4

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

"BASES DE DATOS: BASE DE DATOS PARA LA DISTRIBUCION DE PEDIDOS DE ARTICULOS PROMOCIONALES EN LA INSTITUCION BANCARIA"

TRABAJO DE SEMINARIO

QUE PARA OBTENER EL TITULO DE

LICENCIADA EN INFORMATICA

P R E S E N T A

MONICA BEATRIZ GARCIA SOTRES

ASESOR:

ING. VICTOR HUGO ARROYO HERNANDEZ

CUAUTITLAN IZCALLI, EDO. DE MEX. 2000

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN

UNIDAD DE LA ADMINISTRACION ESCOLAR

DEPARTAMENTO DE EXAMENES PROFESIONALES

DR. JUAN ANTONIO MONTARAZ CRESPO
DIRECTOR DE LA FES CUAUTITLAN
P R E S E N T E

ATN: Q. Ma. del Carmen García Mijares
Jefe del Departamento de Exámenes
Profesionales de la FES Cuautitlán

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

Bases de Datos: Base de Datos para la Distribución de Pedidos de Artículos Promocionales en la Institución Bancaria

que presenta la pasante: Mónica Beatriz García Sotres

con número de cuenta: 9102825-9 para obtener el título de

Licenciada en Informática

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

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
Cuautitlán Izcalli, Méx. a 24 de Octubre de 2000

MODULO	PROFESOR	FIRMA
I	Ing. Víctor Hugo Arroyo Hernández	
II	MC. Araceli Nivon Zaghi	
IV	Lic. Carlos Pineda Muñoz	

A mis padres,

Como eterno agradecimiento a mi existencia, valores y formación que me han dado con su esfuerzo, sacrificio y ejemplo, permitiéndome llegar a esta etapa de mi vida.

A mis hermanos,

Que con su apoyo y comprensión me han ayudado a no detenerme a lo largo del camino, impulsándome a seguir adelante en mi vida profesional y personal a pesar de las adversidades.

A mi familia,

Por todas las palabras de aliento que me han brindado y todos los grandiosos y felices momentos que me han permitido pasar en su compañía.

A Rubén,

Por todo su amor, alegría y optimismo con que me ha enseñado a vivir, y por estar ahí respaldándome siempre en cada decisión y paso que doy.

Doy Gracias a Dios por hacerme parte de su creación permitiéndome así ser parte de todos ustedes y por encaminar mis pasos con todo su amor.

Gracias...

INDICE

INDICE DE FIGURAS

INTRODUCCIÓN

Un sistema de manejo de bases de datos es la capa de software necesaria para la creación, manipulación y modificación de los datos que conforman una base de datos. Las tareas fundamentales de estos sistemas son el control de concurrencia de acceso a los datos, la seguridad de los datos para protección física y lógica y la eficiencia del sistema que normalmente se evalúa en términos del tiempo de respuesta en la evaluación de las consultas de los usuarios.

En torno de las bases de datos y sus sistemas de manejo se abren numerosos temas de investigación. Los modelos y lenguajes de manipulación de datos, la evaluación y optimización de consultas conforman tópicos fundamentales de investigación. Las bases de datos relacionales que son las de mayor interés para el presente trabajo, surgen de un artículo de Codd en 1970, etapa a partir de la cual es posible encontrarlas en todo centro de cómputo. Muchas aplicaciones y usuarios de bases de datos tienen a su alcance software para el manejo de bases de datos relacionales de diversas marcas comerciales, habiéndolos para computadoras personales, medianas y grandes, sin embargo son pocas las veces en que se reflexiona sobre cual es el motivo de su éxito y por qué se han popularizado tanto, en este trabajo se hace una revisión y se muestra que una de las causas principales es el modelo de datos matemático que respalda este tipo de bases de datos lo que las ha hecho tan comunes.

La línea de investigación en sistemas de bases de datos tiene un impacto social que de alguna forma es indirecto puesto que los grandes proyectos tanto en las ramas sociales, científicas y comerciales son articulados en torno de los sistemas de información implementados en sistemas manejadores de bases de datos En cuanto al impacto económico se tiene que este es directo en la medida que posibilita un incremento en la productividad de las empresas, ya que académica y comercialmente los sistemas manejadores de bases de datos se han consolidado como la solución para desarrollar los poderosos sistemas de información que las sociedades modernas necesitan. Este trabajo es por tanto, desarrollado con la finalidad de contar con los conocimientos teóricos necesarios para poder evaluar, mejorar o implementar sistemas de base de datos eficientes que faciliten

el desempeño del trabajo evitando así los registros en archiveros y permitiendo llevar un adecuado control de inventarios, nóminas, ventas, compras, etc. dentro de las empresas públicas y privadas, al mismo tiempo que proporcionan al usuario un control centralizado de sus datos que le permitirá tomar decisiones con seguridad.

Para estos fines el presente trabajo se ha estructurado en cuatro partes. El primer capítulo integrado con temas correspondientes a antecedentes, conceptos básicos, ventajas y estructura de una base de datos. El segundo capítulo es el que nos proveerá de las bases necesarias para comprender lo que es un sistema gestor de base de datos así como sus lenguajes y modelos existentes. Un tercer capítulo que explicará el modelo relacional y la arquitectura cliente/servidor que son el modelo y plataforma sobre la que se basará el desarrollo del caso práctico que se expondrá en el último capítulo.

Objetivo General:

Adquirir los conocimientos fundamentales para implementar una base de datos relacional que solucione la problemática existente en la Institución Bancaria para la distribución de pedidos de los artículos promocionales que se venden.

Objetivos Particulares:

Obtener mejores tiempos de respuesta y atención al cliente por medio de la solución propuesta.

Contar con una herramienta confiable y amigable para el usuario final que le permita indicar al tarjetahabiente la situación de su pedido.

CAPITULO 1 . INTRODUCCION A LAS BASES DE DATOS

1.1 Antecedentes de las bases de datos

En un inicio los sistemas de información que existían estaban basados en múltiples ficheros debido a que se creaban ficheros específicos para una determinada aplicación Este tipo de sistemas traía como consecuencia la carga y repetición de trabajo, ya que los datos se recolectaban varias veces por estar repetidos en cada uno de los archivos en los que se requería de ellos para el correcto funcionamiento de la aplicación.

Debido a que las aplicaciones eran independientes unas de otras y los datos no podían transferirse entre ellas comenzó a existir gran redundancia en los datos, lo cual originó grandes divergencias en los resultados. Este tipo de inconsistencia se debía en gran medida a que era imposible la actualización de todos y cada uno de los datos en los diversos archivos de forma simultánea.

En ese momento se deduce la necesidad de una gestión más racional del conjunto de datos, en donde los datos se organicen y se almacenen en una estructura que permita satisfacer las necesidades de información de todas las aplicaciones y de la organización misma.

Así pues, en 1963 aparece la expresión de base de datos, en un simposio que tuvo lugar en Santa Mónica (EEUU), sin embargo, la definición que se le dio a la expresión Data Base, no fue aceptada universalmente y es hasta 1967 cuando el grupo de estandarización Codasyl[1] cambia la denominación de está expresión por la de Data Base Task Group.

A pesar de que las Bases de Datos surgen de la necesidad de mejorar la calidad, consistencia y veracidad de la información emitida en las empresas para la toma de decisiones en el manejo de estas, su uso ha tenido mayor auge que en sus inicios debido a la gran evolución tecnológica por la que pasamos día a día. Un claro ejemplo de esto es el crecimiento de la infraestructura en telecomunicaciones provocando que el flujo y manejo de información demande el uso de bases de datos

1.2 Ventajas de las bases de datos

El surgimiento de las bases de datos además de mejorar la calidad de las prestaciones de los sistemas informáticos y aumentar sus rendimientos, presentan grandes ventajas frente a los sistemas de información basadas en ficheros como son entre otras las siguientes:

- Proporcionar un control centralizado de los datos de operación de la empresa.

- Independencia de los datos.

  Proveer de independencia de datos es un objetivo esencial de los sistemas de bases de datos. Esta implica el hecho de que las aplicaciones no dependan de ninguna estructura de almacenamiento o de alguna estrategia de acceso en particular, es decir, la inclusión de nueva información, la desaparición de otras y los cambios en la estructura física no deben ocasionar que se alteren los programas.

  La independencia de datos es la capacidad que se tiene de modificar la definición de esquema en un nivel del sistema de base de datos sin que afecte la definición del esquema del nivel inmediato superior. En la independencia de datos se encuentran dos tipos que son:

  La independencia física de datos, que es poder modificar el esquema interno(físico) sin tener que alterar el esquema conceptual, esto implica que al hacer alguna modificación al esquema interno no se tengan que volver a escribir los programas de aplicación que usan la base de datos. En ocasiones en necesario modificar el esquema interno para reorganizar ciertos archivos físicos o para mejorar el funcionamiento. Este tipo de independencia es más fácil de lograr que la independencia lógica de los datos.

  La independencia lógica de datos, es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos y no tener que realizar modificación alguna a los programas de aplicación que hacen uso de la base de datos. Las modificaciones que

[1] Conference on Data System Languages

se hacen en el esquema conceptual, se da regularmente para ampliar o reducir la base de datos, es decir, cuando se altera la estructura lógica de la base de datos.

Cuando el sistema gestor de base de datos cuenta con independencia lógica de datos, sólo es preciso modificar las definiciones de las vistas y las restricciones podrán ser modificadas en el esquema conceptual sin afectar los esquemas externos.

Como se puede observar la independencia de datos es ventajosa por diversas razones como las siguientes:

- Las modificaciones de las estructuras de almacenamiento físico no involucran modificaciones en los programas de aplicación.
- La introducción de nuevas tecnologías en las memorias auxiliares no afecta los programas de aplicación.
- Los datos son compartidos puesto que vistas diferentes pueden ser extraídas de la misma estructura de almacenamiento, por lo tanto la duplicación de datos se reduce
- La programación de las aplicaciones es facilitada dado que ella es realizada sobre vistas determinadas y que ella no está involucrada por las estructuras de almacenamiento físico ni por los problemas de acceso.
- La ejecución de operaciones ilícitas tales como la actualización o eliminación de datos que no pertenecen a una vista es fácilmente prohibida.

- Coherencia

El uso de las bases de datos implica que los datos que la constituyen sean recolectados y almacenados una sola vez, por esto cuando los usuarios manipulan los datos de la base utilizan los mismos datos, ocasionando así que los resultados de todos sean coherentes y comparables.

- Disponibilidad en los datos

Al encontrarse los datos almacenados en una estructura que no posee un dueño único, cada usuario deja de ser propietario de los datos, puesto que éstos se comparten entre el conjunto de aplicaciones, de tal manera que se facilita la disponibilidad de los datos para

todos aquellos usuarios que tienen necesidad de ellos. Además de permitir que las aplicaciones existentes puedan compartir los datos de la base de datos, es factible que las nuevas aplicaciones que se desarrollen puedan operar con los mismos datos almacenados sin necesidad de crear nuevos archivos almacenados.

- Mejor y más normalizada documentación de la información.

  Como la misma base de datos incluye la semántica de los datos, los estándares pueden ser reformados. Con un control central de la base de datos, el Administrador de Base de Datos puede asegurar que todos los estándares aplicables serán seguidos en la representación de la base de datos.

  La estandarización de los formatos de datos almacenados es particularmente deseable como una ayuda para el intercambio de datos o migración entre sistemas.

- Reducción en el espacio de almacenamiento.

  El dejar de almacenar los datos repetidas veces en los distintos archivos que eran necesarios para cada aplicación se genera la desaparición de las redundancias lo cual lleva en los sistemas de base de datos a una menor ocupación de almacenamiento

1.3 Conceptos de bases de datos

Hoy en día, hablar de Base de Datos es referirse a un activo de singular importancia en cualquier organización; por lo tanto, el objetivo de toda organización no es sólo tener una base de datos, sino una base de datos correctamente diseñada y administrada, de forma tal que brinde una información correcta para la dinámica de la organización, en el tiempo adecuado y a la persona conveniente.

Por otro lado, el proceso de conversión a la tecnología de base de datos requiere de una gran inversión para toda organización, tanto en el software de base así como en las herramientas de desarrollo y por ende la capacitación del personal responsable de su aplicación y operatividad; es por esta razón, que este proceso debe ser planificado y administrado en forma cuidadosa.

Definición de Base de Datos

Una Base de Datos sirve para registrar hechos o acontecimientos que tienen lugar en la vida de un organismo, de modo que los registros hechos sean recuperables cuando sean necesarios o para cuando se desee extraer conclusiones mediante la combinación de varios hechos elementales, permitiendo además la centralización, integración y difusión de la información registrada.

Se define a una Base de Datos como el conjunto de datos relacionados a los que se puede acceder y manipular obteniendo con ello la información necesaria para un fin determinado El uso de una base de datos permite obtener información necesaria para el manejo, control y administración de una empresa o institución.

Características de una base de datos

Las principales características de una Base de Datos son:

- Se conforma por un conjunto de datos lógicamente coherente con cierto significado inherente.
- Representa algún aspecto del mundo real.
- Los datos están relacionados y estructurados.
- La redundancia es controlada, ya que no debe existir redundancia lógica pero es permitible que exista cierta redundancia física.
- Existe la independencia tanto física como lógica de los datos y de los procesos.
- Soporta a múltiples usuarios y a diferentes aplicaciones.
- La actualización y recuperación de datos debe asegurar Integridad, Seguridad y Confidencialidad de los datos

La descripción y manipulación de la base de datos se apoya en un modelo de datos, esta definición o descripción del conjunto de datos que se encuentran contenidos en la base son únicas y están integradas con los mismos datos La definición de la base de datos se

denomina estructura o esquema de la base de datos. En las bases de datos la descripción y documentación, se almacenan junto con los datos (diccionario de datos) de modo que estos están autodocumentados, es por eso que muchas veces se dice que las bases de datos son auto-descriptivas[2].

