        showError
        mEstatus = ""
        Resume endMantCat
endMantCat:
End Sub
```

Módulo de Mantenimiento a Pólizas

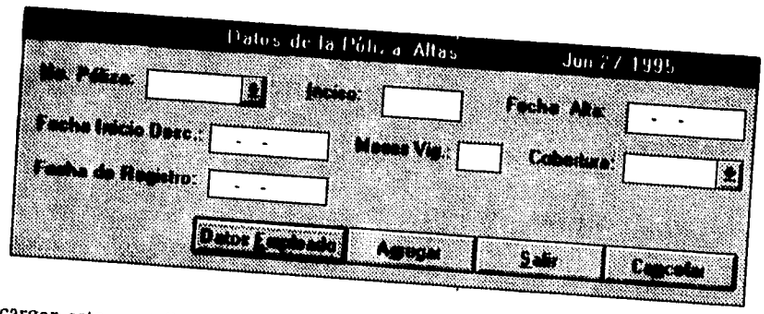
La pantalla principal de este módulo tiene el siguiente diseño: un menú de *"Pólizas"* el cual muestra las tres opciones que ofrece el sistema para el mantenimiento de pólizas: *"Registro"*, *"Cancelación"* y *"Siniestros"*. Dentro de cada opción es posible realizar: altas, bajas, modificaciones, consultas, emisión de comunicados y de reportes.



Mediante la opción *"Registro"* se da mantenimiento a las pólizas de la vigencia. En este módulo se tuvo que hacer un diseño que facilitará al usuario la captura de los datos de la póliza dado que la cantidad de éstos es considerablemente grande. El diseño para este módulo fue presentar al usuario 3 pantallas en forma de cascada.



En la primera pantalla se capturan los datos generales de la póliza: número de póliza, inciso, fecha de alta de la póliza, fecha de inicio de descuento de nómina, fecha de registro, meses de vigencia y tipo de cobertura. Se tiene un botón con el nombre de *"Datos del Empleado"* el cual permite el paso a la siguiente pantalla una vez que se haya elegido una póliza.



Al cargar esta pantalla se llena el combo de *"No.Poliza"* con las pólizas disponible para la Institución y al seleccionar una, en el campo *"Inciso"* se muestra el último inciso disponible para dicha póliza.

Código al cargar la forma de *"Datos de la Póliza"* para Registro de pólizas

```
Sub Form_Load ()
    Dim sEmp As String, sDiv As String, sRmo  As String
    Dim cPol As Integer
    FechaVeri = ""  'Inicializa variables
```

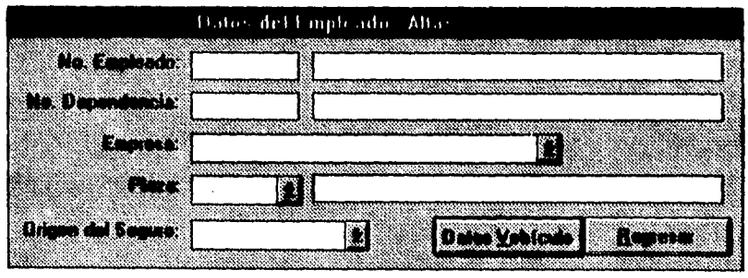
```
    CobVeri = ""
    VerFeDes = ""
    totEndosos = 0
    cmdRegistro.Enabled = False
    cmdRegistro.Enabled = False
    cmdDtsEmpleado.Enabled = False
    frmPolRegDP.cmdCancelar.Enabled = False
    ' LLenado del combo de Tipo de Cobertura
    cboCobertura.AddItem "Amplia"
    cboCobertura.AddItem "Limitada"
    cboCobertura.ListIndex = -1
    'Llenado del Combo de Pólizas
    On Error GoTo OKerr
    relojmouse
    Set dsPol = dbVehiculos.CreateDynaset("execute VehTraePol" )
    DB_SQLPASSTHROUGH)
    If dsPol.EOF And dsPol.BOF = True Then
        MsgBox "No hay pólizas dadas de alta.", MB_ICONSTOP
    Else
        'Calcula los renglones en dsPol:
        dsPol.MoveLast
        cPol = dsPol.RecordCount
        ReDim aPol(cPol)
        dsPol.MoveFirst
        Do While Not dsPol.EOF
            cboPol.AddItem dsPol.Fields(0)
            'Mete en el arreglo el nombre de la Póliza: Empresa[+División]+Ramo
            sEmp = dsPol.Fields("NomEmpresa")
            sDiv = IIf(IsNull(dsPol.Fields("NomDivision")), "",
           dsPol.Fields("NomDivision"))
            sRmo = dsPol.Fields("NomRamo")
            aPol(cboPol.ListCount - 1) = MayusculaInicial(sEmp) + ", " +
           MayusculaInicial(sDiv) + ": " + MayusculaInicial(sRmo)
            dsPol.MoveNext
        Loop
    End If
    resetmouse
    dsPol.Close
    cboPol.ListIndex = -1
    indCboPol = -1
    GoTo Okend
OKerr:
    resetmouse
    If Not dsPol Is Nothing Then dsPol.Close
    showError
    Resume Okend
Okend:
End Sub
```

Búsca el máximo número de inciso para la póliza elegida en el caso de altas

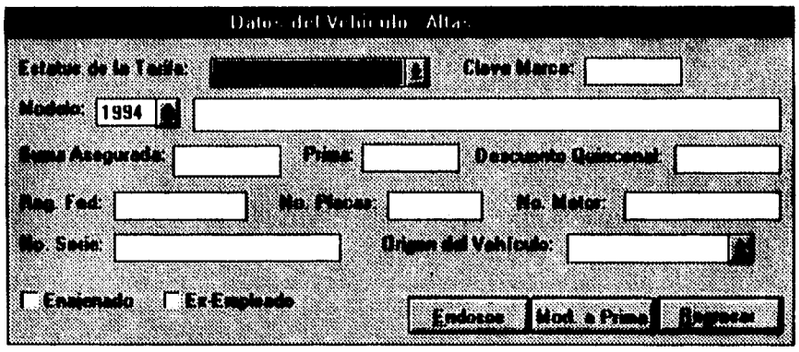
```
Sub traeMaxInciso ()
    Dim sql As String
    Dim dsRamo  As dynaset
    indCboPol = cboPol.ListIndex
    mEstatus = aPol(cboPol.ListIndex)
    traeStsPertenencia
    relojmouse
    On Error GoTo errRamo
    sql = " select NumInciso = Max(NumInciso + 1) from  VehVigencia  where
    NumPoliza = '" + cboPol + "' "
    Set dsRamo = dbVehiculos.CreateDynaset(sql, DB_SQLPASSTHROUGH)
    If IsNull(dsRamo.Fields("NumInciso").Value) Then
        txtNumInciso.Text = "1"
    Else
        txtNumInciso.Text = dsRamo.Fields("NumInciso").Value
    End If
```

```
    dsRamo.Close
    cmdRegistro.Default = True
    resetmouse
    GoTo finRamo
errRamo:
    resetmouse
    If Not dsRamo Is Nothing Then dsRamo.Close
    showError
    Resume finRamo
finRamo:
End Sub
```

La pantalla siguiente permite capturar los datos del propietario del vehículo: número de empleado, nombre del propietario, número de dependencia, empresa, plaza y origen del seguro. Esta pantalla tiene 2 botones, uno con el nombre de *"Datos del Vehiculo"* que permite el paso a la pantalla de *"Datos del Vehiculo"* y otro con el nombre de *"Regresar"* que al seleccionarlo se regresa a la pantalla anterior.

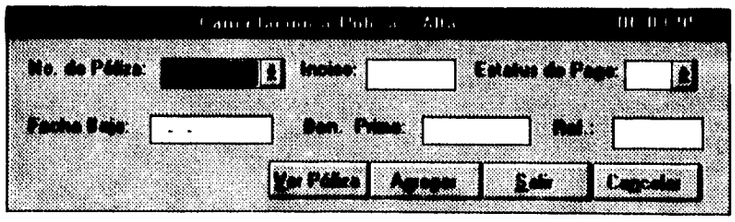


En la última pantalla se capturan los datos del vehículo: estatus de la tarifa, clave de marca, modelo, descripción del vehículo, suma asegurada, registro federal de automóviles, número de placas, número de motor, número de serie, origen del vehículo, si fue enajenado y si pertenece a un ex-empleado; la prima y el descuento quincenal los genera el sistema dependiendo del la fecha de alta, el tipo de cobertura y el tipo de vehículo . Esta última pantalla cuenta con un botón que al seleccionarlo nos manda a otra pantalla en la que se capturan los endosos de la póliza si así se desea.



Estas ventanas son de uso múltiple y en el título indicará la función que está realizando: altas, bajas, consultas o modificaciones. Después que el usuario del sistema captura todos los datos de las pantallas anteriores, se elige el botón de *"Agregar"*, *"Borrar" o "Actualizar"* según sea el caso, de la primera pantalla que es el sitio en donde la aplicación procede a válidar que los datos estén completos y, en caso de que ésta resulte exitosa, se ingresa a la base de datos de donde se toma para ser agrupada con otras pólizas y pasar a Pagos y Bonificaciones. En caso de encontrarse errores en la validación, la póliza *no* se incorpora a la base de datos.

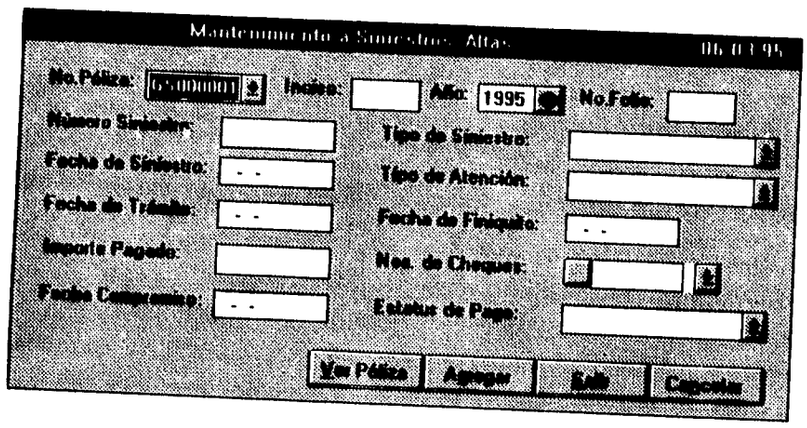
La opción de *"Cancelación"* se utiliza para eliminar del registro de pólizas vigentes a aquellas cuya baja se ha solicitado explícitamente. Al registrar una cancelación hace la transferencia del registro de pólizas vigentes al de bajas y le agrega datos tales como la fecha del sistema al momento de registrar la baja, la fecha de baja de la póliza, la prima a bonificar y el número de referencia. En el diseño de está pantalla se contemplo la necesidad de ver los datos de la póliza, por medio del botón de *"Ver Póliza"* se hace la conexión con Registro de Pólizas.



Valida la fecha de cancelación de la póliza y ejecuta el store procedure para calcular la prima a bonificar de cancelaciones.