La descripción de la estructura de la base de datos es parte de la misma base de datos. A esta descripción se le llama metadatos[3]. Los metadatos suelen almacenarse en forma de tablas que son llamadas tablas de sistema. Guardar los metadatos en tablas es eficiente para el sistema gestor de base de datos pero también es conveniente para el usuario de la base de datos pues una consulta en los metadatos se hace de la misma manera en que se consultan las tablas de los datos.

Cuando nos referimos a Base de Datos, estamos haciendo referencia a un conjunto de conceptos que manejamos cotidianamente y que están fuertemente ligados al concepto mismo de Base de Datos. Así tenemos que:

- El término Base de Datos involucra un conocimiento profundo y detallado de la realidad referida a la organización.
- La Base de Datos debe ser la representación más fidedigna de la organización.
- Los datos que representan la organización deberán ser utilizados de manera coordinada e integral que facilite la "difusión" de los mismos.
- El proceso de difusión de los datos obliga a definir y establecer objetivos que protejan la organización.

De esto se deducen cuatro conceptos involucrados que son:

1. Coherencia, asociado a la validez de los datos.
2. Integridad, sobre el total de la información a representar.
3. Seguridad, como garantía de los datos en su representación y
4. Confidencialidad, otorgando acceso de acuerdo a los niveles de la organización.

[2] Una base de datos es auto-descriptiva porque contiene además de los datos, la descripción de su estructura.
[3] También conocidos como diccionario de datos.

1.4 Estructura de las bases de datos

Antes del surgimiento de las bases de datos, los sistemas de información contaban con una estructura basada en dos niveles, el nivel lógico y el nivel físico, comprendiéndose dentro del nivel lógico las vistas de usuarios y dentro del nivel físico la forma en que se almacenan los datos.

Un objetivo importante en los sistemas gestores de base de datos es proporcionar a los usuarios una visión abstracta de los datos. Para esto el sistema debe ser manejable, es decir, los datos deben extraerse de forma eficiente. Esto se ha logrado mediante el diseño de estructuras de datos complejas para la representación de los datos en la base de datos.

La Arquitectura de tres niveles

En el enfoque de bases de datos una arquitectura que se acepta con regularidad es la propuesta por el grupo ANSI/SPARC[4], quien propone una arquitectura estándar en el que se consideran tres niveles:

1.- Un nivel conceptual.
2 - Un nivel lógico
3.- Un nivel físico

Es por esto que las bases de datos cuentan con una estructuración conocida como arquitectura de tres esquemas, la cual se lleva a cabo para poder obtener características como son la independencia entre datos y las aplicaciones y el manejo de varias vistas para el usuario. Esta arquitectura persigue como objetivo formar una separación entre las aplicaciones y la base de datos física y en ella se pueden definir los tres niveles siguientes:

[4] American National Standars Institute Standars Planning and Requirements Committee

Nivel Interno o Fisico

Este es el nivel más bajo de abstracción de datos. En el se especifica la forma en que se organizan los datos físicamente, es decir, describe la estructura física de almacenamiento de la base de datos, además de describir todos los detalles para el almacenamiento de los mismos y los caminos de acceso que van a existir para la base de datos.

Dentro de la estrategia de almacenamiento se va a incluir la asignación de espacios para cada conjunto de datos, mientras que en los caminos de acceso se especifican las claves, índices y punteros. Este nivel lleva asociada una representación de los datos, que es lo que denominamos Esquema Físico.

Nivel Conceptual

Este nivel va a responder al enfoque del conjunto de la empresa, describe la estructura de toda la base de datos para una comunidad de usuarios. Es a este nivel en donde se incluye la descripción de todos los datos así como las relaciones que hay entre éstos y en donde se definen las restricciones de integridad y de confidencialidad.

A este nivel de abstracción se ocultan los detalles de las estructuras físicas de almacenamiento y se describen las entidades, los tipos de datos, vínculos, operaciones de los usuarios y restricciones. Este nivel lleva asociado el Esquema Conceptual

Nivel Externo o de Vistas

El nivel Externo es el nivel más alto de abstracción y sólo describe una parte de la base de datos completa, también es llamado nivel de vistas debido a que describe la parte de la base de datos que le interese a un grupo de usuarios determinado, ocultando el resto de la base de datos a ese grupo.

Este nivel sirve para simplificar la interacción con el sistema, en donde se pueden especificar varias vistas para la misma base de datos, permitiendo tener la visión necesaria sobre la base de datos a cada usuario en particular con las restricciones de uso correspondientes como lo son los derechos de insertar, borrar o modificar determinados datos. El esquema asociado a éste nivel es el Esquema de Visión.

Estos tres niveles sólo contienen descripciones de los datos, existiendo datos realmente en el nivel interno. Los sistemas gestores de bases de datos no llegan a distinguir los tres niveles de abstracción pero algunos de ellos se basan en este tipo de arquitectura y deben ofrecer lenguajes para definir la base de datos conceptual, las vistas externas y la representación almacenada de la base de datos.

A menudo el nivel físico no es facilitado por muchos sistemas gestores de base de datos, esto es, no permiten al usuario elegir como se almacenan sus datos y vienen con una forma estándar de almacenamiento y manipulación de los datos.

La arquitectura a 3 niveles se puede representar como se muestra en la Figura 1.

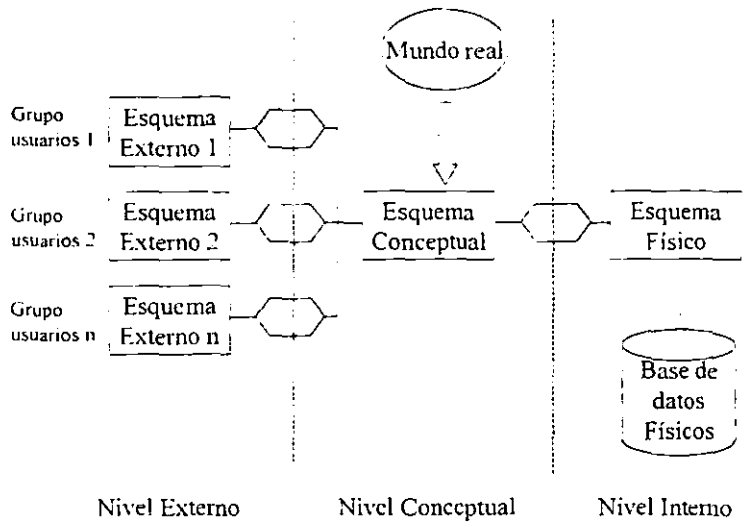


Figura 1. Arquitectura de 3 Niveles

CAPITULO 2. LOS SISTEMAS GESTORES DE BASES DE DATOS

2.1 Sistema Gestor de Base de Datos

Se debe distinguir la diferencia que existe entre una Base de Datos y un Sistema de Gestión de Bases de Datos (SGBD [5]). Un Sistema de Gestión de Bases de Datos va a ser aquella aplicación con la cual se accede y maneja los datos registrados dentro de la Base de Datos, va a proveer al usuario de una interfaz gráfica para poder recuperar y manipular los datos almacenados de una forma sencilla sin que los datos pierdan su consistencia e integridad. La unión de una Base de Datos con un Sistema de Gestión de Bases de Datos formarán un Sistema de Bases de Datos.

Un Sistema de Base de Datos debe responder a las siguientes características:

- **Independencia de los Datos.** Los datos no dependen del programa y por tanto cualquier aplicación puede hacer uso de los datos.
- **Reducción de la Redundancia.** Llamamos redundancia a la existencia de duplicación de los datos, al reducir ésta al máximo conseguimos un mayor aprovechamiento del espacio y además evitamos que existan inconsistencias entre los datos. Las inconsistencias se dan cuando nos encontramos con datos contradictorios.
- **Seguridad.** Un sistema gestor de base de datos debe permitir que tengamos un control sobre la seguridad de los datos.

Y deberá estar formado por: personas, máquinas, programas y datos. Los datos es a lo que se conoce como base de datos propiamente dicha, para manejar estos datos utilizamos una serie de programas, los cuales son los encargados de manejar los datos, estos programa son conocidos como SGBD (Sistema Gestor de Base de Datos).

[5] En inglés, Database Management System DBMS.

El Sistema de Gestión de Bases de Datos tiene como objetivo proporcionar un entorno conveniente y eficiente para ser utilizado al extraer y almacenar información de la Base de Datos. ofreciendo al usuario la posibilidad de interactuar con la Base de Datos en una forma de diálogo que le permita buscar. seleccionar, modificar y eliminar datos mediante una visión abstracta de los datos. Establece las adecuadas interfaces entre los diferentes tipos de usuarios y la base de datos.

Funciones de los sistemas gestores de base de datos

El sistema gestor de base de datos tiene dos funciones principales que son:

Función de Descripción.

El sistema gestor de base de datos debe permitir al diseñador especificar los datos que integran la base, la estructura y las relaciones que hay entre ellos, las reglas de integridad semántica, asi como las caracteristicas de tipo físico y las vistas lógicas de los usuarios.

Los sistemas gestores de base de datos realizan esta función por medio del lenguaje de definición de datos con que cuentan, suministrando los medios para definir las tres estructuras de datos.

Función de Manipulación.

Esta función se refiere al hecho de permitir al usuario interactuar con la base de datos en forma de diálogo. es decir, debe permitirle introducir, suprimir, buscar o actualizar los datos en la estructura creada para la base de datos. Para llevar a cabo esta función se debe tomar en cuenta los permisos y restricciones de acceso y manipulación de los datos que especifique el administrador de la base de datos.

El lenguaje que se encarga de llevar a cabo esta función es el lenguaje de manipulación de datos quien facilita los instrumentos necesarios para la realización de estas tareas.

Sin embargo, también se encarga de otras funciones más, aunque no tan principales pero si de gran importancia.

Función de Control.

Mediante el control se reúnen todas las interfaces que necesitan los diferentes usuarios para comunicarse con la base y proporciona un conjunto de procedimientos para el administrador.

Función de Integridad.

Cuanto mayor es el volumen de la información que se almacena en la base de datos, se corre mayor riesgo de que los datos almacenados no se ajusten a la realidad, sin embargo, el sistema gestor de base de datos ofrece al usuario la posibilidad de definir reglas que permitan mantener la integridad dentro de la base de datos Las restricciones de integridad siempre deben cumplirse en la base de datos.

Función de Confidencialidad.

Una característica de las bases de datos es que son multiusuarios, de modo que son varios los usuarios que van a compartir la base de datos y no todos podrán accesar a toda la información, así bien, el sistema gestor de base de datos ofrece mecanismos que verifican el derecho de acceso de los usuarios de la base de datos.

Función de Acceso Concurrente.

Los programas de aplicación que se soportan en la base de datos son muy a menudo accesados por varios usuarios al mismo tiempo, lo cual ocasionaría grandes problemas al manipular la información Para evitar esto el sistema gestor de base de datos cuenta con mecanismos que permitan la detección de aquellos casos en los que se producirían conflictos de acceso, como lo son los accesos para las actualizaciones.

Ventajas de los sistemas gestores de base de datos

Un Sistema de Gestión de Bases de Datos va a simplificar y facilitar el acceso a los datos proporcionando un control centralizado de los mismos y proveyendo las siguientes ventajas

- Tener datos al día.
- Reducir la redundancia y evitar la inconsistencia.
  Para conservar la consistencia, se crea un diseño que almacene cada dato lógico en un solo lugar de la base de datos, lo cual ahorrará espacio de almacenamiento, evitará la inconsistencia y reducirá la redundancia de los datos.
- Compartir datos.
- Unificar formatos de los datos.
- Aplicar restricciones de seguridad.
  Debido a que los valores de la base de datos deben cumplir con ciertas restricciones de consistencia, el sistema gestor de base de datos determina si las actualizaciones a la base de datos violan o no las restricciones establecidas por el administrador de la base de datos.
- Conservar la integridad.
- Agilizar la obtención de información.
- Poder tomar decisiones.
- Tener independencia en los datos.

2.2 Lenguajes de los sistemas gestores de bases de datos

Los SGBD utilizan algún lenguaje específico, de tal forma que sólo a través de él se puede interactuar con el sistema para realizar alguna operación con los datos. La estructura del lenguaje depende directamente del modelo de datos que implementa el SGBD. Cuando se habla de un lenguaje de SGBD, se tiende a pensar que el lenguaje es el SGBD en sí, y esto es un error, ya que el lenguaje como tal sólo es parte de un conjunto de herramientas que permiten la implementación de sistemas de información computacionales.

El papel que juegan los lenguajes de SGBD es simplemente proveer un protocolo de comunicaciones con los SGBD, de forma que cualquier usuario tenga acceso al SGBD. De

esta manera, cuando un usuario desea comunicarse con un SGBD, puede hacerlo sólo a través del lenguaje del SGBD, ya sea que se utilice una interfaz de línea de comandos o una aplicación cliente específica, desarrollada para satisfacer necesidades particulares de información, y que utiliza el mismo lenguaje.

El objetivo de los Sistemas Gestores de Bases de Datos es almacenar los datos en un computador, de forma segura y consistente, para permitir las solicitudes de acceso a los datos y que son formuladas por los usuarios. Estos sistemas permiten la implementación de bases de datos utilizando algún modelo, con el fin de satisfacer demandas particulares de los usuarios.
Para la definición de la estructura de la base de datos, realizada por el diseñador según un modelo de datos, así como para la utilización de ella por los usuarios, son necesarios lenguajes de comunicación con los SGBD. Estos lenguajes deben así permitir las siguientes operaciones:

- Definir de Estructura de la Base de Datos.
- Crear las bases de datos necesarias.
- Eliminar bases de datos existentes.
- Reestructurar las bases de datos existentes.
- Consultar acerca de la estructura de las bases de datos existentes.
- Manipular los Datos Almacenados.
- Insertar datos en las bases de datos.
- Eliminar datos de las bases de datos.
- Modificar datos de las bases de datos.
- Consultar datos de las bases de datos.

Mediante las operaciones mencionadas es posible utilizar los SGBD para almacenar bases de datos, las cuales pueden ser parte de sistemas informáticos completos, solucionándose así la necesidad de los usuarios por almacenamiento persistente de datos.

Los lenguajes de SGBD deben permitir realizar todas las operaciones antes mencionadas sobre las bases de datos y los datos que ellas contienen, de este modo para poder realizar las funciones correspondientes, surge la siguiente clasificación de los lenguajes del sistema gestor de base de datos:

- Lenguaje de Definición de Datos.
  Permite definir y manipular la estructura de bases de datos almacenadas en el SGBD,

- Lenguaje de Manipulación de Datos.
  Permite ingresar, modificar y eliminar datos de las bases de datos almacenadas en el sistema gestor de base de datos, además de realizar y controlar las transacciones.

Los sistemas gestores de base de datos actuales no distinguen entre los tipos de lenguajes, más bien, emplean un lenguaje integrado que cuente con elementos para definir esquemas conceptuales, definir vistas, manipular datos y definir su almacenamiento. Un ejemplo representativo es el lenguaje de base de datos relacionales SQL[6].

Lenguajes de Definición de Datos (DDL).

Los LDD permiten definir la estructura de la base de datos, y generalmente es el diseñador de ésta quien lo utiliza. La estructura de la base de datos se define en los tres niveles antes mencionados: nivel interno, nivel conceptual y nivel externo, es decir, el esquema de la base de datos se especifica por medio de un conjunto de definiciones expresadas mediante un lenguaje llamado Lenguajes de Definición de Datos.

Por tanto el sistema gestor de base de datos debe disponer de instrumentos que permitan al administrador de la base de datos describir y especificar la estructura de los datos, y cuentan con lenguajes autocontenidos que no necesitan bases de programación. El sistema gestor de base de datos contará con un compilador de DDL, el cuál se va a encargar de procesar enunciados escritos en el lenguaje de definición de datos para identificar las descripciones de los elementos de los esquemas y almacenar la descripción del esquema en el catálogo del sistema gestor de base de datos, es decir, compila las sentencias y el resultado será un conjunto de tablas que se almacenarán en un diccionario de datos, así antes de leer o modificar información de la base de datos este diccionario de datos es consultado para

[6] Structured Query Language.

verificar que las funciones de manipulación que se deseen llevar a cabo cumplan con el esquema que se ha definido para la base.
Algunos sistemas gestores de base de datos definen el esquema conceptual e interno con los lenguajes de definición de datos, sin embargo cuando exista una clara separación entre los niveles conceptual e interno, este lenguaje sólo será empleado para especificar el esquema conceptual.

Este tipo de lenguaje facilitará al administrador los medios para describir la estructura lógica global, para hacer las especificaciones de la estructura interna y para declarar las estructuras externas requeridas para el desarrollo de las aplicaciones.

En este tipo de lenguajes se hace la siguiente clasificación:

- Lenguaje de definición de la estructura lógica global.

  El administrador debe disponer de un instrumento de descripción que permita asignar nombres a los campos, a los agregados de datos, a los registros, establecer sus longitudes y características así como las relaciones y restricciones.

- Lenguajes para la definición de la estructura interna.

  Cuando en un sistema gestor de base de datos son independientes la estructura lógica y la estructura física, el sistema gestor de base de datos puede a partir de la estructura lógica, definir la estructura interna sin intervención del usuario, siempre y cuando se suministre al sistema gestor de base de datos las informaciones precisas como volúmenes, crecimiento previsto, tipos de registros más accedidos, relación entre actualizaciones y consultas.

- Lenguajes de definición de las estructuras externas.

  El sistema gestor de base de datos cuenta con medios que les permita a los usuarios recuperar o actualizar los datos contenidos en la base de acuerdo a la estructura externa (vista) que precise cada aplicación. Al definir una estructura externa es preciso darle un

nombre e indicar qué datos y que relaciones de la estructura lógica se encontrarán en la misma.

Lenguajes de Manipulación de Datos (DML).

Este lenguaje permite la utilización o manipulación de los datos almacenados en la base de datos, y es usado generalmente por usuarios finales o por aplicaciones cliente utilizadas por ellos. La forma más común de utilización del DML es mediante el uso de lenguajes de programación convencionales que poseen al DML del SGBD como lenguaje embedido. Así, se puede programar la aplicación cliente en el lenguaje que se elija, pudiendo realizar llamadas a sentencias del DML cuando sea necesario.

Se entiende por manipulación a toda aquella acción que permita recuperar, modificar, suprimir o insertar información en la base de datos. El lenguaje de manipulación de datos capacita a los usuarios a acceder y manipular los datos almacenados, proporcionando una interacción eficiente entre los usuarios y el sistema a través de vistas de usuario.

Los DML ofrecen a los usuarios la posibilidad de referirse a determinados conjuntos de datos que cumplan ciertos criterios de selección, para que el sistema gestor de base de datos acceda al soporte físico de donde extraerá los datos definidos para posteriormente transferirlos al dispositivo de salida o realizar la acción especificada por el usuario.

Las funciones de manipulación son cubiertas por el sistema gestor de base de datos mediante lenguajes autocontenidos que ofrecen facilidades a los usuarios con pocos conocimientos de programación, para que desde la estación de trabajo en forma interactiva, pueda acceder a la base y manipular los datos almacenados. Dentro de los lenguajes de manipulación de datos, se hace la siguiente una clasificación que los divide en dos, lenguajes no procedimentales y lenguajes por procedimientos[7].

[7] También son conocidos como Lenguajes de alto nivel y lenguajes de bajo nivel respectivamente

Lenguajes no procedimentales

Estos lenguajes especifican y recuperan muchos registros con una sola instrucción de lenguaje de definición de datos, por tanto se les llama lenguaje de definción de datos de conjunto, por conjunto u orientados a conjuntos. En estos lenguajes las consultas sólo especifican qué datos hay que obtener y no cómo obtenerlos, es decir, son declarativos y son los lenguajes que van a emplear los usuarios finales.

Lenguajes por procedimientos

Los lenguajes por procedimientos están incorporados en un lenguaje de programación de propósito general, y obtienen registros individuales de la base de datos y los procesan por separado, este tipo de lenguaje es también conocido como lenguaje de definición de datos de registro por registro. Debido a que este lenguaje requiere que se especifique qué datos y como obtenerlos, son los programadores quienes hacen uso de este tipo de lenguaje.

Es importante notar que los sistemas gestores de base de datos además de disponer de las facilidades proveídas por los lenguajes con que se conforman, también cuentan con utilerías que de igual forma simplifican y facilitan las tareas del administrador de la base de datos. De las utilerías más comunes con que cuenta un sistema gestor de base de datos se encuentran los siguientes:

- Carga de ficheros. Esta utilería permite la carga de archivos de datos ya existentes en la base de datos.
- Respaldo. Permite crear copias de seguridad de la base de datos por lo regular en cintas magnéticas, los respaldos son de gran utilidad para cuando ocurre alguna catástrofe con la base de datos y se desea restaurar la base de datos.
- Reorganización de archivos. Esta utilería sirve para modificar la organización de los archivos de la base de datos con el propósito de mejorar el rendimiento.
- Vigilancia del rendimiento. Son utilerías que vigilan la utilización de la base de datos y proporcionan estadísticas útiles para el administrador.

Caracteristicas de un buen lenguaje

Anteriormente se describió lo que son los lenguajes en un sistema gestor de base de datos, sin embargo todo lenguaje tanto de programación, de modelamiento, o de cualquier tipo, debe cumplir con algunas características que son consideradas fundamentales para hablar de un buen lenguaje, como son:

- Claridad, simplicidad y unidad
  Los lenguajes deben ser integrales, es decir, deben poder proveer de un conjunto de conceptos y de reglas para relacionarlos, de manera simple, clara y unificada. Además, deben ser legibles, para facilidad de comprensión.

- Ortogonalidad
  Esto quiere decir que las características del lenguaje, como funciones y procedimientos, deben ser combinables en todas sus alternativas Así, todas las posibles expresiones de un cierto tipo se pueden utilizar en todos los lugares en que se puede utilizar una expresión de ese tipo.

- Adaptación natural al problema
  Todos los lenguajes deben ser aplicables con naturalidad a los problemas. Es por ello que de aqui que surjan diferentes lenguajes para diferentes dominios de aplicación.

- Soporte de abstracción
  Los lenguajes deben proveer medios de abstracción, ya sea de datos y/o procedural. Así se permite al programador el representar el dominio del problema, por medio del lenguaje, sin preocuparse por los detalles de implementación más que una vez.

- Verificación fácil de programas
  Los lenguajes deben permitir una verificación fácil de los programas construidos, ya sea mediante verificación formal, lectura de ellos, o test.

- Ambiente apropiado de codificación
  No sólo el lenguaje es importante, sino también el ambiente en que se realiza la programación en él Parte de este ambiente es una implementación completa, correcta y

documentada del lenguaje, así como herramientas de creación, de verificación, o de prueba de código.

- Portabilidad de programas
  Los lenguajes deben estar definidos de manera independiente de los sistemas en que se utilizan, de forma estandarizada. Así es posible la existencia de implementaciones del mismo lenguaje en diferentes sistemas, para poder utilizar un mismo código en todos ellos sin modificaciones.

- Costo de uso bajo
  El costo de uso de un lenguaje esta asociado al costo en tiempo y recursos de hardware, software y otros que es necesario para crear, compilar, ejecutar, probar, utilizar y mantener los programas.

Objetivos de los lenguajes de los SGBD

Los SGBD pretenden ofrecer servicios de almacenamiento de datos de una forma abstracta, pudiendo el usuario de esos datos utilizarlos sin preocuparse por detalles de representación o de implementación de la base de datos. En general, los SGBD ofrecen un servicio que cumple con las características que se mencionan a continuación. Éstas deben estar presentes en los lenguajes de SGBD, ya que es a través de él que el usuario del SGBD aprecia las características.

- Soporte para Modelo de Datos
  Los SGBD permiten almacenar datos de manera abstracta, en base a un modelo de datos determinado El lenguaje del SGBD debe ser capaz de definir una estructura lógica de los datos de acuerdo al modelo de datos elegido, soportándolo en su totalidad.

- Independencia Física
  La forma en que se almacenan los datos en la base de datos no influye en su manipulación lógica (ya sea por DDL o por DML), de manera que cambios en el almacenamiento físico no influyen en los programas de usuario.

- Independencia Lógica

  La ejecución de operaciones del DML por parte de un usuario no repercute en las operaciones que realizan otros usuarios sobre la misma base de datos, de este modo los problemas de concurrencia deben ser transparentes al lenguaje del SGBD

- Flexibilidad

  El DDL del lenguaje del SGBD debe permitir definir diferentes formas de ver los datos, para satisfacer requerimientos diferentes por parte de usuarios diferentes

- Uniformidad

  La estructura lógica de la base de datos definida mediante el DDL debe ser uniforme y acorde al modelo de datos del SGBD, para facilitar la manipulación de esta estructura.

- Sencillez

  El lenguaje del SGBD debe ser sencillo, para facilitar a los usuarios la comprensión de la estructura lógica de los datos.

- DDL y DML

  Como ya se vio anteriormente, los lenguajes de SGBD necesitan de dos partes: un lenguaje de definición de datos, y uno de manipulación de datos. Al momento de decidir qué lenguaje utilizar, una posibilidad es usar dos lenguajes diferentes para cada parte del lenguaje. Sin embargo, generalmente se define un solo lenguaje, el cual incluye en una misma sintaxis el DDL y el DML. Cualquiera que sea la elección, el lenguaje debe poseer sentencias para cada funcionalidad requerida, ya sea del DDL o del DML.

- DDL

  En el caso específico del DDL, el lenguaje del SGBD debe ser capaz de definir la estructura lógica de la BD, sin entrar en detalles de implementación ni mecanismos en que se accede a los datos de la BD. La forma idónea de realizar esto es mediante un lenguaje declarativo, el cual permite declarar la estructura del modelo de acuerdo al modelo de datos que utiliza el SGBD.

- DML

  Para el caso del DML, el lenguaje del SGBD debe incluir formas de especificar qué se desea hacer con los datos (insertar, recuperar, modificar o borrar datos), sin entrar en detalles acerca de cómo se realizan estas operaciones.

2.3 Modelo de datos

El concepto de Modelo de Datos se refiere al grupo de herramientas conceptuales utilizadas para la descripción de la realidad de un sistema de información, es decir la descripción de un conjunto de datos y operaciones necesarias para manipular los datos. Este grupo de herramientas se compone de los datos, sus relaciones, su semántica y sus relaciones; instrumentos que utilizamos para el diseño de una Base de Datos a nivel lógico, dentro de la Arquitectura de Tres Niveles aceptada por la Norma ANSI/SPARC

A través de este modelo se van a expresar las entidades que conforman la base de datos y las relaciones que existen entre éstas, este conjunto de conceptos ayudarán a describir el esquema[8] de una base de datos.

Existen muchos modelos de datos pero estos pueden clasificarse en dos categorías de acuerdo a los tipos de conceptos que ofrecen para poder describir el esquema de la base de datos, por lo tanto se tienen modelos conceptuales y modelos lógicos.