```
Sub FecValCan (fecha As Control)
    Dim sMsj, CRLF, Fec, mes, meses, sp As String
    Dim nMes As Integer
    Dim dsVig As dynaset
    Dim dsPrima As dynaset
    CRLF = Chr(13) + Chr(10)
    meses = "JanFebMarAprMayJunJulAugSepOctNovDec"
    nMes = Val(Mid(fecha, 4, 2))
    If nMes > 0 And nMes < 13 Then
        mes = Mid(meses, Val(Mid(fecha, 4, 2)) * 3 - 2, 3)
        Fec = Format(mes + " " + Left(fecha, 2) + " " + Right(fecha, 4),
        "mmm dd yyyy")
    Else
        Fec = ""
    End If
    If Not IsDate(Fec) Then
        sMsj = fecha + ": No es una Fecha Válida." + CRLF
        sMsj = sMsj + "Introduzca la Fecha en Formato dd-mm-aaaa"
        MsgBox sMsj, MB_ICONEXCLAMATION
        fecha.SetFocus
    Else
         If Right$(Fec, 4) < "1900" Then
            MsgBox "Fecha Inválida", MB_ICONEXCLAMATION
            fecha.SetFocus
         Else
            sp = "execute VehTraeFecVal " & "'" & cboPol & "'" & "," &
            txtNumInciso & "," & "'" & Fec & "'"
            On Error GoTo errFecha
            Set dsVig = dbVehiculos.CreateDynaset(sp, DB_SQLPASSTHROUGH)
               If dsVig("valor") = 1 Then
                   MsgBox "Fecha fuera de vigencia para la Póliza",
                   MB_ICONEXCLAMATION
                   fecha.SetFocus
               Else
                   sp = "execute VehTraePrimBon " & "'" & FeIniPol &
                   "','" & Fec & "'," & Importe
                   Set dsPrima = dbVehiculos.CreateDynaset(sp,
                   DB_SQLPASSTHROUGH)
                   txtPrimaBon = dsPrima.Fields(0)
                   dsPrima.Close
               End If
            resetmouse
            dsVig.Close
        End If
     End If
     GoTo finfecha
errFecha:
     resetmouse
     If Not dsVig Is Nothing Then dsVig.Close
     If Not dsPrima Is Nothing Then dsPrima.Close
     showError
     Resume finfecha
finfecha:
End Sub
```

La opción de *"Siniestros"* permite registrar los siniestros de las pólizas vigentes. Al registrar un siniestro hace una transferencia de datos del registro de pólizas vigentes al de siniestros actualizándolo con datos tales como el número de folio, el número de siniestro, la fecha de siniestro, la fecha de finiquito, la fecha trámite, el tipo de atención, el tipo de siniestro, los números de cheques y el importe. Si son pérdidas totales, se registra el siniestro y posteriormente se registra la cancelación y se bonifica el resto de la prima.



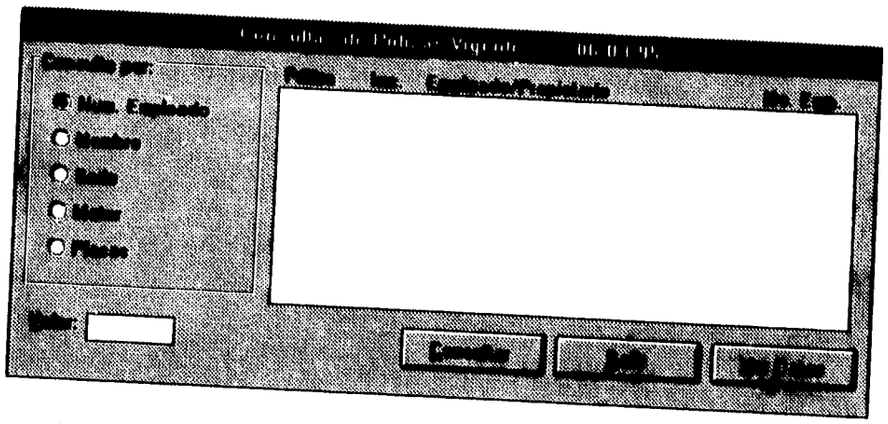
Desde aquí también se pueden ver los datos de la póliza por medio del botón *"Ver Póliza"*.

Valida que el inciso exista para la póliza seleccionada o no este cancelada y si es así trae el total de siniestros para la Póliza e Inciso.

```
Sub txtNumInciso_LostFocus ()
    Dim dsInciso As dynaset
    Dim sql As String, poliza As String, Inciso As String
    Inciso = txtNumInciso.Text
    poliza = cboPol.Text
    If txtNumInciso <> "" Then
        sql = "select FecInicio, StsVigencia, TotSiniestros from VehVigencia
        where NumPoliza = '" + poliza + "' and NumInciso  = " + Inciso + ""
        On Error GoTo OkIncerr
        relojmouse
        Set dsInciso = dbVehiculos.CreateDynaset(sql, DB_SQLPASSTHROUGH)
        If dsInciso.BOF = True And dsInciso.EOF = True Then
            resetmouse
            MsgBox "No Existe el inciso para la Póliza", MB_ICONEXCLAMATION
            txtNumInciso = ""
        Else
            Vigencia = dsInciso.Fields(1)
            total = dsInciso.Fields(2)
            If Vigencia <> "1" Then
                MsgBox "La Póliza está cancelada", MB_ICONEXCLAMATION
                cmdDaPol = False
```

```
                cmdSiniestro.Enabled = False
                ChecaVigencia = "1"
            End If
        End If
        dsInciso.Close
    End If
    resetmouse
    GoTo OkIncend
OkIncerr:
        resetmouse
        If Not dsInciso Is Nothing Then dsInciso.Close
        showError
        Resume OkIncend
OkIncend:
End Sub
```

En el caso de que se requieran consultar los datos de vehículos asegurados, de los cuales no se conozcan el número de póliza y el número de inciso, es posible localizarlos utilizando distintos criterios de búsqueda. Para ello se diseño la pantalla de *"Consultas Generales"*, la cual permite hacer consultas por el número de empleado, nombre del propietario, número de serie del vehículo, número de motor o el número de placas Solo se tiene que elegir una opción del diálogo *"Consulta por"* y proporcionar su valor.



A continuación se muestra el código del botón *"Consultar"* que arma el *query* para la consulta y hace la petición al servidor.

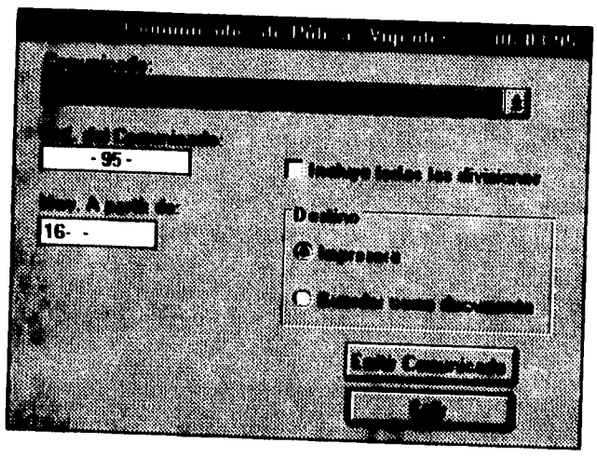
Código para ejecutar la consulta

```
Sub cmdGenConsulta_Click ()
    Static arrAgrega(4)
    ReDim tabs(1 To 3) As Integer
    Dim dsConsulta As dynaset
    Dim i As Integer, x  As Integer, j As Integer, totPol As Integer, r As
    Integer
```