Modelos Conceptuales [9].

Los modelos conceptuales representan la realidad a un nivel de abstracción alto, es a través de estos modelos mediante los cuales se puede construir una descripción de la realidad que sea fácil de entender y de interpretar, ya que en ellos se dispone de conceptos muy cercanos al modo en como los usuarios perciben los datos.

Los modelos conceptuales deben cumplir con algunas cualidades como son:

[8] Se denomina esquema a la estructura de la base de datos.
[9] Conocidos también como Modelos de alto nivel.

1 Expresividad.
2. Simplicidad. Debe ser simple para que el esquema creado con este modelo sea fácil de entender por los diseñadores y usuarios de la base de datos.
3. Minimalidad. Cada concepto en el modelo debe tener un significado distinto con respecto a los demás.
4 Formalidad. Todos los conceptos del modelo deben tener una interpretación única, precisa y bien definida.

Modelos Lógicos [10].

Los modelos lógicos van a proporcionar conceptos que describan los detalles de cómo se almacenan los datos. Este tipo de modelo tienen una correspondencia sencilla con la estructura física de la base de datos y se encuentran soportados por los sistemas gestores de base de datos ya que están orientados a describir los datos a nivel lógico para el sistema gestor de base de datos.

El modelo de datos representa la organización conceptual que tienen los datos y ayuda a la organización y visualización de ideas. Los sistemas gestores de bases de datos utilizan un modelo de datos a través del cual se define la estructura que van a tener los datos, fundamentándose para ello en varios modelos de acuerdo a las necesidades existentes.

Hacer una clasificación adecuada de los sistemas gestores de base de datos puede ser difícil, sin embargo, el principal criterio para realizar una clasificación es aquel en el que se toma en cuenta el modelo de datos sobre el que están basados. Para representar el mundo real a través de esquemas conceptuales se han creado una serie de modelos, sin embargo los modelos de datos que se emplean con mayor frecuencia en los sistemas gestores de base de datos comerciales son el Modelo Relacional, el Modelo de Red y el Modelo Jerárquico.

[10] Reconocidos también como Modelos de bajo nivel o Modelos físicos.

Modelo Relacional

La visión relacional corresponde al almacenamiento en forma de tablas, aquí los datos se representan en forma de tablas llamadas relaciones. A las columnas de una tabla se les conoce como atributos y a los renglones o filas de la relación se les conoce como tupias. El modelo representa al mundo real mediante tablas relacionadas entre si por columnas comunes y se ocupa de tres aspectos de los datos: su estructura, su integridad y su manipulación.

Una relación debe cumplir con ciertas características. Las celdas de la tabla deben tener un solo valor, no se permiten arreglos como valores. Todos los valores de cualquier columna deben ser del mismo tipo. Cada columna tiene un nombre único, es decir, en una misma tabla no pueden existir dos columnas con el mismo nombre, además el orden de las columnas es insignificante. Dos tuplas de una relación no pueden ser idénticos y el orden de las tuplas es insignificante.

Casi todas las base de datos relacionales tienen lenguajes de consulta de alto nivel y manejan una forma limitada de vistas de usuario.

Modelo de Red

Representa al mundo real como registros lógicos que representan a una entidad y que se relacionan entre si por medio de flechas. Una base de datos en red se construye a partir de registros que se conectan entre sí por medio de ligas. Cada registro es un conjunto de campos (atributos). Un campo almacena el valor de un solo dato y una liga puede asociar exclusivamente dos registros. Los diagramas de estructura de datos fueron introducidos por Bachman en 1969. Estos diagramas sirven para representar una base de datos de red. En estos diagramas representaremos un tipo de registro por medio de un cuadro y una liga con una línea.
Este modelo tiene un lenguaje de registro por registro asociado que se debe incorporar en un lenguaje de programación anfitrión.

En el modelo de red hay dos estructuras de datos fundamentales que son los tipos de registros y los conjuntos. Los tipos de registros son una colección de elementos de datos lógicamente relacionados y un conjunto expresa una relación uno a muchos entre dos tipos de registros. La Figura 2 ilustra un esquema de red para una base de datos:

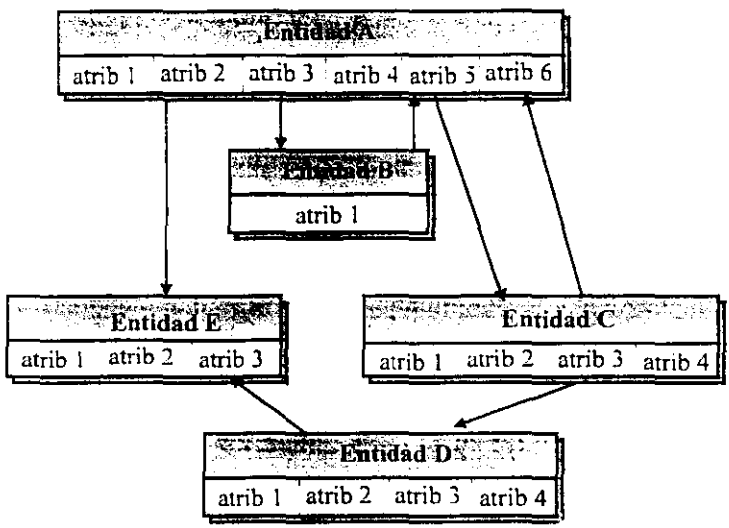


Figura 2. Modelo de Red

Modelo Jerárquico

El modelo de datos jerárquico es muy similar al modelo de datos de red. Una base de datos jerárquica se compone de un conjunto de *registros* conectados entre si por medio de *ligas*. Un registro, igual que en el modelo de red, es un conjunto de datos o atributos que almacenan cada uno un valor. Una liga es una asociación entre dos registros. Los registros en el modelo de datos jerárquico se organizan formando conjuntos de árboles a diferencia del modelo de red en donde se forman gráficas arbitrarias.

Este modelo representa los datos como estructuras jerárquicas de árbol y cada jerarquía representa varios registros relacionados entre sí, tiene forma de árbol invertido en donde un padre puede tener varios hijos pero cada hijo sólo puede tener un padre

Para el modelo jerárquico no existe un lenguaje estándar, aunque la mayoría de los sistemas gestores de base de datos jerárquicos cuentan con lenguajes de registro por registro.

Un diagrama de estructura de árbol es el esquema de una base de datos jerárquica, utiliza cuadros para representar tipos de registros y líneas que representan ligas. Las ligas conectan dos tipos de registros Cuando la línea termina en flecha se indica que es una relación uno a muchos (la flecha apunta al *uno* en la relación), si la liga tiene flechas en ambos extremos de la línea se indica una relación uno a uno. Un diagrama de estructura de árbol se representaría como en la Figura 3, se muestra.

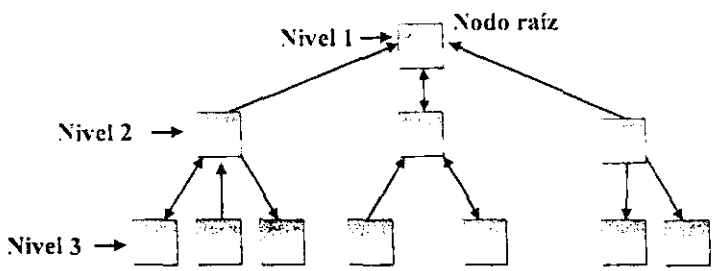


Figura 3. Modelo Jerárquico

CAPITULO 3. BASES DE DATOS RELACIONALES

3.1 El Modelo Relacional

Las bases de datos relacionales surgen a partir de un articulo de Codd en 1970, desde entonces es posible encontrarlas en todo centro de cómputo. Muchas aplicaciones y usuarios de bases de datos tienen a su alcance software para el manejo de bases de datos relacionales de muchas marcas comerciales debido al modelo de datos matemático que lo respalda.

El trabajo publicado por Codd además de tener como objetivo fundamental mantener la independencia de esta estructura lógica respecto al modo de almacenamiento y a otras caracteristicas de tipo fisico, perseguía una serie de objetivos como son.

- La independencia física, el modo en que se almacenan los datos no debe influir en su manipulación lógica.
- La independencia lógica, la manipulación de los datos como añadir, modificar o eliminar no debe repercutir en los programas y/o usuarios que estén accediendo a subconjuntos parciales de los mismos.
- Flexibilidad, poder ofrecer a cada usuario los datos de la forma más adecuada a la correspondiente aplicación.
- Uniformidad, las estructuras lógicas de los datos presentan un aspecto uniforme, lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- Sencillez, al cumplir con las caracteristicas anteriores, el modelo de datos Relacional se vuelve fácil de comprender y utilizar por parte del usuario final.

Hasta el principio de la década de los sesenta el modelo más empleado como estructura básica era el modelo jerárquico, sin embargo a principios de esa misma década Codd introduce el modelo relacional junto con dos lenguajes de manipulación de datos basados en la lógica como lo son el álgebra relacional y el cálculo relacional, lo cual permitió dar más poder en el acceso y procesamiento de los datos, pero sin dejar de existir ciertas desventajas.

Desventajas del Modelo Relacional

El Modelo Relacional, es con mucho, el de más auge en la actualidad. El aumento considerable de los sistemas gestores de bases de datos relacionales hoy en día, no hace más que afirmar su gran valía como modelo de datos, por esto han llegado a ser un estándar en el mercado, especialmente en lo que respecta a las operaciones comerciales.

El modelo relacional se ocupa de tres aspectos de los datos: su estructura, su integridad y su manipulación. En lo que respecta a la estructura, son los conceptos de relación, atributos y dominios; su integridad está dada por el concepto de clave, posibilidades de valores nulos y dos reglas de integridad; mientras que la manipulación se refiere a los operadores, que incluyen operadores de actualización y la llamada álgebra relacional.

Conceptos Fundamentales en el Modelo Relacional

En un modelo relacional se representa a la base de datos como una colección de relaciones, una relación representa una simple tabla de dos dimensiones. En términos de lo que es el modelo relacional una relación se conforma por filas denominadas tuplas y columnas llamadas atributos mientras que el tipo de datos que describe los tipos de valores que pueden aparecer en cada columna se llama dominio.

El dominio es un conjunto de valores atómicos, es decir cada valor del dominio es indivisible. La especificación de los dominios consiste en especificar un tipo de datos al cual pertenecen los valores que constituyen el dominio. Además debe especificarse un tipo de datos o formato para cada dominio, por tanto un dominio debe tener un nombre, un tipo de datos y un formato

Cada columna de la relación es un atributo que identifica o caracteriza a un tipo de dato de la relación. El número de atributos en una relación se llama grado de la relación, mientras que el número de tuplas de una relación se denomina cardinalidad.
Las tuplas se distinguen unas de otras por medio de su Clave Primaria. Toda tupla tiene una clave primaria, por lo tanto, toda tupla es distinguible. Si en una relación aparece un atributo que es clave primaria en otra relación, se le denomina Clave Foránea.

La clave primaria es un identificador único para la tabla, es decir, un atributo o combinación de atributos tal que nunca van a existir dos tuplas de la relación con el mismo valor en ese atributo o combinación de atributos.

Estructuras de datos en el Modelo Relacional

En el modelo relacional las estructuras de datos son los conceptos de relación, dominio y atributo, por tanto se profundizará en dichos conceptos.

Relación: denota una colección ó conexión entre dos conjuntos de datos que tienen los mismos tipos de características o atributos.
Atributo: característica o propiedad que identifica o describe a la relación, cada atributo tiene un dominio asociado.
Dominio: es el conjunto de valores que puede tomar un atributo.

Las relaciones se representan por tablas donde las columnas son los atributos. En los renglones se almacenan los elementos de datos con sus valores para cada atributo. Una representación de una relación es indicar su nombre y entre llaves el conjunto de atributos. A esta representación también se le llama esquema de la relación.

Por ejemplo, si se tiene una relación llamada TarjetaHabiente cuyos atributos son: Nombre, Domicilio, Teléfono y NumCuenta, el esquema de la relación quedaría de la siguiente forma:

TarjetaHabiente = {Nombre, Domicilio, Teléfono ,NumCuenta }

Ahora bien, cada atributo tiene un dominio asociado. Los dominios son los conjuntos de los valores posibles, por tanto estos se representarían de la siguiente forma:

Dom (Nombre) = {El conjunto de nombres de los tarjeta habientes}

Dom (Domicilio) = {El conjunto direcciones de los tarjeta habientes }

Dom (Teléfono) = {El conjunto de números telefónicos de 8 dígitos enteros válidos en el país}

Dom (NumCuenta) = {El conjunto de números de cuentas de 16 dígitos enteros válidos para la institución bancaria}

Características de las Relaciones.

Una relación debe cumplir con ciertas características propias de las relaciones con respecto a las tablas como son las siguientes:

- Todas las tuplas tienen que ser distintas, es decir, no pueden existir tuplas duplicadas.
- Las tuplas de una relación no tienen un orden específico ya que el ordenamiento de las tuplas no forma parte de la definición de una relación.
- El orden de los atributos y sus valores no es importante en tanto se mantenga la correspondencia entre los atributos y sus valores.
- Cada valor en una tupla debe ser un valor atómico. No se permiten atributos compuestos ni multivaluados, esto quiere decir que en el cruce de un atributo y una tupla sólo puede haber un valor, por tanto no se permiten arreglos como valores.
- Todos los valores de cualquier atributo deben ser del mismo tipo, es decir, si un atributo contiene números de cuenta, todas las tuplas en ese atributo deberán contener números de cuenta.

Integridad en el Modelo Relacional

El modelo relacional incluye varias restricciones que son usadas para verificar la validación de los datos dentro de la base de datos como son restricciones de integridad de la entidad y restricciones de integridad referencial.

Es muy importante en este modelo todo lo referente a la integridad y consistencia del mismo. Los conceptos de definiciones de integridad para el modelo relacional son clave primaria y foránea, valores nulos y reglas de integridad.

Como norma general se han introducido dos reglas de integridades o propiedades de tipo semántico que la base de datos debe cumplir, para entender estas reglas es necesario abordar los temas siguientes.

Claves primarias.

Una clave candidata de una relación es un conjunto de atributos que identifican unívoca y mínimamente cada tupla de la relación, la clave candidata es cualquier conjunto de atributos que puede ser elegido como una clave de una relación.

Una clave candidata para una relación R es un atributo K posiblemente compuesto, tal que satisface las siguientes dos propiedades independientes del tiempo:

- Unicidad. En cualquier momento dado, no existen dos tuplas en R con el mismo valor de K.
- Minimalidad. Si K es compuesto, no será posible eliminar ningún componente de K sin destruir la propiedad de unicidad.

Cuando se escoge una clave candidata para identificar las tuplas de la relación, esta clave se convierte en clave primaria. Ningún componente de la clave primaria de una relación puede en algún momento no tener valor, es decir aceptar nulos. Esto significa que no tiene sentido modelar una entidad que no podemos identificar ni distinguir unas de otras.

Como ya se mencionó dentro de los atributos debe haber uno o varios que desempeñen el papel de clave primaria, a fin de mantener la integridad a lo largo del tiempo en una base de datos relacional, por tanto se debe cumplir con algunas restricciones en cuanto a los valores de las claves primarias, de donde se deriva la primera regla de integridad:

1. Integridad de Relaciones

Ningún valor de una clave primaria puede ser nulo. Esto es porque el valor de la clave primaria sirve para identificar las tuplas individuales en una relación.

Claves Foráneas.

En el modelo relacional se denominan claves foráneas[11] a una referencia de una relación a otra, mediante su clave. Una clave foránea es un conjunto de atributos en una relación que constituyen una clave en alguna otra relación. Una Relación (R1) puede poseer como uno de sus atributos (A) una clave primaria de otra relación (R2). Este atributo (A) constituye una clave foránea en R1 y referencia a R2.

Una relación puede tener una o varias claves foráneas y es de aquí de donde se deriva la segunda regla de integridad que indica:

2. Integridad de Referencia

Todo valor de una clave foránea debe ser distinto de nulo y además pertenecer al conjunto de valores de la relación donde dicha clave sea primaria, esto quiere decir que una tupla en una relación que haga referencia a otra relación deberá referirse a una tupla existente en esa relación. Esta regla de integridad se especifica entre dos relaciones y sirve para mantener la consistencia entre tuplas de las dos relaciones, de modo que la base de datos no debe contener valores de claves foráneas sin concordancia.

Estas dos reglas de integridad se ven complementadas por una serie de restricciones de integridad, que en cada modelo persiguen el objetivo de salvaguardar la consistencia y verificabilidad de los datos.

[11] Los textos literarios también suelen referirse a ellas como claves ajenas.

Algebra Relacional

El álgebra relacional consiste en un conjunto de operadores de alto nivel que operan sobre relaciones. Cada uno de estos operadores toma una o dos relaciones como entrada y produce una nueva relación como salida. Codd definió un conjunto de 8 operadores que se describen a continuación.

1. Restricción. Extrae las tuplas especificadas de una relación dada (o sea, restringe la relación sólo a las tuplas que satisfagan una condición especificada).
2. Proyección. Extrae los atributos especificados de una relación dada.
3. Producto. A partir de dos relaciones especificadas, construye una relación que contiene todas las combinaciones posibles de tuplas, una de cada una de las dos relaciones.
4. Unión. Construye una relación formada por todas las tuplas que aparecen en cualquiera de las dos relaciones especificadas.
5. Intersección. Construye una relación formada por todas aquellas tuplas que aparecen en las dos relaciones especificadas.
6. Diferencia. Construye una relación formada por todas las tuplas de la primera relación que no aparezcan en la segunda de las dos relaciones especificadas.
7. Reunión. A partir de dos relaciones especificadas, construye una relación que contiene todas las posibles combinaciones de tuplas, una de cada una de las dos relaciones, tales que las dos tuplas participantes en una combinación dada satisfagan alguna condición especificada.
8. División. Toma dos relaciones, una binaria y otra unaria, y construye una relación formada por todos los valores de un atributo de la relación binaria que concuerdan (en el otro atributo) con todos los valores en la relación unaria.

Teoría de la Normalización

El proceso esencial para organizar datos en tablas relacionales es el de normalización que consiste en convertir relaciones complejas en otras más simples que cumplan las condiciones del modelo relacional. Sin embargo, al descomponer la información en tablas

normalizadas, reduce la velocidad de las busquedas por lo que ha de asumirse un compromiso entre rigor organizativo y rapidez en la gestión.
El diseño de esquemas para generar bases de datos relacionales debe considerar el objetivo de almacenar información sin redundancia innecesaria, pero que a la vez nos permitan recuperar información fácilmente. Una técnica consiste en diseñar esquemas que tengan una forma normal adecuada.

Las relaciones se clasifican por los tipos de anomalías de modificación a las que son vulnerables. Estas anomalías se pueden clasificar y prevenir por medio de las llamadas formas normales, las cuales están anidadas, lo cual quiere decir que una relación en segunda forma normal está también en primera forma normal, y una relación en quinta forma normal, está también en 4NF, BCNF, 3NF, 2NF y 1NF. Las formas normales, definidas en la teoría relacional, nos permiten evitar que propiedades indeseables aparezcan en una base de datos basada en un esquema mal diseñado. Un esquema debe estar a lo menos en tercera forma normal, para que sea aceptable.

Formas Normales

- Primera Forma Normal

  Cualquier tabla que cumpla con la definición de relación está en primera forma normal, es decir cada celda de la tabla debe contener un solo valor, no se permiten arreglos como valores. Todos los atributos deben ser del mismo tipo, cada columna tiene un nombre único en donde el orden de las columnas es insignificante, además de que dos renglones en una tabla no pueden ser idénticos y el orden de estos es insignificante. En otras palabras una relación está en primera forma normal (1FN) si y sólo si todos los dominios simples subyacentes contienen sólo valores atómicos.

- Segunda Forma Normal

  Una relación está en segunda forma normal (2FN) si y sólo si está en 1FN y todos los atributos no clave dependen por completo de la clave primaria.

Según la definición, todas las relaciones que tienen por clave a uno solo de sus atributos, están automáticamente en segunda forma normal. Si la clave es de un solo atributo, todos los atributos no-clave son dependientes de toda la clave.

- Tercera Forma Normal

Una relación está en tercera forma normal (3FN) si y sólo si está en 2FN y todos los atributos no clave dependen de manera no transitiva de la clave primaria
La tercera forma normal es transgredida cuando una propiedad no identificada (no clave) es un dato acerca de otro campo no clave.

Estas son las tres formas normales básicas, existen además la forma normal de Boyce/Codd, la cuarta forma normal y quinta forma normal.

3.2 Lenguaje de Consulta SQL Server

El lenguaje relacional SQL fue desarrollado originalmente por IBM en un sistema prototipo llamado System R, con el nombre de SEQUEL, en los años 1974-1975. Posteriormente por razones legales, pasó a llamarse SQL[12]. En 1979 surgieron los primeros productos comerciales que lo implementaron (SQL Oracle). Fue hasta 1992 que el lenguaje SQL fue finalmente estandarizado por los grupos ANSI e ISO, llamándose SQL-92 o SQL2.

Las siglas SQL quieren decir lenguaje de consulta estructurado, y es utilizado para la creación y manipulación de B.D. relacionales. Existen ciertos estándares SQL cuyas normas vienen dadas por el Instituto Nacional Norteamericano de Normalización, conocido como ANSI y que se refieren a la semántica y sintaxis del mismo.

La función del lenguaje SQL es la de soportar la definición, manipulación y control de los datos en una base de datos relacional, este lenguaje está compuesto por comandos,

[12] En Inglés Structured Query Languaje

cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos

De acuerdo a la función de las sentencias los comandos SQL se van a dividir en

- *Lenguaje de Definición de Datos* (DDL): Que permiten la creación y modificación de la estructura o esquema de la B.D.

COMANDO SQL	DESCRIPCION
Create	Crea objetos en la base de datos, crea nuevas tablas, campos e índices.
Alter	Modifica objetos de la base de datos, modifica las tablas agregando campos o cambiando la definición de los campos.
Drop	Elimina objetos de la base de datos como son tablas e índices.

- *Lenguaje de Manipulación de Datos* (DML): Que permite generar consultas para ordenar, filtrar y extraer datos, así como el mantenimiento de los datos.

COMANDO SQL	DESCRIPCION
Select	Consulta registros de la base de datos que cumplan con un criterio determinado.
Insert	Carga lotes de datos en la base de datos en una única operación.
Update	Utilizado para modificar los valores de los campos y registros especificados.
Delete	Elimina registros de una tabla de una base de datos.

- *Lenguaje de Control de Datos* (DCL): Proporciona seguridad e integridad de los datos.

COMANDO SQL	DESCRIPCION
Grant	Da acceso a los Objetos de la base de datos.
Revoke	Quita acceso a los Objetos de la base de datos
Commit	Confirmación de Transacción
RollBack	Vuelta atrás de Transacción.

Además de los comandos se encuentran también las cláusulas, operadores lógicos, operadores de comparación y funciones de agregado que se listarán a continuación.

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular y existen las siguientes:

CLAUSULA SQL	DESCRIPCION
From	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
Where	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
Group by	Utilizada para separar los registros seleccionados en grupos específicos
Having	Utilizada para expresar la condición que debe satisface cada grupo
Order by	Utilizada para ordenar los registros seleccionados de acuerdo con un orden especifico

Operadores Lógicos

OPERADOR SQL	USO
And	Es el "y" lógico. Evalúa dos condiciones y devuelve un valo de verdad sólo si ambas son ciertas.
Or	Es el "o" lógico. Evalúa dos condiciones y devuelve un valo de verdad si alguna de las dos es cierta.
Not	Negación lógica. Devuelve el valor contrario de la expresión.

Operadores de Comparación

OPERADOR SQL	USO
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que
Between	Utilizado para especificar un intervalo de valores.
Like	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

FUNCIÓN	DESCRIPCIÓN
AVG	Utilizada para calcular el promedio de los valores de un campo determinado.
COUNT	Utilizada para devolver el número de registros de la selección.

SUM	Utilizada para devolver la suma de todos los valores de u campo determinado.
MAX	Utilizada para devolver el valor más alto de un camp especificado.
MIN	Utilizada para devolver el valor más bajo de un campo especificado.

SQL no es un lenguaje de programación ya que no posee sentencias de selección o iteración, por lo tanto, cuando se desean utilizar los datos de la base de datos para realizar algún proceso, se debe programar en otro lenguaje conocido como lenguaje anfitrión, desde el cual se hacen llamadas a sentencias SQL[13] cuando se desea interactuar con la base de datos.

Es un lenguaje bastante natural y fácil de leer, y se aplica de forma natural al problema de implementar bases de datos relacionales. sin embargo, es poco estructurado y utiliza muchas palabras superfluas que podrían omitirse quitando claridad a las sentencias.

SQL permite abstraer al usuario de la estructura lógica de la base de datos mediante las vistas, aunque éstas imponen bastantes restricciones a los procesos de insertar y modificar datos en ellas.

El SQL es mucho más que un lenguaje de consulta de bases de datos, ya que permite:

- Definir la estructura de los datos, recuperar y manipular datos;
- Administrar y controlar el acceso a los datos;
- Compartir datos de forma concurrente;
- Asegurar su integridad.

Esto se hace siempre basándose en el Modelo Relacional de datos, por tanto SQL permite hacer efectivos los objetivos de los SGBDR[14], permitiendo a los usuarios utilizar todo el potencial de los SGBDR.

[13] Mejor conocido como SQL embedido.
[14] Sistema Gestor de Base de Datos Relacionales.

El lenguaje SQL permite representar casi todos los elementos del modelo relacional, salvo los conceptos de dominio (que no se soporta) y de relación (que se soporta de forma restringida). La manipulación de datos proveida por SQL, es más poderosa que la definida originalmente en el modelo relacional, mediante el álgebra relacional. Esto ha motivado que ésta haya sido extendida para representar las posibilidades de SQL.

Dominios Relacionales

Los dominios SQL y los dominios del modelo relacional son conceptos diferentes. Un dominio relacional es un tipo de dato con las siguientes características:

- Los dominios pueden ser definidos por el usuario;
- La definición de un dominio especifica un conjunto de valores permitidos para éste;
- Los valores del dominio pueden ser de una complejidad arbitraria;
- La representación interna de dichos valores no es visible para el usuario;
- Los valores son manipulados solamente por operadores definidos por el dominio;
- Los dominios pueden ser subtipos de otros dominios;
- Se puede definir un dominio en términos de otro dominio (herencia): Si B es un subtipo de A, entonces todos los operadores que son aplicables a A son también aplicables a B, pero B puede tener operadores propios que no son aplicables a A.

Los dominios del modelo relacional ofrecen, además, la propiedad de "strong typing", lo que significa que cuando un usuario escribe una expresión, el sistema chequea que los operandos de cada operador sean los permitidos por el dominio. En SQL, sin embargo, el propósito de los dominios es permitir la especificación de un tipo de dato, tal como char(5). Los tipos de datos se definen sólo una vez, permitiendo de este modo que dicha especificación sea usada por columnas pertenecientes a muchas tablas.

Las diferencias más importantes entre los dominios SQL y dominios relacionales son:

- Los dominios SQL son especificaciones sintácticas; no son tipos de datos definidos por el usuario.