```
Dim sql As String, poliza As String, condicion As String, tb As String
Dim Nombre As String, rng As String
'Llena un arreglo con el nombre de los campos en la tabla
arrAgrega(0) = "NumEmpleado"
arrAgrega(1) = "NomTitular"
arrAgrega(2) = "NumSerie"
arrAgrega(3) = "NumMotor"
arrAgrega(4) = "NumPlacas"
'Define tabs para la lista
tabs(1) = 40
tabs(2) = 65
tabs(3) = 215
r = SendMessage(lstCriterio.hWnd, LB_SETTABSTOPS, 0, 0&)
r = SendMessage(lstCriterio.hWnd, LB_SETTABSTOPS, 3, tabs(1))
tb = Chr(9)
lstCriterio.Clear
If txtValor = "" Then
    MsgBox "Debe Proporcionar la condicion .", MB_ICONEXCLAMATION
    Exit Sub
End If
' Forma la condición para la consulta
For i = 0 To 4
    If Op3Condicion(i) Then condicion = arrAgrega(i)
Next i
If Op3Condicion(0) Then
    'condicion = condicion & " = " & txtValor & " and StsVigencia = " &
    Status
    sql = "select NumPoliza, NumInciso, NumEmpleado, "
    sql = sql & "NomTitular "
    sql = sql & "from VehVigencia "
    sql = sql & "where NumEmpleado = " & txtValor & " and "
    sql = sql & "      StsVigencia = " & Status ElseIf Op3Condicion(1) Then
    sql = "execute VehTraePolEmpl " & "'%" & txtValor & "%'," & Status
ElseIf Op3Condicion(2) Or Op3Condicion(3) Or Op3Condicion(4) Then
    condicion = condicion & " = '" & txtValor & "'" & " and StsVigencia = "
    & Status
    sql = "select NumPoliza, NumInciso, NumEmpleado, NomTitular from
    VehVigencia where " & condicion
End If
mEstatus = "Buscando información en el Servidor. Favor de esperar..."
relojmouse
On Error GoTo OKerrConsul
Set dsConsulta = dbVehiculos.CreateDynaset(sql, DB_SQLPASSTHROUGH)
mEstatus = ""
If dsConsulta.EOF And dsConsulta.BOF = True Then
   resetmouse
   MsgBox "No existen Pólizas para la condicion especificada.",
   MB_ICONEXCLAMATION
Else
    dsConsulta.MoveLast
    totPol = dsConsulta.RecordCount
    dsConsulta.MoveFirst
    If totPol < 20 Then
        'Llena la lista
        For x = 0 To totPol - 1
            rng = dsConsulta("NumPoliza") & tb & dsConsulta("NumInciso")
            rng = rng & tb & mayusculaInicial(Left(dsConsulta("NomTitular"),
            40)) & tb
            rng = rng & dsConsulta("NumEmpleado")
            lstCriterio.AddItem rng                    ' adicionar a la lista.
            dsConsulta.MoveNext
        Next x
        cmdAbrir.Visible = True
    ElseIf totPol > 20 Then
        MsgBox "Favor de especificar más el criterio de búsqueda.",
        MB_ICONEXCLAMATION
    End If
```

```
    End If
    dsConsulta.Close
    resetmouse
    GoTo OkendConsul
OKerrConsul:
    resetmouse
    If Not dsConsulta Is Nothing Then dsConsulta.Close
    showError
    Resume OkendConsul
OkendConsul:
End Sub
```

La pantalla de *"Comunicados"* sirve para emitir los distintos comunicados que envia la Gerencia de Seguros de Automóviles. Dependiendo de dónde se invoque (Registro, cancelación o Siniestros) llenará el combo de *"Comunicado"* con la descripción de los comunicados existentes para dicha opción. Esta pantalla tiene la opción de elegir entre guardar el documento o mandarlo a imprimir.



Para el desarrollo de esta pantalla se utilizó OLE (Object Linking and Embedding) para conectar al Sistema con Word. A continuación se muestra el código para arrancar word que se encuentra en el evento form_load de esta pantalla. Posteriormente se encuentra el código que emite el comunicado.

Código del evento form_load para arrancar word

```
Sub Form_Load ()
    Dim sql As String, pathWord As String, sMsj As String, nl As String
    Dim cComun As Integer, i As Integer, x As Integer
    nl = Chr(13) + Chr(10)
    nomMes(1) = "enero"
    nomMes(2) = "febrero"
```

```
    nomMes(3) = "marzo"
    nomMes(4) = "abril"
    nomMes(5) = "mayo"
    nomMes(6) = "junio"
    nomMes(7) = "julio"
    nomMes(8) = "agosto"
    nomMes(9) = "septiembre"
    nomMes(10) = "octubre"
    nomMes(11) = "noviembre"
    nomMes(12) = "diciembre"
    txtNumRefb = Right(Format(Now, "dd-mm-yyyy"), 2)
    txtAnio = Right$(Format$(Now, "dd-mm-yyyy"), 4)
    relojmouse
    On Error GoTo errComun
    cdlArchSalida.InitDir = app.Path
    'Arranca Word si no está corriendo:
    If findWindow("OpusApp", NILL) = 0 Then
         pathWord = String$(255, 0)
         x = GetPrivateProfileString("Extensions", ByVal "doc", "", pathWord,
         255, "WIN.INI")
         If pathWord = "" Then
             sMsj = "No se encontró el programa Word." + nl
             sMsj = sMsj + "No es posible emitir los comunicados."
             MsgBox sMsj, MB_ICONSTOP
             frmMantMain.BackColor = &H80000005
             Unload Me
         End If
         pathWord = Left(pathWord, InStr(pathWord, " ") - 1)
         mEstatus = "Arrancando Microsoft Word. Favor de esperar..."
         x = Shell(pathWord, 6)
         DoEvents
         mEstatus = ""
     End If
     resetmouse
     GoTo finComun
errComun:
    showError
    Resume finComun
finComun:
End Sub
```

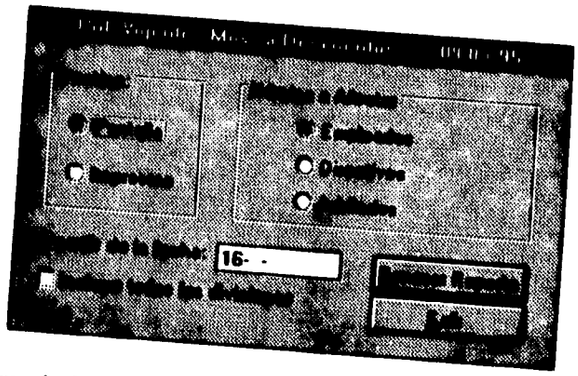
Genera el comunicado de envío de documentación a la compañía de seguros

```
Sub emiteAlEnSoCS ()
    Dim wordObj As object
    Dim dsPol As dynaset, dsVig As dynaset, dsFec As dynaset
    Dim sSql As String, sPol As String, nomDiv As String, nomRamo As String
    Dim nl As String, tb As String, FecOpe As String, numRef As String
    Dim nFol As Integer, sumInc As Integer, totInc As Integer, r As Integer
    Dim sql As String, numRefa As String, Comunicado As String, fecha As String
    numRefa = txtNumRefa + txtNumRefb + txtNumRefc
    Comunicado = cboComunicado.ItemData(cboComunicado.ListIndex)
    fecha = Format(Now, "mmm dd yyyy")
    On Error GoTo errEmitir
    If findWindow("OpusApp", NILL) = 0 Then
        MsgBox "No está activo Word. No es posible continuar.", MB_ICONSTOP
    ElseIf cboNumRef.ListIndex < 0 Then
        MsgBox "Debe elegir un número de Referencia.", MB_ICONEXCLAMATION
    ElseIf txtNumFolio = "" Then
        MsgBox "Debe indicar el Número de Folio inicial.", MB_ICONEXCLAMATION
    Else
        relojmouse
        numRef = Mid(cboNumRef, 15, 3)+Mid(cboNumRef, 19, 2)+Right(cboNumRef, 4)
        sSql = "execute VehTraePolRef '" + numRef + "'"
        Set dsPol = dbVehiculos.CreateDynaset(sSql, DB_SQLPASSTHROUGH)
        sSql = "execute VehTraeFecVig"
```

```
        Set dsVig = dbVehiculos.CreateDynaset(sSql, DB_SQLPASSTHROUGH)
        sSql = "execute VehTraeFecOpe '" + numRef + "'"
        Set dsFec = dbVehiculos.CreateDynaset(sSql, DB_SQLPASSTHROUGH)
        If dsPol.EOF And dsPol.BOF Then
            MsgBox "No hay datos de esa referencia.", MB_ICONEXCLAMATION
        ElseIf dsVig.EOF And dsVig.BOF Then
            MsgBox "No está registrada la fecha de Inicio de la Vigencia",
            MB_ICONEXCLAMATION
        ElseIf dsVig.EOF And dsVig.BOF Then
            MsgBox "No está registrada la fecha de envio del Comunicado.",
            MB_ICONEXCLAMATION
        Else
            'Usa el objeto expuesto por Word:
            Set wordObj = CreateObject("Word.Basic")
            DoEvents
            wordObj.ArchivoAbrir app.Path + "\" + "alensocs.doc", , , , , , , , 1
            DoEvents
            FecOpe = "México, D.F. a " + Trim(dsFec("Dia")) + " de "
            FecOpe = FecOpe + nomMes(Val(dsFec("Mes"))) + " de " + dsFec("Ano")
            wordObj.EdiciónReemplazar"<FECHA>",FecOpe,0,1,1,0, , , True,0,0
            wordObj.EdiciónReemplazar"<REFERENCIA>",Mid(cboNumRef,15),0,1,1,0, ,
            ,True, 0, 0
            wordObj.EdiciónReemplazar "<VIGENCIA>", dsVig("Rango"),0,1,1,0, ,
            , True, 0, 0
            sPol = ""
            nFol = Val(txtNumFolio)
            tb = Chr(9)
            nl = Chr(13)
            Do While Not dsPol.EOF
                nomDiv = dsPol("NomDivision")
                nomDiv = nomDiv + Space(30 - Len(nomDiv))
                totInc = Val(dsPol("TotIncisos"))
                nomRamo = Left(Trim(dsPol("NomRamo")), 12)
                nomRamo = nomRamo + Space(12 - Len(nomRamo))
                sPol = sPol + nomDiv + tb + dsPol("NumPoliza") + tb
                sPol = sPol + nomRamo + tb + justnum(nFol, 6) + tb
                sPol = sPol + justnum(nFol + totInc - 1, 6) + tb +
                justnum(totInc, 6) + nl
                dsPol.MoveNext
                nFol = nFol + totInc + 1
                sumInc = sumInc + totInc
            Loop
            wordObj.EdiciónBuscar "<POLIZAS>"
            wordObj.Insertar sPol
            wordObj.EdiciónReemplazar "<SUMA>", justnum(sumInc, 6),0,1,1,0, ,
            , True, 0, 0
            If op3Destino(0) Then
                wordObj.ArchivoImprimir
                wordObj.ArchivoGuardarComo app.Path + "\doctemp.doc"
                MsgBox "Se terminó de Imprimir.", MB_ICONEXCLAMATION
            Else
                cdlArchSalida.Action = 2        'Save As...
                wordObj.ArchivoGuardarComo cdlArchSalida.Filename
            End If
            sql = "execute VehInsertarComu " & "'" & numRefa & "'," & Comunicado
            & ",'" & fecha                & "'"
            r = dbVehiculos.ExecuteSQL(sql)
            wordObj.ArchivoCerrarTodo 2
        End If
        dsPol.Close
        dsVig.Close
        dsFec.Close
    End If
    resetmouse
    GoTo finEmitir
errEmitir:
resetmouse
```