- Los dominios SQL no siempre se usan, ya que las columnas de las tablas pueden definirse directamente en términos de tipos de datos soportados por el sistema.
- Los dominios SQL no soportan definiciones de dominios en términos de otros dominios, ya que se definen siempre en términos de tipos de datos del sistema (no hay herencia).
- Los dominios SQL no permiten chequeos, ya que los únicos requerimientos para hacer comparaciones es que los comparandos sean del mismo tipo de datos.
- SQL no permite que los usuarios definan operaciones que se apliquen a los dominios.
- SQL no hace una clara distinción entre un dominio SQL y la representación de ese dominio en términos de tipos de datos soportados por el sistema.

La sintaxis de definición de dominios en SQL es la siguiente:

CREATE DOMAIN *domain-data-type*
[*default-definition]*
[*dom-constraint-deflist]* ;

en donde:

domain-data-type es el tipo de dato SQL para el dominio;
default-definition especifica un valor por defecto que se aplica a todas las columnas que están definidas sobre el dominio y no permite la definición de un valor por defecto explícito perteneciente a los propios del dominio;
dom-constraint-deflist especifica un conjunto de restricciones de integridad que se aplican a todas las columnas definidas sobre ese dominio. Las reglas de integridad sólo permiten enumeraciones de valores.

Relaciones y Tablas Base

Las propiedades de las relaciones en el modelo relacional son:

- No existen tuplas repetidas, dado que una relación es un conjunto matemático

- Las tuplas no están ordenadas, dado que el cuerpo de una relación es un conjunto matemático, y por definición éstos no son ordenados.
- Los atributos no están ordenados, dado que la cabecera de una relación se define también como un conjunto.

Las tablas de SQL no son relaciones, ya que no cumplen con las propiedades recién mencionadas. La forma en que se representa una relación en SQL es mediante una tabla base, las cuales tienen las siguientes características anómalas:

- Las tablas base de SQL permiten filas repetidas, a no ser que se defina un mecanismo que lo evite;
- Las tablas SQL tienen un orden de columnas, de izquierda a derecha, y de filas, de arriba a abajo.

Claves Candidatas

Las claves candidatas se declaran en SQL como:

UNIQUE (*column-commalist*) ó PRIMARY KEY (*column-commalist*)

En donde:

column-commalist no debe ser vacio.

El modelo relacional asegura que toda relación tiene al menos una clave candidata. La importancia de éstas es que constituyen el mecanismo de direccionamiento a nivel de tuplas en el sistema relacional. Anteriormente mencionamos que las tablas SQL pueden contener tuplas repetidas, lo que contradice el concepto de clave. Aún más, aquellas tablas que contengan tuplas repetidas presentarán un comportamiento extraño y errado en muchas circunstancias. Otra consecuencia de lo anterior es que si un sistema no maneja correctamente las claves primarias, presentará problemas en la actualización de vistas.

Reglas de integridad

El término integridad en las bases de datos se refiere a asegurar que los datos sean válidos. Estas reglas se definen actualmente mediante procedimientos almacenados y/o triggers (programación, lo cual no es parte del estándar SQL)

Es deseable que los SGBD almacenen las reglas de integridad en el catálogo, de forma que el subsistema de integridad esté constantemente monitoreando las transacciones que realizan los usuarios, asegurando que las reglas se cumplan, por tanto se vuelve necesario especificar las reglas de integridad, mediante un comando SQL, tal como lo es el CREATE INTEGRITY RULE.

Este comando hipotético de SQL representa la estructura de una regla de integridad, consistente del nombre de la regla, la restricción (que especifica la evaluación de una expresión como verdadera o falsa) y la respuesta a la violación de la regla (que define qué acciones se tomarán cuando la regla no se cumpla).

Ortogonalidad y SQL

Un lenguaje es ortogonal cuando en todo lugar en que se espera un valor de cierto dominio, se puede incluir una expresión cualquiera que retorne un valor de ese dominio
En particular, las funciones de agregados de SQL carecen de ortogonalidad, por ejemplo, realizar una consulta que implique sumar ciertos valores numéricos de un cierto atributo (con la función SUM) y luego consultar cuáles de aquellos valores (ya sumados) cumplen una cierta condición, por ejemplo mayores que un cierto número.

Facilidades de Implementación de BD Relacionales con SQL

Para poder ofrecer funcionalidades a los diseñadores de sistemas de bases de datos, los SGBDR ofrecen extensiones del SQL, basadas en triggers, procedimientos almacenados y cursores.

Además el SQL con su característica de ser un lenguaje dual, le entrega a los usuarios finales la posibilidad de que sean ellos mismos los que realicen consultas a la base de datos

Dualidad del Lenguaje

Los comandos del lenguaje SQL pueden ser ejecutados interactivamente o bien formando parte de un programa de aplicación. Esta característica de SQL se conoce con el nombre de principio de modo dual

Existen diferencias entre las sentencias SQL declaradas interactivamente y su contraparte embebida. Además, algunas sentencias de SQL embebido no pueden usarse interactivamente y viceversa.

Para realizar eficientemente programas de aplicación, los SGBDR comerciales ofrecen una extensión de SQL, que consiste en agregar sentencias procedurales tradicionales, tales como IF-THEN-ELSE, FOR, WHILE, etc.

Triggers

Los Triggers son eventos que se disparan cuando se detectan ciertos estados en la base de datos. Su función es permitir la implementación de reglas corporativas y permanentes, y su uso más típico ha sido el de proteger la integridad referencial de la base de datos.

El "trigger" llama "stored procedures" que se han programado previamente para atender a cada uno de dichos eventos.

Procedimientos Almacenados

Los procedimientos almacenados son un conjunto de transacciones SQL, a las que se les asigna un nombre, son compiladas y almacenadas en un servidor de base de datos. Los

stored procedures se programan para cumplir la parte de la lógica de la aplicación que se desea se ejecute en el servidor.

Después de que se define un procedimiento almacenado en la base de datos, éste puede ser llamado desde un programa de aplicación, mediante su nombre, por otro de ellos, por un trigger o, directamente desde el cliente, mediante una llamada remota

Esta funcionalidad está presente en la mayoría de los SGBDR comerciales, dentro de lo que se conoce como API (Application Program Interface).

Cursores

Los cursores soportan el desarrollo de programas de base de datos, permitiendo al usuario moverse libremente a través de un conjunto de resultados de consultas. Este tipo de interfaz de usuario es muy valorada en aplicaciones de soporte de decisiones y en aplicaciones de bases de datos basadas en consultas.

Esta extensión de SQL, es entregada por muchos SGBDR comerciales, proporcionando funcionalidad a los programadores.

3.3 La Arquitectura Cliente/Servidor

Antecedentes

Hace unos años cuando se hablaba mucho y se hacía muy poco sobre el tema, se decía que el desarrollo de aplicaciones Cliente / Servidor era inevitable por varias razones:

- En muchas situaciones es más eficiente que el procesamiento centralizado, dado que éste experimenta una "des-economía" de escala cuando aumenta mucho la cantidad de usuarios.
- Existían ya en ese momento servidores razonablemente eficientes y confiables.

- Se habia establecido un estándar de hecho para una interfase Cliente/Servidor: el ODBC SQL, adoptado por todos los fabricantes importantes de servidores
- Era imprescindible, para apoyar con información a la creciente cantidad de ejecutivos de nivel medio que necesitan tomar decisiones ante el computador, ayudándose con las herramientas "front office" que utilizan con toda naturalidad (planillas electrónicas, procesadores de texto, graficadores, correos electrónicos, etc.).

Sin embargo, existía tecnología para esta arquitectura desde hacía ya bastantes años sin que nada ocurriera. Los primeros trabajos conocidos para la arquitectura Cliente/Servidor los hizo Sybase, que se fundó en 1984 pensando en lanzar al mercado únicamente productos para esta arquitectura. A fines de la década pasada el producto fue lanzado para el voluminoso segmento "low-end" del mercado, en conjunción con Microsoft, teniendo como soporte de la base de datos un servidor OS/2, y como herramienta "front end" básica el Dbase IV de Ashton Tate. El Dbase IV no se mostró como una herramienta adecuada, y los desencuentros comerciales entre Sybase, Microsoft e IBM (en aquel momento socia de Microsoft para el OS/2) hicieron el resto.

La situación era muy diferente en 1994, cuando los principales fabricantes tradicionales (Informix, Oracle, Sybase) habían lanzado al mercado poderosos servidores y, a ellos, se agregaba IBM que estaba lanzando su producto DB2 para, prácticamente, todos los sistemas operativos importantes (además de sus clásicos MVS y VM, ahora anunciaba AIX, OS/2,Windows NT, Hewlett Packard's UNIX, Sun´s UNIX, Siemens' UNIX, etc) y Microsoft que, luego de finalizar su acuerdo con Sybase, partió para su propio SQL Server para Windows NT.

Existía un conjunto de lenguajes "front end" como, por ejemplo, Delphi, Foxpro, Powerbuilder, SQL Windows, Visual Basic, etc se decía en aquel momento que Visual Basic, más allá de sus méritos intrínsecos como lenguaje, era el favorito para dominar el mercado, cosa que está ocurriendo.

Por otra parte, existían en la comunidad informática muchas dudas sobre la calidad de los optimizadores de los sistemas de gerencia de base de datos, cuyas fallas del pasado habían sido causantes de verdaderas historias de horror.

¿Qué ha ocurrido en los últimos años? Que los servidores se han mostrado sólidos y eficientes, que sus optimizadores probaron, en general, ser excelentes. Que una cantidad muy importante de empresas, en todo el mundo, ha encarado aplicaciones Cliente/Servidor, y que las que lo están haciendo con los planes necesarios y con las herramientas adecuadas, están obteniendo éxitos muy importantes, mientras que los que lo han hecho desaprensivamente, han cosechado fracasos.

La tecnología de los servidores de base de datos ha evolucionado mucho en los últimos años y todos los fabricantes trabajan con tecnología sensiblemente equivalente. Parecen mucho más importantes para la elección elementos que están fuera de la tecnología: la confianza que nos despierta el fabricante, su compromiso con el producto, su tendencia a mantenerse siempre actualizado, su situación económico/financiera, las garantías que nos brinde el soporte local y, en menor medida, el precio.

Aunque inicialmente fueron los propios usuarios quienes impulsaron esta nueva tecnología, la situación ha cambiado drásticamente. Hoy en día, el modelo Cliente/Servidor se considera clave para abordar las necesidades de las empresas. El proceso distribuido se reconoce actualmente como el nuevo paradigma de sistemas de información, en contraste con los sistemas independientes. Este cambio fundamental ha surgido como consecuencia de importantes factores (negocio, tecnología, proveedores), y se apoya en la existencia de una gran variedad de aplicaciones estándar y herramientas de desarrollo, fáciles de usar que soportan un entorno informático distribuido.

Los ordenadores personales y los paquetes de software de aplicaciones proliferan comercialmente. Estos ordenadores están conectados a las Redes de Area Local (LAN), mediante las cuales, los grupos de usuarios y profesionales comparten aplicaciones y datos. Las nuevas tecnologías de distribución de funciones y datos en una red, permiten desarrollar aplicaciones distribuidas de una manera transparente, de forma que múltiples procesadores de diferentes tipos, puedan ejecutar partes distintas de una aplicación.

Conceptos Cliente/Servidor

La arquitectura cliente-servidor se creó básicamente para un gran número de estaciones de trabajo, computadores personales, servidores de archivos, impresoras y otros equipos que estén interconectados a través de una red. Este tipo de arquitectura tiene como idea principal definir servidores especializados con funciones especificas en donde los recursos que proveen dichos servidores estén a disposición de muchos clientes.

Esta arquitectura se está incorporando cada vez más en los paquetes de Sistemas de Gestión de Bases de Datos (SGBD) comerciales, los cuales dividen el software de SGBD en dos niveles: cliente y servidor, de este modo algunos sitios pueden ejecutar exclusivamente el software de cliente mientras que otros sitios son máquinas dedicadas que ejecutan sólo el software del servidor o bien pueden existir sitios que ejecuten módulos tanto de cliente como de servidor.

Como aún no hay una forma precisa de dividir la funcionalidad del SGBD entre cliente y servidor, varios productos de SGBD relacionales han adoptado la posibilidad de incluir la funcionalidad de un SGBD centralizado en el nivel de servidor, en el que se proporciona un servidor SQL a los clientes y cada cliente formula sus consultas SQL apropiadas y provee la interfaz con el usuario y las funciones de interfaz con los lenguajes de programación. En este enfoque el servidor SQL recibe el nombre de procesador de base de datos en tanto que el cliente se denomina procesador de aplicaciones.

El concepto de cliente/servidor proporciona una forma eficiente de utilizar todos estos recursos de máquina, de tal forma que la seguridad y fiabilidad que proporcionan los entornos mainframe se traspasa a la red de área local, añadiendo a esto la ventaja de la potencia y simplicidad de los ordenadores personales.

El término cliente/servidor describe un sistema en el que una máquina cliente solicita a una segunda máquina llamada servidor que ejecute una tarea especifica. El cliente suele ser una computadora personal común conectada a una LAN, y el servidor es, por lo general, una máquina anfitriona, como un servidor de archivos PC, un servidor de archivos de UNIX o una macrocomputadora o computadora de rango medio.

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos, y servidor al proceso que responde a las solicitudes. Los principales componentes del esquema cliente/servidor son entonces los Clientes, los Servidores y la infraestructura de comunicaciones.