```
    If Err = 32755 Then
        If findWindow("opusApp", NILL) <> 0 Then
            wordObj.ArchivoGuardarComo app.Path + "\doctemp.doc"
            wordObj.ArchivoCerrarTodo 2      '"Save As" fué cancelado.
        End If
    Else
        If Not dsPol Is Nothing Then dsPol.Close
        If Not dsVig Is Nothing Then dsVig.Close
        If Not dsFec Is Nothing Then dsFec.Close
        showError
    End If
    Resume finEmitir
finEmitir:
End Sub
```

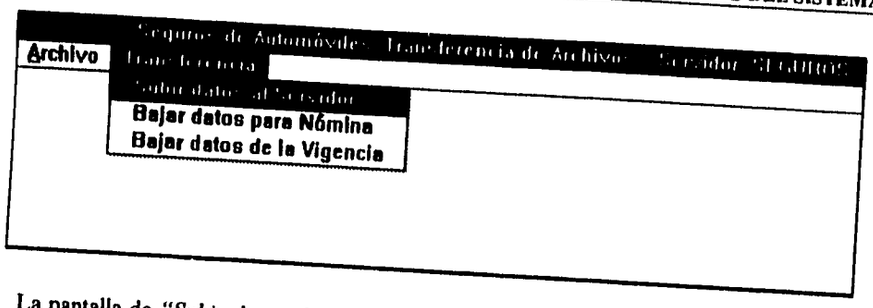
La opción de *"Reportes"* sirve para emitir reportes relativos al registro, cancelación o siniestros de pólizas.



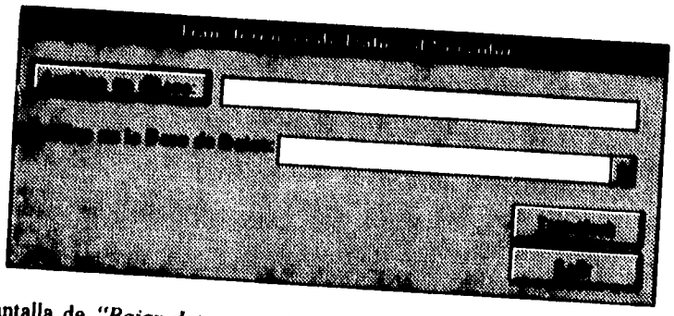
Módulo de Transferencia de Información

Este módulo del sistema se diseñó para que el usuario pueda integrar a la base de datos la información de las pólizas de manera amigable que envía el corredor a la Gerencia de Seguros; también desde éste modulo se graban disquetes con información de altas y bajas de pólizas para enviar al área de nómina y permite respaldar la información de la base de datos.

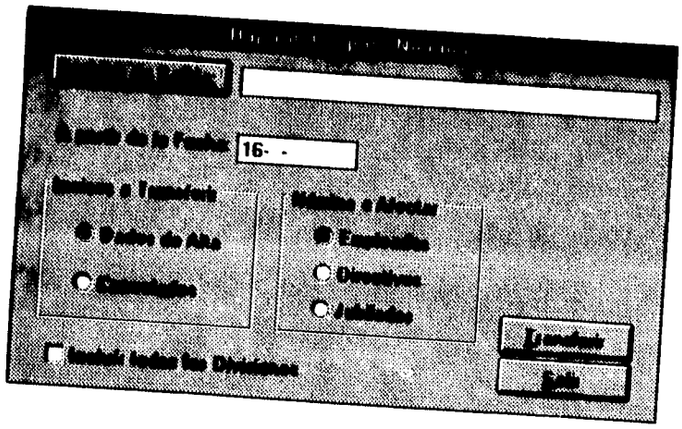
La pantalla principal tiene un menú con las opciones *"Subir datos al Servidor"*, *"Bajar datos para Nómina"* y *"Bajar datos de la Vigencia"*.



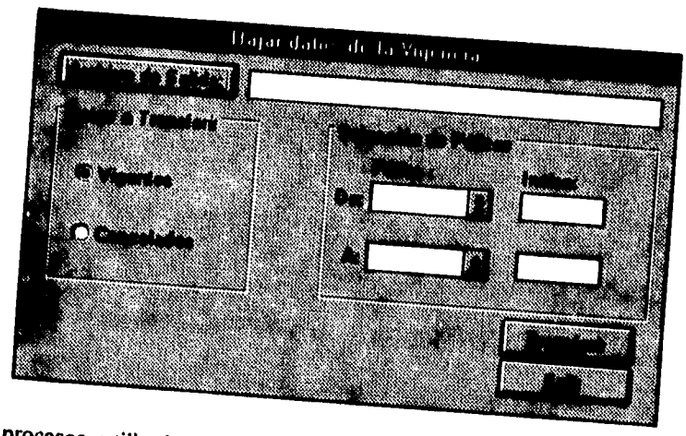
La pantalla de *"Subir datos al Servidor"* sirve para integrar la información que envía el corredor de seguros o el área de nómina a la Gerencia de Seguros de Automóviles. Esta pantalla pide el nombre del archivo por medio de un cuadro de dialógo y el destino de éste en la base de datos.



La pantalla de *"Bajar datos para Nómina"* graba la información referente a pólizas dadas de alta o canceladas a partir de cierta fecha y tipo de nómina a un archivo. Estos archivos serán enviados al Area de Nómina.



La pantalla *"Bajar datos de la Vigencia "* permite respaldar información en la base de datos.



Los procesos utilizados para estas pantallas se hicieron por medio de *stores procedures* debido a que son procesos robustos. Enseguida se muestra el store procedure que integra a la base datos la información de renovación de pólizas.

Integra la información a la base de datos

```
create procedure VehIntegra @tabla tinyint as
/* Verifica que haya datos en VehParametros: */
if (select max(Valor) from VehParametros) = ''
begin
raiserror 50000 '>>No se han capturado los Parámetros para la Vigencia.'
return
end
if (select Valor from VehParametros where Nombre='FecIniVig') = '' or
(select count(*) from VehParametros where Nombre='FecIniVig') = 0
begin
raiserror 50000 '>> Falta la Fecha de Inicio de Vigencia en el Catálogo de
Parámetros.'
return
end
if (select Valor from VehParametros where Nombre='TotMssVig') = '' or
(select count(*) from VehParametros where Nombre='TotMssVig') = 0
begin
raiserror 50000 '>> Faltan los Meses de Vigencia en el Catálogo de Parámetros.'
return
end
if (select Valor from VehParametros where Nombre='IVA') = '' or
(select count(*) from VehParametros where Nombre='IVA') = 0
begin
raiserror 50000 '>> Falta el IVA en el Catálogo de Parámetros.'
return
end
if @tabla = 0
begin                          /* A R E A S */
begin transaction
```

```
delete VehAreas
insert into VehAreas
select convert(tinyint,AREA),1,NOMAREA
from AREAS
if @@error != 0
begin
rollback transaction
end
delete AREAS
commit transaction
end
if @tabla = 1                    /* A U X I L I A R E S */
begin
begin transaction
/* Pasa los datos de AUXILIARES a VehAuxiliares. */
delete VehAuxiliares
insert VehAuxiliares
select distinct convert(int,substring(CUENTA,1,4)),
convert(int,substring(CUENTA,6,2)),
convert(int,substring(CUENTA,9,2)),
convert(tinyint,substring(CUENTA,14,2)),
convert(int,substring(AUXI,1,8)),
NOMCTA,convert(int,CENTROCOST)
from AUXILIARES
if @@error != 0
begin
rollback transaction
end
delete AUXILIARES
commit transaction
end
if @tabla = 2                    /* C U E N T A S */
begin
begin transaction
/* Verifica que haya datos en VehAuxiliares: */
if (select count(*) from VehAuxiliares) = 0
begin
raiserror 50000 '>>El catálogo de Auxiliares debe contener datos.'
commit transaction
return
end
/* Agrega los niveles contables de las cuentas usadas en VehAuxiliares */
delete VehCuentas
insert VehCuentas
select distinct convert(smallint,c.CTA),convert(tinyint,c.SCTA),
convert(tinyint,c.SSCTA),convert(tinyint,c.SSSCTA),
c.DESCRIP
from CUENTAS c, VehAuxiliares a
where convert(smallint,c.CTA) = a.Cuenta
if @@error != 0
begin
rollback transaction
end
delete CUENTAS
commit transaction
end
if @tabla = 3                    /* D E P E N D E N C I A S */
begin
begin transaction
/* Transfiere los datos de los departamentos de EMPLEADOS y de
DEPENDENCIAS (la DIRECCION) a VehDependencias */
/* Verifica que haya datos en EMPLEADOS: */
if (select count(*) from EMPLEADOS) = 0
begin
raiserror 50000 '>>Debe transferir primero el Catálogo de Empleados.'
commit transaction
return
```

```
end
delete VehDependencias
insert VehDependencias
select convert(int,a.DEPEN), NOM_DEPEN, DIRECCION, convert(smallint,PLAZA)
from EMPLEADOS a, DEPENDENCIAS b
where a.DEPEN *= b.DEPEN and
convert(smallint,a.PLAZA) = 90
group by a.DEPEN,NOM_DEPEN,DIRECCION,PLAZA
if @@error != 0
begin
rollback transaction
end
delete DEPENDENCIAS
commit transaction
end
if @tabla = 4                    /* D I V I S I O N E S */
begin
begin transaction
/* Transfiere los datos a VehDivisiones de DIVISIONES y EMPLEADOS */
/* Verifica que haya datos en EMPLEADOS: */
if (select count(*) from EMPLEADOS) = 0
begin
raiserror 50000 '>>Debe transferir primero el Catálogo de Empleados.'
commit transaction
return
end
delete VehDivisiones
insert VehDivisiones
select convert(smallint,d.DIVISION), d.NOMDIV,0
from DIVISIONES d, EMPLEADOS e
where d.DIVISION = e.DIVISION
group by d.DIVISION, NOMDIV
if @@error != 0
begin
rollback transaction
end
delete DIVISIONES
commit transaction
end
if @tabla = 5                    /* E M P L E A D O S */
begin
begin transaction
/* Pasa los datos de los empleados de EMPLEADOS a VehEmpleados,
separando la columna nombre en ApePaterno, ApeMaterno y Nombre */
delete VehEmpleados
insert VehEmpleados
select convert(int,NUM_EMPL),convert(int,DEPEN),1,
ltrim(substring(NOMBRE,1,charindex('/',NOMBRE)-1)),
ltrim(substring(NOMBRE,charindex('/',NOMBRE)+1,charindex('/',
substring(NOMBRE,charindex('/',NOMBRE)+1,99))-1)),
ltrim(rtrim(substring(NOMBRE,charindex('/',substring(NOMBRE,
charindex('/',NOMBRE)+1,99))+charindex('/',NOMBRE)+1,99)))
from EMPLEADOS
if @@error != 0
begin
rollback transaction
return
end
commit transaction
/* Le elimina un número que traen al final algunos nombres y que al
parecer significa que el empleado proviene de una fusión. */
update VehEmpleados
set Nombre = substring(Nombre,1,datalength(Nombre)-1)
where right(Nombre,1)  between '1' and '9'
end
/*
if @tabla = 6                    /* E N C A R G A D O S */
```

```
begin
begin transaction
/* Verifica que haya datos en VehPlazas: */
if (select count(*) from VehPlazas) = 0
begin
raiserror 50000 '>>El catálogo de Plazas no contiene datos.'
commit transaction
return
end
delete VehEncargados
insert VehEncargados
select NumPlaza,NOMBRE,DEPTO,0
from ENCARGADOS,VehPlazas
where PLAZA = NomPlaza
if @@error != 0
begin
rollback transaction
end
delete ENCARGADOS
commit transaction
end
*/
if @tabla = 6                    /* P L A Z A S */
begin
begin transaction
/* Transfiere los datos a VehPlazas de PLAZAS y EMPLEADOS (los
números de DIVISION y AREA). */
/* Verifica que haya datos en EMPLEADOS: */
if (select count(*) from EMPLEADOS) = 0
begin
raiserror 50000 '>>Debe transferir primero el Catálogo de Empleados.'
commit transaction
return
end
delete VehPlazas
insert VehPlazas
select convert(smallint,p.PLAZA), p.NOMPLAZA, convert(smallint,e.DIVISION),
convert(tinyint,e.AREA)
from PLAZAS p, EMPLEADOS e
where p.PLAZA = e.PLAZA
group by p.PLAZA,p.NOMPLAZA,e.DIVISION,e.AREA
if @@error != 0
begin
rollback transaction
end
delete PLAZAS
commit transaction
end
if @tabla = 7                    /* P O L I Z A S */
begin
begin transaction
/* Le mete a VehPolizas los datos que vienen en POLIZAS */
delete VehPolizas
insert VehPolizas
select POLIZA_NVA,convert(smallint,DIVISION),0,0
from POLIZAS
if @@error != 0
begin
rollback transaction
return
end
commit transaction
/* Define el número de empresa: 1 para división < 90 */
update VehPolizas
set NumEmpresa = 1
where NumDivision<90
/* Empresa >1 para división > 90 */
```

```
update VehPolizas
set NumEmpresa = NumDivision - 90
where NumDivision>=90
/* Genera el número de ramo a partir de su descripción: */
/* Tipo de Vehículo: Automóviles,Motocicletas,Camionetas.*/
update VehPolizas
set NumRamo=NumSubRamo*100
from VehPolizas,POLIZAS,VehSubRamos
where NumPoliza = POLIZA_NVA and
RAMO like '%'+TpoVehiculo+'%'
update VehPolizas
set NumRamo=100
where NumRamo <100
/* Origen:Nacionales,Fronterizos,Antiguos y Clásicos,Contado */
update VehPolizas
set NumRamo=NumRamo+NumSubRamo*10
from VehPolizas,POLIZAS,VehSubRamos
where NumPoliza = POLIZA_NVA and
RAMO like '%'+Origen+'%' and
RAMO not like '%'+'CONT'+'%'
update VehPolizas
set NumRamo=NumRamo+40
from VehPolizas,POLIZAS
where NumPoliza = POLIZA_NVA and
RAMO like '%'+'CONT'+'%'
/* Propietario: Exempleados, Funcionarios, Directivos,... */
update VehPolizas
set NumRamo=NumRamo+NumSubRamo
from VehPolizas,POLIZAS,VehSubRamos
where NumPoliza = POLIZA_NVA and
RAMO like '%'+Propietario+'%'
/* R A M O S */
/* Genera la tabla VehRamos a partir de VehPolizas */
delete VehRamos
insert VehRamos
select distinct NumRamo,substring('AUTOMOVILES MOTOCICLETASCAMIONETAS  ',
(charindex(substring(str(NumRamo,3),1,1),'123')-1)*12+1,12),
substring('NACIONALES          FRONTERIZOS          ANTIGUOS Y CLASICOS
NACIONALES CONTADO  NACIONALES CONTADO  ',
(charindex(substring(str(NumRamo,3),2,1),'12345')-1)*20+1,20),
substring('EXEMPLEADOS FUNCIONARIOSDIRECTIVOS  JUBILADOS   EMPLEADOS   PERS.
AJENO UTILITARIOS ENAJENADOS  ',
(charindex(substring(str(NumRamo,3),3,1),'12345678')-1)*12+1,12)
from VehPolizas
delete POLIZAS
end
if @tabla in (8,9,10)          /* S U M A S   A S E G U R A D A S */
begin
   /* Verifica que no exista la clave AMIS en VehParametros: */
   declare @Id tinyint, @Nombre varchar(30), @Valor varchar(100), @DesParametro
varchar (100),
           @TpoVehiculo tinyint
   select @Id = (select isnull(max(Id),0) from VehParametros)
   select @Valor = (select max(AMIS) from SUMASG)
   if @tabla = 8
   begin
      select @Nombre = 'CveSumAsgAutomoviles'
      select @TpoVehiculo = 2
      select @DesParametro = 'Clave de suma asegurada para Autom_viles'
   end
   if @tabla = 9
   begin
      select @Nombre = 'CveSumAsgCamionetas'
      select @TpoVehiculo = 5
      select @DesParametro = 'Clave de suma asegurada para Camionetas'
   end
   if @tabla = 10
```