En este modelo, las aplicaciones se dividen de forma qué el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario. Los Clientes interactúan con el usuario, usualmente en forma gráfica y frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad. El programa cliente cumple dos funciones distintas:

- Por un lado gestiona la comunicación con el servidor, solicita un servicio y recibe los datos enviados por aquél.
- Por otro, maneja la interfaz con el usuario: presenta los datos en el formato adecuado y brinda las herramientas y comandos necesarios para que el usuario pueda utilizar las prestaciones del servidor de forma sencilla.

El programa servidor en cambio, básicamente sólo tiene que encargarse de transmitir la información de forma eficiente. No tiene que atender al usuario. De esta forma un mismo servidor puede atender a varios clientes al mismo tiempo.

Los clientes realizan generalmente funciones como:

- Manejo de la interface del usuario.
- Captura y validación de los datos de entrada
- Generación de consultas e informes sobre las bases de datos

Los Servidores proporcionan un servicio al cliente y devuelven los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente,

verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente.

Además, deben manejar los interbloqueos, la recuperación ante fallas, y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes Por esta razón se utilizan PCs poderosas estaciones de trabajo, minicomputadores o sistemas grandes. Además deben manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema ("login"), auditoría y recuperación, y contabilidad.

Habitualmente en los servidores existe algún tipo de servicio de bases de datos En ciertas circunstancias, este término designará a una máquina. Los servidores realizan, entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.
- Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste, le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo

La interacción que existe entre el cliente y el servidor durante el procesamiento de una consulta se puede llevar como lo muestra la Figura 4

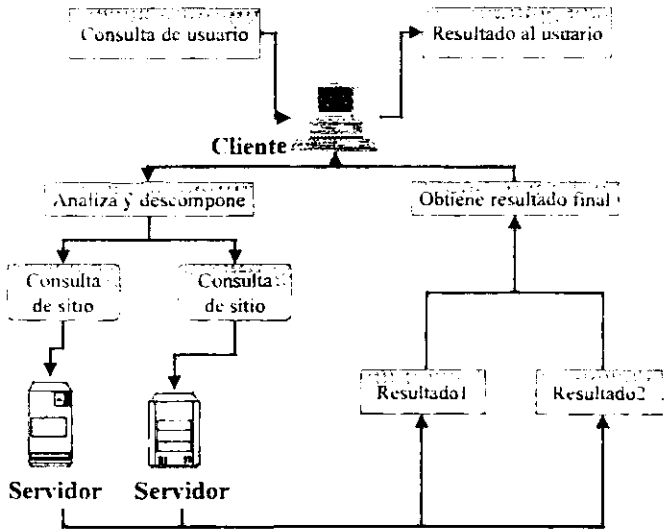


Figura 4. Funciones Cliente-Servidor

En donde:

1. El cliente analiza sintácticamente la consulta del usuario y la descompone en varias consultas de sitio independientes.
2. Cada consulta de sitio se envía al sitio del servidor apropiado.
3. Cada servidor procesa la consulta local y envía el resultado al sitio cliente.
4. El sitio cliente combina los resultados de las subconsultas para producir el resultado de la consulta original y mostrarla al usuario.

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura de comunicaciones, la cual proporciona los mecanismos básicos de direccionamiento y transporte. La mayoría de los sistemas Cliente/Servidor actuales, se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo cual implica que las

aplicaciones deben hacer las verificaciones. La red debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración

Características de la Arquitectura Cliente/Servidor

Entre las principales características de la arquitectura cliente/servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interface única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interface externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Características Funcionales

Existe un conjunto de variantes de la Arquitectura Cliente/Servidor, dependiendo de dónde se ejecutan los diferentes elementos involucrados:

A. Administración de los datos
B. Lógica de la aplicación
C. Lógica de la presentación.

De acuerdo a esto la arquitectura se puede clasificar en cinco niveles, según las funciones que asumen el cliente y el servidor:

- *Primer Nivel*

Aquí el cliente asume parte de las funciones de presentación de la aplicación, ya que siguen existiendo programas en el servidor, dedicados a esta tarea. Dicha distribución se realiza mediante el uso de productos para el "maquillaje" de las pantallas del mainframe. Esta técnica no exige el cambio en las aplicaciones orientadas a terminales, pero dificulta su

mantenimiento. Además, el servidor ejecuta todos los procesos y almacena la totalidad de los datos. En este caso se dice que hay una presentación distribuida.

Es en la presentación distribuida, donde tanto la administración de los datos, como la lógica de la aplicación, funcionan en el servidor y la lógica de la presentación se divide entre el servidor (parte preponderante) y el cliente (donde simplemente se muestra)
Esta alternativa es extremadamente simple, porque generalmente no implica programación alguna. Con ello se obtiene una mejor presentación, desde el punto de vista estrictamente cosmético, y ciertas capacidades mínimas para vincular las transacciones clásicas con el entorno Windows. Desde el punto de vista del uso de los recursos, este nivel es similar a una Arquitectura Centralizada. La presentación distribuida tiene como único objetivo poder seguir utilizando sin cambios, aplicaciones desarrolladas para una arquitectura centralizada, y aporta ciertas contribuciones en lo relativo a la cosmética y a cierta conectividad, bastante limitada, con aplicaciones usuales del ambiente Windows. Es un recurso muy modesto, y sólo puede justificarse como transitorio, mientras se desarrollan verdaderas aplicaciones Cliente/ Servidor.

- *Segundo Nivel.*

En este la aplicación está soportada directamente por el servidor, excepto la presentación que es totalmente remota y reside en el cliente. Los terminales del cliente soportan la captura de datos, incluyendo una validación parcial de los mismos y una presentación de las consultas. En este caso se dice que hay una presentación remota.
En la Administración de Datos Remota, dicha administración de los datos se hace en el servidor, mientras que tanto la lógica de la aplicación, como la de la presentación, funcionan en el Cliente. Desde el punto de vista de las necesidades de potencia de procesamiento, esta variante es la óptima. Se minimiza el costo del procesamiento en el Servidor (sólo se dedica a administrar la base de datos, no participando en la lógica de la aplicación que, cada vez, consume más recursos), mientras que se aumenta en el cliente, donde es irrelevante, teniendo en cuenta las potencias de Cliente necesarias, de todas maneras, para soportar el sistema operativo Windows.

El otro elemento a tener en cuenta es el tránsito de datos en la red. Esta variante podrá ser óptima, buena, mediocre o pésima, de acuerdo a este tránsito. En el caso de transacciones o

consultas activas, donde prácticamente todos los registros seleccionados son necesarios para configurar las pantallas a mostrar, este esquema es óptimo.
Por otro lado, en el caso de programas "batch", donde en realidad no se muestra nada, esta alternativa es teóricamente indefendible (no obstante, si el cliente está ligado al servidor por una red de alta velocidad, los resultados prácticos, a menudo, son aceptables).
Una variante interesante es la de complementar el procesamiento en el cliente con procesamiento en el servidor.

- *Tercer Nivel.*

En este nivel, la lógica de los procesos se divide entre los distintos componentes del cliente y del servidor. El diseñador de la aplicación debe definir los servicios y las interfaces del sistema de información, de forma que los papeles de cliente y servidor sean intercambiables, excepto en el control de los datos, que es responsabilidad exclusiva del servidor. En este tipo de situaciones se dice que hay un proceso distribuido o cooperativo.

- *Cuarto Nivel.*

El cliente realiza tanto las funciones de presentación como los procesos. Por su parte, el servidor almacena y gestiona los datos que permanecen en una base de datos centralizada. En esta situación se dice que hay una gestión de datos remota.

- *Quinto Nivel.*

Aquí el reparto de tareas es como en el anterior y además el gestor de base de datos divide sus componentes entre el cliente y el servidor. Las interfaces entre ambos, están dentro de las funciones del gestor de datos y, por lo tanto, no tienen impacto en el desarrollo de las aplicaciones. En este nivel se da lo que se conoce como bases de datos distribuidas.

Características Físicas

Los niveles anteriormente descritos dan una idea de la estructura física de conexión entre las distintas partes que componen una arquitectura cliente/servidor. Esta idea consiste en aprovechar la potencia de los ordenadores personales para realizar los servicios de presentación y algunos procesos o incluso algún acceso a datos locales de acuerdo al nivel

en que se clasifique. De esta forma se descarga al servidor de ciertas tareas para que pueda realizar otras más rápidamente.

Existe una plataforma de servidores que sustituye al ordenador central tradicional y que da servicio a los clientes autorizados. Incluso a veces el antiguo ordenador central se integra en dicha plataforma como un servidor más. Estos servidores suelen estar especializados por funciones y dependiendo de las dimensiones de la instalación se pueden reunir en un servidor una o varias de estas funciones.
Las unidades de almacenamiento masivo en esta arquitectura, se caracterizan por incorporar elementos de protección que evitan la pérdida de datos y permiten multitud de accesos simultáneos
Para la comunicación de todos estos elementos se emplea un sistema de red que se encarga de transmitir la información entre clientes y servidores. Físicamente consiste en un cableado o en conexiones mediante señales de radio o infrarrojas, dependiendo de que la red sea: local, metropolitana o de área extensa.

Para la comunicación de los procesos con la red se emplea un tipo de equipo lógico denominado middleware[15] que controla las conversaciones. Su función es independizar ambos procesos (cliente y servidor). La interface que presenta es la estándar de los servicios de red, hace que los procesos "piensen" en todo momento que se están comunicando con una red.

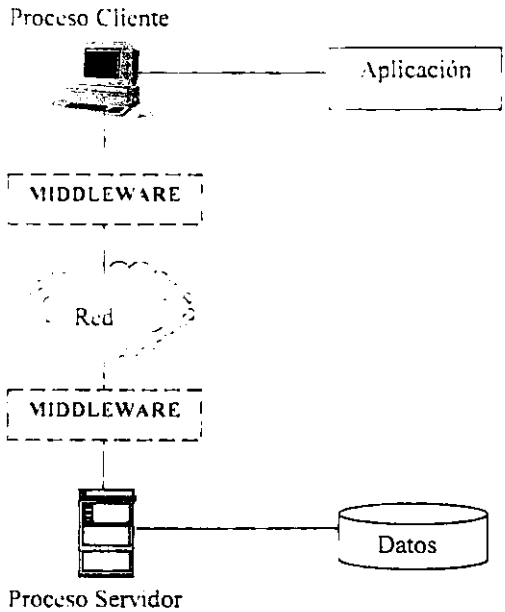


Figura 5. Middleware

Características Lógicas

De entre las principales aportaciones de esta arquitectura a los sistemas de información se encuentra la interface gráfica de usuario. Gracias a ella se dispone de un manejo más fácil e intuitivo de las aplicaciones mediante el uso de un dispositivo tipo ratón. En esta arquitectura los datos se presentan, editan y validan en la parte de la aplicación cliente.

En cuanto a los datos, cabe señalar que en la arquitectura cliente/servidor se evitan las duplicidades teniendo siempre una imagen única y correcta de los mismos, disponible en línea para su uso inmediato. Todo esto tiene como fin que el usuario de un sistema de

[1] Es un modulo intermedio que actúa como conductor entre dos modulos de software

información soportado por una arquitectura cliente/servidor, trabaje desde su estación de trabajo con distintos datos y aplicaciones, sin importarle dónde están o dónde se ejecuta cada uno de ellos.

Importancia del Middleware en soluciones Cliente/Servidor

El middleware es un módulo intermedio que actúa como conductor entre dos módulos de software, para compartir datos, los dos módulos de software no necesitan saber cómo comunicarse entre ellos, sino cómo comunicarse con el módulo de middleware.

El middleware debe ser capaz de traducir la información de una aplicación y pasarla a la otra. En una aplicación cliente/servidor el middleware reside entre la aplicación cliente y la aplicación del sistema host que actúa como servidor.

Características Generales

- Simplifica el proceso de desarrollo de aplicaciones.
- Es el encargado del acceso a los datos: acepta las consultas y datos recuperados directamente de la aplicación y los transmite por la red. También es responsable de enviar de vuelta a la aplicación, los datos de interés y de la generación de códigos de error.
- Es diferente desarrollar aplicaciones en un entorno middleware que la utilización de APIs[16] directas del sistema. El middleware debe ser capaz de manejar todas las facilidades que posee el sistema operativo y ésto, no es sencillo. Por eso, muchas veces se pierde potencia con la utilización del middleware en lugar de las APIs del sistema operativo directamente.
- La adopción dentro de una organización implica la utilización de unos paquetes de software específicos para desarrollar estos módulos. Esto liga a un suministrador y a su política de actualización del producto, que puede ser distinta que la de actualización de los sistemas operativos con los que se comunica el módulo middleware.

Campos de Aplicación

- Migración de los Sistemas Host. Rediseño de Aplicaciones

La aplicación debería diseñarse en base a módulos intermedios middleware, encargados de la comunicación entre el ordenador personal y el host. Esto permite desarrollar hoy la aplicación, sin tener en cuenta los futuros cambios tecnológicos que puedan sufrir los sistemas host. Si el sistema host cambia, o las aplicaciones de host se migran a plataformas de ordenadores personales, todo lo que se necesita es un nuevo módulo middleware. La interface de usuario, la lógica y el código interno permanecen sin cambios.
Por ejemplo, si el equipo lógico del sistema host se traslada desde el mainframe a una base de datos de plataforma PC ejecutándose en un servidor de ficheros, sólo hay que sustituir el módulo de middleware de forma que realice llamadas SQL.

- Arquitectura cliente/servidor

El concepto de middleware permite también independizar los procesos cliente y servidor.
Siempre que las funciones y los objetos que se definan en el módulo intermedio middleware se basen en el flujo de actividades que realiza el usuario, éstos son válidos independientemente del entorno. Por eso, si se mantiene ese módulo separado puede servir para desarrollos futuros.

El middleware es una herramienta adecuada de solución, ya que no sólo es flexible y segura, sino que también protege la inversión en tecnología y permite manejar diferentes ambientes de computación.