```
   begin
      select @Nombre = 'CveSumAsgMotocicletas'
      select @TpoVehiculo = 4
      select @DesParametro = 'Clave de suma asegurada para Motocicletas'
   end
   if not exists(select * from VehParametros where Nombre = @Nombre)
      insert VehParametros values(@Id+1,@Nombre,56,4,@Valor,@DesParametro)
   else
      update VehParametros
      set Valor = @Valor, DesParametro = @DesParametro
      where Nombre = @Nombre
   /* Inserta los datos de SUMASG en VehSumAsg */
   begin transaction
      delete VehSumAsg
      where TpoVehiculo=@TpoVehiculo
      insert VehSumAsg
      select
@TpoVehiculo,convert(float,SA),convert(float,PAMP),convert(float,PLIM)
      from SUMASG
      if @@error != 0
      begin
         rollback transaction
      end
      delete SUMASG
   commit transaction
end
if @tabla = 11                  /* T A R I F A S */
begin
begin transaction
/* Pasa los datos de TARIFAS a VehTarifas: */
delete VehTarifas
insert VehTarifas
select convert(int,CVEAMIS),convert(smallint,MODELO),
DESCRIPCIO,convert(float,SA),convert(float,AMPLIA),
convert(float,LIMITADA)
from TARIFAS
if @@error != 0
begin
rollback transaction
end
delete TARIFAS
commit transaction
end
if @tabla = 12                  /* C A N C E L A D O S */
begin
execute VehGenVigencia 2
/* Si hubo error hace "rollback" manualmente: */
if @@error != 0
begin
/* Borra los datos de la vigencia: */
delete VehVigencia
from VehVigencia,CANCELADOS
where NumPoliza = POLIZA and
NumInciso = convert(int,INCISO)
/* Borra datos de Cancelados: */
delete VehCancelados
from VehCancelados,CANCELADOS
where NumPoliza = POLIZA and
NumInciso = convert(int,INCISO)
/* borra datos de la vigencia anterior: */
delete VehVigAnt
from VehVigAnt,CANCELADOS
where NumPoliza = POLIZA and
NumInciso = convert(int,INCISO)
/* borra los datos de vehiculos fuera de la tabla de tarifas: */
delete VehSinTarifa
from VehSinTarifa,CANCELADOS
```

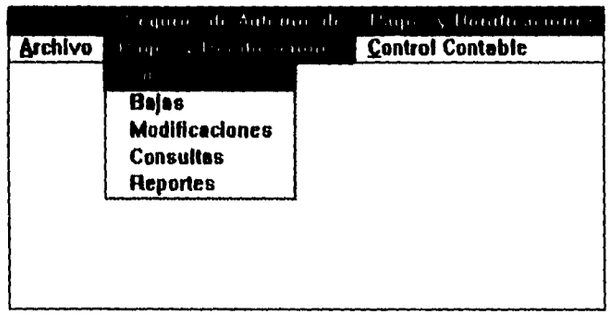
```
where NumPoliza = POLIZA and
NumInciso = convert(int,INCISO)
raiserror 50000 '>>Hubo error en la operación. Deberá reiniciarla.'
end
delete CANCELADOS
end
if @tabla = 13                    /* S I N I E S T R O S */
begin
begin transaction
/* Verifica que haya datos en VehVigencia: */
if (select count(*) from VehVigencia) = 0
begin
raiserror 50000 '>>No se tienen capturados datos de la Vigencia.'
commit transaction
return
end
/* Pasa los datos correspondientes de SINIESTROS a VehSiniestros
(Sólamente toma en cuenta datos de Incisos que aparezcan en
VehVigencia */
insert VehSiniestros
select POLIZA,convert(int,INCISO),convert(int,AÑO+FOLIO),
NO_SINIEST,convert(smalldatetime,FECSINIEST),convert(tinyint,TIPO_SINI),
CHARINDEX(TIPO_ATTE,' ATGMPDPTRP')/2,CHARINDEX(ESTADO,'FPS'),
convert(smalldatetime,FECHATRAM),convert(smalldatetime,FECHAFIN),
convert(float,IMPORTE),CHEQUES,convert(smalldatetime,FECHACOM)
from SINIESTROS a, VehVigencia b
where a.POLIZA = b.NumPoliza and
convert(int,INCISO) = b.NumInciso
if @@error != 0
begin
rollback transaction
delete SINIESTROS
return
end
commit transaction
/* Actualiza VehVigencia con el número de siniestros por inciso: */
select NumPoliza,NumInciso,CtaSiniestros=count(*)
into #ctaSiniestros
from VehSiniestros
group by NumPoliza,NumInciso
update VehVigencia
set TotSiniestros = CtaSiniestros
from VehVigencia v,#ctaSiniestros c
where v.NumPoliza = c.NumPoliza and
v.NumInciso = c.NumInciso
/* delete SINIESTROS */
end
if @tabla = 14                    /* V I G E N C I A */
begin
execute VehGenVigencia 1
/* Si hubo error hace "rollback" manualmente: */
if @@error != 0
begin
/* Borra los datos de la vigencia: */
delete VehVigencia
from VehVigencia,VIGENCIA
where NumPoliza = POLIZA and
NumInciso = convert(int,INCISO)
/* Borra datos de Vigencia anterior: */
delete VehVigAnt
from VehVigAnt,VIGENCIA
where NumPoliza = POLIZA and
NumInciso = convert(int,INCISO)
/* borra datos de Vehículos fuera de la tabla de tarifas: */
delete VehSinTarifa
from VehSinTarifa,VIGENCIA
where NumPoliza = POLIZA and
```

```
NumInciso = convert(int,INCISO)
raiserror 50000 '>>Hubo errores en la operación. Deberá repetirla.'
end
delete VIGENCIA
end
/*
if @tabla = 16                    /* C O N T A B L E */
begin
/* Verifica que haya datos en VehDivisiones: */
if (select count(*) from VehDivisiones) = 0
begin
raiserror 50000 '>>El Catálogo de Divisiones debe contener datos.'
commit transaction
return
end
begin transaction
/* primero pasa datos a la tabla de datos generales: */
insert VehContable
select distinct convert(smallint,CLAVE),convert(smalldatetime,FECSYS),
convert(smalldatetime,FECHA),convert(smallint,LOTE),
convert(int,CENTROCOST),0,CONCEPTO,0
from CONTABLE
where CLAVE != '' and
CLAVE != '.'
/* le agrega los números de división que encuentre dada la descripción */
update VehContable
set a.NumDivision = b.NumDivision
from VehContable a, VehDivisiones b, CONTABLE c
where a.NumOperacion = convert(smallint,c.CLAVE) and
c.AREA like '%'+b.NomDivision+'%'
/* ahora pasa el detalle a la tabla correspondiente */
insert VehDetCnt
select convert(smallint,CLAVE),
convert(smallint,substring(CUENTA,1,4)),
convert(tinyint,substring(CUENTA,6,2)),
convert(tinyint,substring(CUENTA,9,2)),
convert(tinyint,substring(CUENTA,14,2)),
convert(int,substring(AUXI,1,8)),
convert(tinyint,str(2-convert(float,CARGO)/(convert(float,CARGO)+
convert(float,ABONO)),1)),
convert(float,CARGO)+convert(float,ABONO)
from CONTABLE
where CLAVE != '' and
CLAVE != '.'
if @@error != 0
begin
rollback transaction
return
end
delete CONTABLE
commit transaction
end
*/
if @tabla = 15                    /* P A G O S   Y   B O N I F I C A C I O N E S
*/
/* Genera los números de concepto y actualiza la tabla de conceptos */
execute VehGenCncPyB
delete PAGBON
GO
```

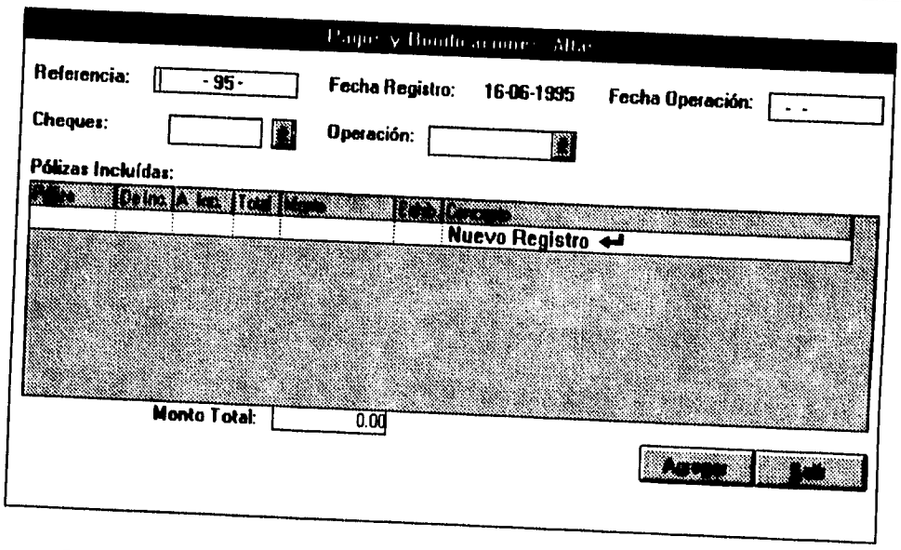
Módulo de Pagos y Bonificaciones.

Este módulo del sistema se utiliza para controlar el registro de pagos de primas y de bonificaciones en caso de bajas. En operación normal, el registro se genera como consecuencia de dar una alta o baja de vehículos. También desde este módulo se controla el registro de movimientos contables generados ya sea por altas (pagos) o bajas por cancelación (bonificaciones), y lleva un arrastre de saldos; los registros se generan automáticamente y son congruentes con la información en la tabla de pagos y bonificaciones.

La pantalla principal de este módulo contiene los menús *"Pagos y Bonificaciones"* y *"Control Contable"*. Las opciones que tienen son: *"Altas"*, *"Bajas"*, *"Modificaciones"*, *"Consultas"* y *"Reportes"*.



La pantalla de *"Pagos y Bonificaciones"* es de uso múltiple y en el título indicará la función que está realizando: altas, bajas, consultas o modificaciones



Esta pantalla está dividida en dos secciones en la primera se deben capturar los datos generales del pago o bonificación: número de referencia, fecha de registro, fecha de operación, cheques, operación (pago o bonificación). En la segunda el usuario solo debe de seleccionar las pólizas que incluirá el movimiento ya que se diseño usando un *grid* para la presentación de los datos y listas ocultas al usuario, que al dar un clic sobre la columna *"Póliza"* del grid presenta la lista con las pólizas y su descripción al seleccionar una, se proporciona en el grid el inciso inicial, el inciso final, el monto a pagar por el pago o bonificación. El usuario solo debe dar un clic en la columna *"Concepto"* del grid y se mostrará una lista con todos los conceptos de pago o bonificación.

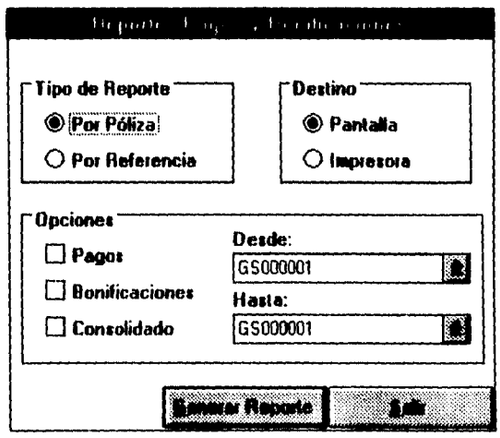
Trae los conceptos correspondientes al tipo de operación: pago o bonificación.

```
Sub traeConceptos (ByVal tpoOpe As Integer)
    Dim sCnc As String, sSql As String
    Dim dsCnc As Dynaset
    relojMouse
    On Error GoTo errConc
    If okIncisos <> tpoOpe Then 'Checa si no tenia datos la lista de conceptos.
        lstcap(1).Clear
        If baseDatos = ODB Then
            Set dsCnc = dbVehiculos.CreateDynaset("VehTraeCnc" + Str(tpoOpe),
            DB_SQLPASSTHROUGH)
        Else
            sSql = "select NumConcepto, NomConcepto "
            sSql = sSql + "from VehConceptos "
            sSql = sSql + "where TpoOperacion =" + Str(tpoOpe) + " "
            sSql = sSql + "order by NumConcepto"
```

```
            Set dsCnc = dbVehiculos.CreateDynaset(sSql)
        End If
        If dsCnc.EOF And dsCnc.BOF Then
            MsgBox "No hay Conceptos dados de alta.", MB_ICONEXCLAMATION
        Else
            Do While Not dsCnc.EOF
                sCnc = dsCnc("NomConcepto")
                lstcap(1).AddItem Left(sCnc, 1) + LCase(Mid(sCnc, 2))
                sCnc = dsCnc("NumConcepto")
                lstcap(1).ItemData(lstcap(1).NewIndex) = sCnc
                dsCnc.MoveNext
            Loop
        End If
        dsCnc.Close
    End If
    resetMouse
    grdPyB.Enabled = True
    GoTo concEnd
errConc:
    resetMouse
    showError
    If Not dsCnc Is Nothing Then dsCnc.Close
    Resume concEnd
concEnd:
End Sub
```

La pantalla de *"Reportes Pagos y Bonificaciones"* se diseñó de tal manera que pudieran existir reportes con diferentes combinaciones.

Por lo tanto se dividio en tres secciones *"Tipo de Reportes"* donde se indica si el reporte se listará por póliza o por referencia, *"Destino"* indica si el reporte se desea ver en la pantalla o se emita directamenta a la impresora y finalmente *"Opciones"* indica si el reporte se requiere de pagos, de bonificaciones o ambos y si el reporte será por póliza se tiene que indicar el rango de pólizas.



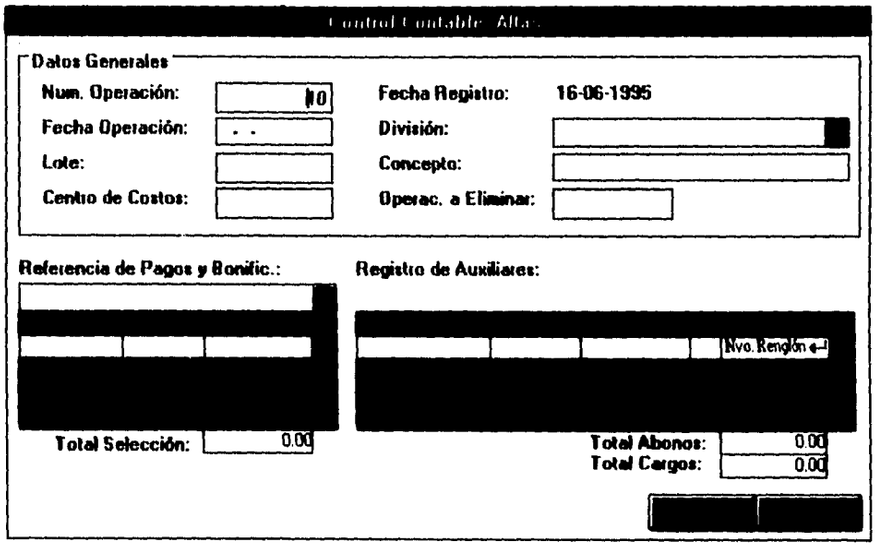
Genera reportes de pagos y bonificaciones

```
Sub cmdGenRpt_Click ()
    Dim r As Integer
    Dim dsIVA As Dynaset
    Dim sSql As String, nomrep As String, sMsj As String
    Dim limInf, limSup, fecIni, fecFin As String
    Dim iva As Single
    On Error GoTo errImpr
    'Revisa si se marcó alguna de las opciones de contenido:
    fecIni = ""
    fecFin = ""
    limInf = ""
    limSup = ""
    If chkContenido(0) = 0 And chkContenido(1) = 0 And chkConsolid = 0 Then
        MsgBox "Debe seleccionar Pagos, Bonificaciones o ambos.",
        MB_ICONEXCLAMATION
        chkContenido(0).SetFocus
    ElseIf chkConsolid = 1 And (cboEmpresas.ListIndex < 0 Or
    cboEmpresas.ListCount = 0) Then
        If cboEmpresas.ListIndex < 0 Then
            MsgBox "Debe seleccionar una empresa.", MB_ICONEXCLAMATION
            cboEmpresas.SetFocus
        Else
            MsgBox "No existen datos de empresas. No se puede emitir el
            reporte.", MB_ICONSTOP
        End If
    Else
        'Obtiene el valor del IVA
        Set dsIVA = dbVehiculos.CreateDynaset("execute VehTraeIVA",
        DB_SQLPASSTHROUGH)
        If dsIVA.EOF And dsIVA.BOF Then
            MsgBox "No existe el parámetro del IVA en la base de datos."
        Else
            iva = dsIVA("IVA")
            iva = iva / 100
            relojMouse
            sMsj = "Obteniendo la información de la Base de Datos."
            sMsj = sMsj + " Un momento por favor..."
            mEstatus = sMsj
            If chkConsolid = 0 Then      'No se eligió consolidado.
                If InStr(cboRango(0), "-") Then
                    limInf = Mid(cboRango(0), 15, 3) + Mid(cboRango(0), 19, 2) +
                    Right(cboRango(0),                                    4)
                    fecIni = cvtFecha(Left(cboRango(0), 10))
                Else
                    limInf = cboRango(0)
                End If
                If InStr(cboRango(1), "-") Then
                    limSup = Mid(cboRango(1), 15, 3) + Mid(cboRango(1), 19, 2) +
                    Right(cboRango(1),                                    4)
                    fecFin = cvtFecha(Left(cboRango(1), 10))
                Else
                    limSup = cboRango(1)
                End If
            End If
            sSql = "execute VehTraePagBon2 " + Str(optSelector(0) * -1) + "," +
            Str(chkConsolid)
            sSql = sSql+",'" + limInf + "','" + limSup + "','" + fecIni + "','"
            + fecFin + "',"
            If chkConsolid = 1 Then
                sSql = sSql + Str(cboEmpresas.ItemData(cboEmpresas.ListIndex))
            Else
                sSql = sSql + "0"
            End If
            r = dbVehiculos.ExecuteSQL(sSql)
```

```
            If r = 0 Then
                MsgBox"No hay registros en el rango elegido.",B_ICONEXCLAMATION
            Else
                nomrep = "rpb" + Trim(Str(optSelector(0) * -1)) +
                Trim(Str(chkConsolid))
                nomrep = nomrep + Trim(Str(chkContenido(0))) +
                Trim(Str(chkContenido(1))) + ".rpt"
                rptPagBon.ReportFileName = app.Path + "\" + nomrep
                'Si es a la pantalla el reporte:
                If optDestino(0) Then
                    frmPagBonMain.ScaleMode = 3
                    rptPagBon.WindowWidth = frmPagBonMain.ScaleWidth
                    rptPagBon.WindowHeight = frmPagBonMain.ScaleHeight
                    rptPagBon.WindowTop = 0
                    rptPagBon.WindowLeft = 0
                    frmPagBonMain.ScaleMode = 1
                    rptPagBon.Destination = 0
                Else
                    rptPagBon.Destination = 1
                End If
                'Fórmulas en los reportes, modificadas para que tomen el iva
                actual:
                rptPagBon.Formulas(0)="PmaNetaPagos= (VehImpPagBon.Pagos) / " +
                CStr(1 + iva)
                rptPagBon.Formulas(1) = "I.V.A.= (VehImpPagBon.Pagos) / " +
                CStr(1 + iva) + " * " +  CStr(iva)
                rptPagBon.Formulas(2) = "PmaNetaBon=
                (VehImpPagBon.Bonificaciones) / " + CStr(1+ iva)
                rptPagBon.Formulas(3) = "IVAbon= (VehImpPagBon.Bonificaciones) /
                " + CStr(1 + iva) + " * " + CStr(iva)
                'Fórmula para que escriba la hora ("feature" de Crystal)
                rptPagBon.Formulas(4) = "Hora= " + Chr(34) + Time$ + Chr(34)
                If chkConsolid = 1 Then
                    rptPagBon.Formulas(5) = "NomEmpresa= " + Chr(34) +
                    cboEmpresas + Chr(34)
                End If
                rptPagBon.Connect = dbVehiculos.Connect
                rptPagBon.Action = 1
            End If
            mEstatus = ""
        End If
        dsIVA.Close
    End If
    resetMouse
    GoTo finImpr
errImpr:
    resetMouse
    If Not dsIVA Is Nothing Then dsIVA.Close
    showError
    Resume finImpr
finImpr:
End sub
```

La pantalla de control contable, controla el registro de movimientos contables generados ya sea por altas (pagos), bajas por cancelación (bonificaciones), y lleva un arrastre de saldos. Esta pantalla se diseñó en tres secciones; en la primera sección se capturan los datos generales: número de operación, lote, centro de costos, fecha de registro, la división, el concepto y la operación a eliminar si es el caso, en la siguiente sección se debe elegir la referencia de pagos y bonificaciones a la que se va afectar y el sistema muestra las

pólizas para esa referencia y en la última sección se debe proporcionar el registro de auxiliares proporcionando la cuenta, el auxiliar, el saldo, indicar si es un abono o un cargo y el monto.



Código del control contable para actualizar la base de datos

```
Sub ejecutaAccion ()
    Dim sMsj As String, nl As String
    If okCaptura() Then
        sMsj = "¿Está seguro de que desea " + IIf(usoFrmCnt = 2, "borrar",
        "actualizar") + " los                datos?"
        If MsgBox(sMsj, MB_ICONEXCLAMATION + MB_YESNO) = IDYES Then
            nl = Chr(13) + Chr(10)
            relojMouse
            'Muestra un mensaje para esperar:
            mEstatus = "Actualizando la Base de Datos. Favor de esperar un
            momento."
            If usoFrmCnt = 1 Or usoFrmCnt = 3 Then  'Altas o Modificaciones
                actualizaDatos
            ElseIf usoFrmCnt = 2 Then               'Bajas
                borraDatos
            End If
            resetMouse
            mEstatus = ""
        End If
    End If
End Sub
'Borra los datos elegidos de la base de Datos.
Sub borraDatos ()
    Dim i As Integer, r As Integer
    Dim sSql As String, sMsj As String
    If okCaptura() Then   'Verifica si hay datos a borrar:
```

```
        On Error GoTo errBorrar
        sSql = "execute VehBorrContable " + txtCapNum(0)
        r = dbVehiculos.ExecuteSQL(sSql)
        If usoFrmCnt = 2 Then limpiaforma False
    Else
        MsgBox "No se ha incluido ninguna Operación.", MB_ICONEXCLAMATION
    End If
    GoTo finBorrar
errBorrar:
    resetMouse
    showError
    Resume finBorrar
finBorrar:
End Sub
```

CAPITULO VI

PRUEBAS E IMPLANTACION DEL SISTEMA

VI.1. Pruebas del Sistema

Para *asegurar la calidad* del sistema se realizaron pruebas para garantizar que éste se desempeña en forma adecuada y que cumple con sus requerimientos para esto se aplicaron diferentes estrategias.

Se desarrollaron casos para probar las rutas de los programas, también se hicieron pruebas tomando en cuenta las especificaciones que señalan lo que el programa debe hacer y cómo lo debe llevar acabo bajo ciertas condiciones.

Primeramente se aplicaron pruebas parciales al sistema centrándose en los módulos independientes entre sí para localizar los errores dentro de ese único módulo. Posteriormente se probo la integración de cada módulo en el sistema para ver la compatibilidad de los módulos.

Se le aplicó al sistema *prueba de carga máxima* activando todas las máquinas al mismo tiempo y operando con los procesos más exigentes, *prueba del tiempo de ejecución* determinando el tiempo de máquina que el sistema necesita para procesar los datos de las transacciones más robustas, *prueba de procedimientos* determinando la claridad de la documentación en lo aspectos de operación y uso del sistema, haciendo que el usuario lleve a cabo exactamente lo que el manual y la ayuda pide.

Para las pruebas del sistema se utilizaron *datos artificiales* para probar todas las combinaciones de formatos y valores, para lo cual se formuló un plan de prueba con un equipo de personas distintas de las que escribieron los programas, estas usaron las especificaciones del sistema.

También se hicieron pruebas con *datos reales* que los usuarios proporcionaron de sus actividades habituales para probar parcialmente el sistema. Estos datos mostraron cómo funciona el mismo para los requerimientos típicos de procesamiento; pero se considero que los datos reales no brindan una verdadera prueba del sistema ya que ignora los casos más probables que causan la falla del mismo

VI.2. Implantación del Sistema

La instalación de la red y el equipo de computo necesitó de una planificación y preparación amplia de las instalaciones. El equipo y el software se ordenó al Grupo Timón con 4 meses de anticipación antes de la implantación del sistema.

Antes de la llegada del equipo, se solicitó al área de soporte técnico su apoyo para preparar las instalaciones y la conexión de la red.

Se hicieron dos tipos de capacitación para familiarizar al personal con los detalles del nuevo sistema. Los operadores y los responsables de la gerencia recibieron dicha capacitación para su uso.

Para familiarizar a los usuarios con el sistema, recibieron capacitación del ambiente *Windows* y *Microsoft Office* que ofreció el Grupo Timón por la compra del equipo, este curso tuvo una duración de 25 horas. Posteriormente se les dio el entrenamiento del uso del sistema el cual fue dirigido por sus diseñadores.

Se estudiaron y probaron los procedimientos de captura de datos, se prepararon ejemplos de reportes y comunicados. Se puso énfasis en las validaciones y el uso del sistema en procesos especiales. Los usuarios emplearon mucho tiempo en pruebas capturando su información y consultándola.

Surgieron muchas preguntas durante la etapa de capacitación, dando la oportunidad de discutir dudas y proponer sugerencias libremente. Se hicieron algunos ajustes según las recomendaciones del personal usuario. Las reuniones fueron productivas.

Para la conversión al nuevo sistema se contó con el apoyo del subgerente del área usuaria quién incluso ayudó a la introducción de los registros reales.

La información que se incorporó a la base de datos fue la siguiente: los datos de los catálogos, datos de pólizas vigentes, datos de pólizas canceladas y los datos de los siniestros de pólizas. Se trabajó en forma paralela con el sistema anterior durante 3 meses.

La implantación se planeó bien y todo se desarrolló sin contratiempos. El sistema se ha estado usando por 3 meses aproximadamente con éxito. Se está en espera de la renovación de las pólizas para la revisión final de éste.

Todo indica que el nuevo sistema ha sido bien aceptado por los usuarios y que cumple con los objetivos que se plantearon al inicio del estudio.

CONCLUSIONES

La presente tesis, una vez llevada a la práctica, ha permitido a la empresa financiera para la que se desarrolló aprovechar en gran parte los recursos de cómputo que poseía, mejorar el servicio a los clientes y la satisfacción y productividad del personal.

Con la aplicación de la teoría del modelo *Cliente/Servidor* se cristalizaron las ventajas que brinda. El cliente proporciona al usuario final del sistema un ambiente gráfico, interface estándar fácil de usar por los iconos y ventanas desplegables.

Las facilidades que *Visual Basic* proporciona para utilizar *OLE (Object Linking and Embedding)* compartiendo características entre el Sistema y la paqueteria de *Windows*, fueron rápidamente asimiladas por los usuarios del sistema facilitando su trabajo y mejorando su desempeño.

Se pudo incorporar una nueva tecnología, aunque se corrió el riesgo derivado de la innovación que hasta ahora muy pocas empresas del sector han decidido probar ya que siguen estando muy amarradas a la arquitectura de mainframe.

El desempeño del nuevo sistema superó las expectativas del usuario, logrando la automatización de todos los procesos que intervienen en la administración de seguros para automóviles, tales como: la solicitud de pólizas a la compañía, el seguimiento de trámites de siniestros, de las cobranzas y el control de vencimiento de las pólizas para su renovación; evitando al máximo los errores humanos.

Por otra parte, el sistema favorece la libertad de operación del usuario respecto a la compañía de seguros, al no depender de ésta para disponer de la información en el momento que se requiera.

El desarrollo de este sistema evidenció que para cumplir con los objetivos cabalmente no basta la aplicación de la técnica, es necesario también considerar las repercusiones que un nuevo sistema tiene en el factor humano, tales como: resistencia al cambio y manejo de conflictos. El analista además de tener la habilidad para modelar un sistema de información y entender las necesidades del usuario, deberá ser capaz de superar los problemas humanos que puedan ser un obstáculo para la realización del sistema.

La tecnología *Cliente/Servidor* ha hecho posible algunas de las promesas que se han hecho a largo de los últimos años a desarrolladores y usuarios finales. Su estudio es necesario hacerlo considerando las ventajas de los *Sistemas Abiertos* pues tratándose de un caso de *Proceso Distribuido Cooperativo* éste es su medio ambiente natural.

A pesar del poderío que ofrece la tecnología *Cliente/Servidor* no es el fin de la evolución de los modelos de computación distribuida, sino un paso intermedio entre los sistemas centralizados del pasado y la tendencia a la distribución total. Lo importante es que esta tecnología satisface las necesidades de información de los negocios en la actualidad.

BIBLIOGRAFIA

- *Biblioteca de la informática Vol. 3.*
Editorial Limusa, S.A. de C.V. 1990.

- *The ODBC driver Catalog*
Affinity Publishing Inc. 1993.

- *Personal Computing México.*
Stefora, Ann; Moller, Mike.
Las soluciones Cliente Servidor;
El Middleweare: tecnología que facilita la transición C/S.
Año 6 No. 75 Agosto 1994

- *Open Client/Server.*
Morris, Henry
Digital Journal. Agosto, 1993.

- *Business Reingeneering with Technology. An implementation* Guide.
Donovan, Jhon.
Techology Group. 1993.

- *Computer World México.*
Alvarez Martínez, Adrián.
3-Tier, otra opción en la arquitectura Cliente/Servidor.
IDG
Año 14. No. 396. 21 de Marzo 1994.

- *PC Semanal.*
García Konieeczny, Juan.
Dos notas acerca de redes.
Año 2. No. 124 Vol. 5, 3 de Octubre de 1994.

- *Red*
Wolf de Córdoba, Heidi.
Los GUI's invaden las redes corportaivas.
Año 4. No. 34. 1994.

- *El libro de Windows NT.*
Custer, Helen.
Microsoft Co.-McGraw-Hill.1994.

- *Redes de Computadoras.*
Black, Uyless.
Macrobit-Rama. 1990.

- *Biblioteca De La Informática Vol. 3.*
Editorial Limusa, S.A. De C.V. 1990.

- *Microsoft.*
Relational Database Design .
Student Work.
Microsoft Corporation 1991.

- *Fundamentos De Base De Datos.*
Henry F. Korth.
Abraham Silberschatz.
Mc Graw Hill.

- *Database Developer's Guide.*
Roger Jennings.
Sams Publishing.

- *Análisis Y Diseño De Sistemas.*
James A. Senn.
Mc Graw Hill.

- *Visual Basic How-To.*
Zane Thomas.
Robert Arnson.
Mitchell Waite.
Waite Wroup Press.

- *Ms Developer Network.*
Library January 1995.
Microsoft.

- *Análisis y Diseño de Sistemas de Información.*
James A. Senn
McGraw Hill. 1994.