Entre los beneficios del uso de la tecnología Cliente/Servidor están:

- Alta productividad de los desarrolladores.
- Bajo costo/alto rendimiento de las plataformas de sistemas y de las redes de comunicación.
- Incremento en la habilidad para construir y entregar soluciones más efectivas para el negocio.

[14] Application Programmer Interface. Lenguaje y formato de mensaje utilizados por un programa para activar e interactuar con las funciones de otro programa o de un equipo físico.

Sin embargo, el reto está en poder construir soluciones Cliente/Servidor, que logren pasar la barrera del simplemente alcanzar la información.

Ventajas e Inconvenientes de la Arquitectura Cliente/Servidor

Ventajas

- Aumento de la productividad:
  - Los usuarios pueden utilizar herramientas que le son familiares, como hojas de cálculo y herramientas de acceso a bases de datos.
  - Mediante la integración de las aplicaciones cliente/servidor con las aplicaciones personales de uso habitual, los usuarios pueden construir soluciones particularizadas que se ajusten a sus necesidades cambiantes.
  - Una interface gráfica de usuario consistente, reduce el tiempo de aprendizaje de las aplicaciones.·
- Menores costos de operación:
  - La existencia de plataformas de hardware cada vez más baratas constituye una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas grandes.
  - Permiten un mejor aprovechamiento de los sistemas existentes, protegiendo la inversión. Por ejemplo, la compartición de servidores (habitualmente caros) y dispositivos periféricos (como impresoras) entre máquinas clientes, permite un mejor rendimiento del conjunto.
  - Se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.

- Proporcionan un mejor acceso a los datos. La interface de usuario ofrece una forma homogénea de ver el sistema, independientemente de los cambios o actualizaciones que se produzcan en él y de la ubicación de la información

- El movimiento de funciones desde un ordenador central hacia servidores o clientes locales, origina el desplazamiento de los costos de ese proceso hacia máquinas más pequeñas y por tanto, más baratas.

- Mejora en el rendimiento de la red:
  - Las arquitecturas cliente/servidor eliminan la necesidad de mover grandes bloques de información por la red hacia los ordenadores personales o estaciones de trabajo para su proceso. Los servidores controlan los datos, procesan peticiones y después transfieren sólo los datos requeridos a la máquina cliente. Entonces, la máquina cliente presenta los datos al usuario mediante interfaces amigables. Todo ésto reduce el tráfico de la red, lo que facilita que pueda soportar un mayor número de usuarios.

  - Se puede integrar PCs con sistemas medianos y grandes, sin que todas las máquinas tengan que utilizar el mismo sistema operacional.

  - Si se utilizan interfaces gráficas para interactuar con el usuario, el esquema Cliente/Servidor presenta la ventaja, con respecto a uno centralizado, de que no es siempre necesario transmitir información gráfica por la red, pues ésta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.

  - Tanto el cliente como el servidor pueden escalar para ajustarse a las necesidades de las aplicaciones.

  - En una arquitectura como ésta, los clientes y los servidores son independientes los unos de los otros, con lo que pueden renovarse para aumentar sus funciones y capacidad de forma independiente, sin afectar al resto del sistema.

  - La arquitectura modular de los sistemas cliente/servidor, permite el uso de ordenadores especializados (servidores de base de datos, servidores de ficheros, etc.)

- Permite centralizar el control de sistemas que estaban descentralizados, como por ejemplo la gestión de los ordenadores personales que antes estuvieron aislados.

- Es más rápido el mantenimiento y el desarrollo de aplicaciones, pues se pueden emplear las herramientas existentes (por ejemplo los servidores de SQL).

- El esquema Cliente/Servidor contribuye además a proporcionar a las diferentes direcciones de una institución soluciones locales, pero permitiendo además la integración de la información relevante a nivel global.

- Las arquitecturas cliente/servidor permiten aprovechar los conceptos de cliente y servidor para desarrollar aplicaciones que accedan a diversas bases de datos, de forma transparente. Esto hace viable cambiar la aplicación en la parte servidora, sin que la aplicación cliente se modifique. Para que sea posible desarrollar estas aplicaciones, es necesaria la existencia de un estándar de conectividad abierta que permita a los ordenadores personales y estaciones de trabajo, acceder de forma transparente a bases de datos corporativas heterogéneas.

Inconvenientes

- Hay una alta complejidad tecnológica al tener que integrar una gran variedad de productos. Por una parte, el mantenimiento de los sistemas es más difícil pues implica la interacción de diferentes partes de hardware y de software, distribuidas por distintos proveedores, lo cual dificulta el diagnóstico de fallas.

- Requiere un fuerte rediseño de todos los elementos involucrados en los sistemas de información (modelos de datos, procesos, interfaces, comunicaciones, almacenamiento de datos, etc). Además, en la actualidad existen pocas herramientas que ayuden a determinar la mejor forma de dividir las aplicaciones entre la parte cliente y la parte servidor. Por un lado, es importante que los clientes y los servidores utilicen el mismo mecanismo, lo cual implica que se deben tener mecanismos generales que existan en diferentes plataformas. Además, se cuenta con muy escasas herramientas para la administración y ajuste del desempeño de los sistemas.

- Es más difícil asegurar un elevado grado de seguridad en una red de clientes y servidores que en un sistema con un único ordenador centralizado. Se deben hacer verificaciones en el cliente y en el servidor. También se puede recurrir a otras técnicas como el encriptamiento. Un aspecto directamente relacionado con esto, es el de cómo distribuir los datos en la red.

- A veces, los problemas de congestión de la red pueden degradar el rendimiento del sistema por debajo de lo que se obtendría con una única máquina (arquitectura centralizada). También la interface gráfica de usuario puede a veces ralentizar el funcionamiento de la aplicación.

- El quinto nivel de esta arquitectura (bases de datos distribuidas) es técnicamente muy complejo y en la actualidad, hay muy pocas implantaciones que garanticen un funcionamiento totalmente eficiente.

- Existen multitud de costos ocultos (formación en nuevas tecnologías, licencias, cambios organizativos, etc.) que encarecen su implantación.

CAPITULO 4. CASO PRACTICO

4.1 Definición del Problema

El área de Servicio a Clientes de tarjeta de crédito (SAC) del edifico corporativo de esta institución financiera, solicita instalar una plataforma que permita realizar la función de ventas por teléfono para futuras promociones, considerando con ello un ahorro en la contratación de compañías externas Esta área debe contar con la infraestructura necesaria para realizar vía PC el Telemarketing "Artículos Promocionales".

El proceso de comercialización se podrá realizar por dos medios:

1. Redención de puntos efectivos: el costo del artículo equivalente al número de puntos correspondientes. El cargo se realizará a los puntos disponibles con abono a la cuenta de cheques del fideicomiso (comercio).
2. Tarjeta de Crédito (TDC): en este caso el cargo será por la cantidad original, previa autorización del sistema.

De acuerdo al contrato de filiación de venta telefónica, existe cierta información que debe contener el levantamiento de pedido como es:

1. Nombre del tarjetahabiente
2. Número de cuenta de la tarjeta
3. Fecha de inicio de la tarjeta
4. Fecha de vencimiento de la tarjeta
5. Domicilio que aparece en el estado de cuenta del tarjetahabiente y domicilio de entrega.
6. Teléfonos del tarjetahabiente (domicilio y oficina)
7. Nombres de las personas susceptibles de recibir el bien o servicio en el domicilio del tarjetahabiente (máximo dos personas)
8. Día y hora de la solicitud del pedido del tarjetahabiente
9. Descripción del producto o servicio

10. Monto de la operación
11. Clave de autorización
12. Nombre del asesor
13. Clave de pedido, solicitud de servicio o contraseña
14. Nombre, domicilio y teléfono de el afiliado
15. Número de afiliación del afiliado
16. Acuse de recibo

    Es necesario se anexe acuse de recibo del producto o servicio incluyendo. nombre del tarjetahabiente, clave, número de pedido o contraseña. domicilio del tarjetahabiente, domicilio de entrega, número de guía, nombre y firma de la persona autorizada para recibir el producto. fecha de entrega, datos de identificación oficial (credencial de elector y/o pasaporte).

    Es importante mencionar que el cliente afiliado deberá apegarse a toda esta información, para eliminar la posibilidad de contracargos por ventas no reconocidas

Es importante tomar en cuenta que debe existir la posibilidad de que el cliente solicite factura, a lo cual deberá complementarse el pedido con la siguiente información:

17. Factura a nombre de:
18. Dirección
19. Registro Federal de Causantes (RFC)
20. Precio de venta

También es importante se maneje un contador en linea de artículos disponibles en bóveda, el cual debe registrar cada salida una vez fincado el pedido, así mismo deberá contener el registro de entradas a bóveda por devoluciones o cancelaciones.

Una vez que se haya realizado la venta será necesario que tanto el área de distribución como el área de telemarketing cuente con algún sistema que permita consultar la situación en la que se encuentra la distribución del pedido, para poder dar respuestas a los tarjetahabientes que requieran este tipo de información y para que distribución lleve un adecuado control de la distribución de estos pedidos.

4.2 Forma de Trabajo Actual

Actualmente SAC cubre un horario de atención 7 por 24 (lunes a domingo las 24 horas del día) y se conforma por células de trabajo que se encuentran integradas por diez asesores telefónicos y un supervisor. Estas células cuentan con el siguiente equipo para poder desempeñar sus funciones:

- PC

Cada estación de trabajo se encuentra conectada a la red local y se comunican por Ethernet protocolo TCP/IP a la casa matriz, lo que le permite tener la comunicación necesaria con los servidores en los que se encuentran las aplicaciones que le permiten al asesor brindar un servicio oportuno y de gran calidad para el cliente. Estas se basan en Windows 95.

- ACD

El ACD (Distribución Automática de Llamadas) es otro elemento importante que distribuye entre los agentes del call center las llamadas, de acuerdo a las cargas de trabajo y la capacitación de cada agente.

- Diadema

Esta diadema se conecta al ACD lo que permite al asesor escuchar por medio de ésta la conversación que se tiene con el tarjetahabiente sin necesidad de emplear las manos, éstas tienen la función de un auricular tradicional.

- IVR

Aunque este componente no se encuentra físicamente en el módulo de trabajo del agente, el IVR (Sistema Interactivo de Respuesta) es un sistema importante cuya labor es sustituir el trabajo de los operadores mediante una interfaz automática, que se basa en grabaciones y cuya interacción de quien llama se da por medio del teclado telefónico[17]. Esto permite reducir el número de llamadas enviadas a los asesores y reducir el tiempo en espera de otros clientes, ya que un gran porcentaje de llamadas entrantes en el call center pueden resolverse satisfactoriamente para el cliente en el IVR.

- CTI

Integración de Cómputo-Telefonía, este término se emplea para designar una computadora conectada a un conmutador telefónico. Su función es tener aplicaciones como Pop Screen, la cual consiste en que, cuando entra una llamada al call center y es transferida a un asesor al mismo tiempo le llega la información sobre el cliente que llama. Esto permite al operador tener toda la información necesaria para atender la llamada sin necesidad de que el cliente vuelva a proporcionar los datos que ya le fueron solicitados por el IVR.

- IG

Interfaz Gráfica. Este es un sistema que fue desarrollado por el área de Sistemas Servicio a clientes y Sistema Distribuido. Esta aplicación se encuentra desarrollada en Visual Basic 3.0 y cuenta con una base de datos en SQL Server 6.5. La función principal para la que se desarrollo es contar con toda la información del TH (número de cuenta, saldos, transacciones, tarjetas robadas, extraviadas, puntos, plásticos, promociones, estados de cuenta).

Expresando esto en forma gráfica, la operación del día a día dentro de la institución es como se muestra en el diagrama de la Figura 6.

[17] Actualmente ya existen sistemas que se activan por medio de la voz, sin embargo en la institución financiera a que se hace referencia aún no se encuentra implementado este tipo de soluciones.

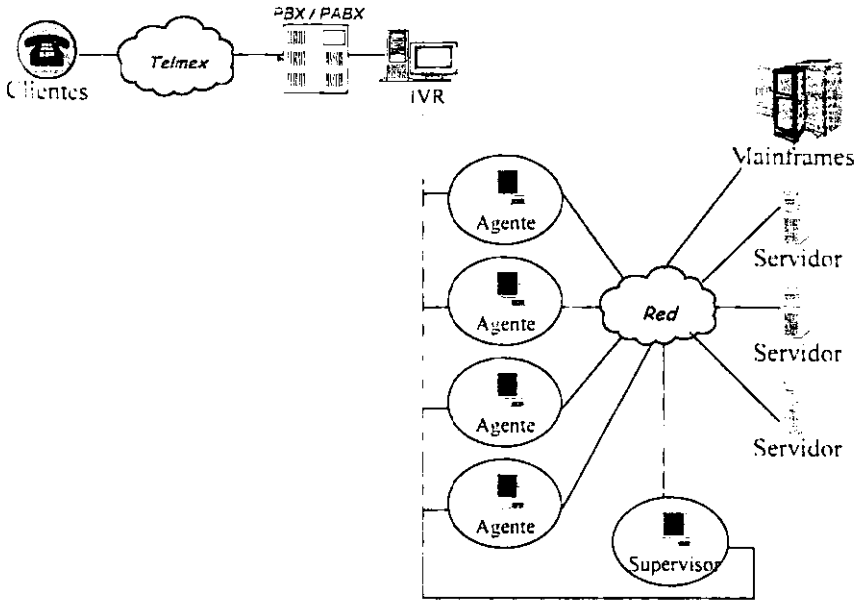


Figura 6. Diagrama Funcional SAC

Sin embargo es importante hacer notar los siguientes puntos:

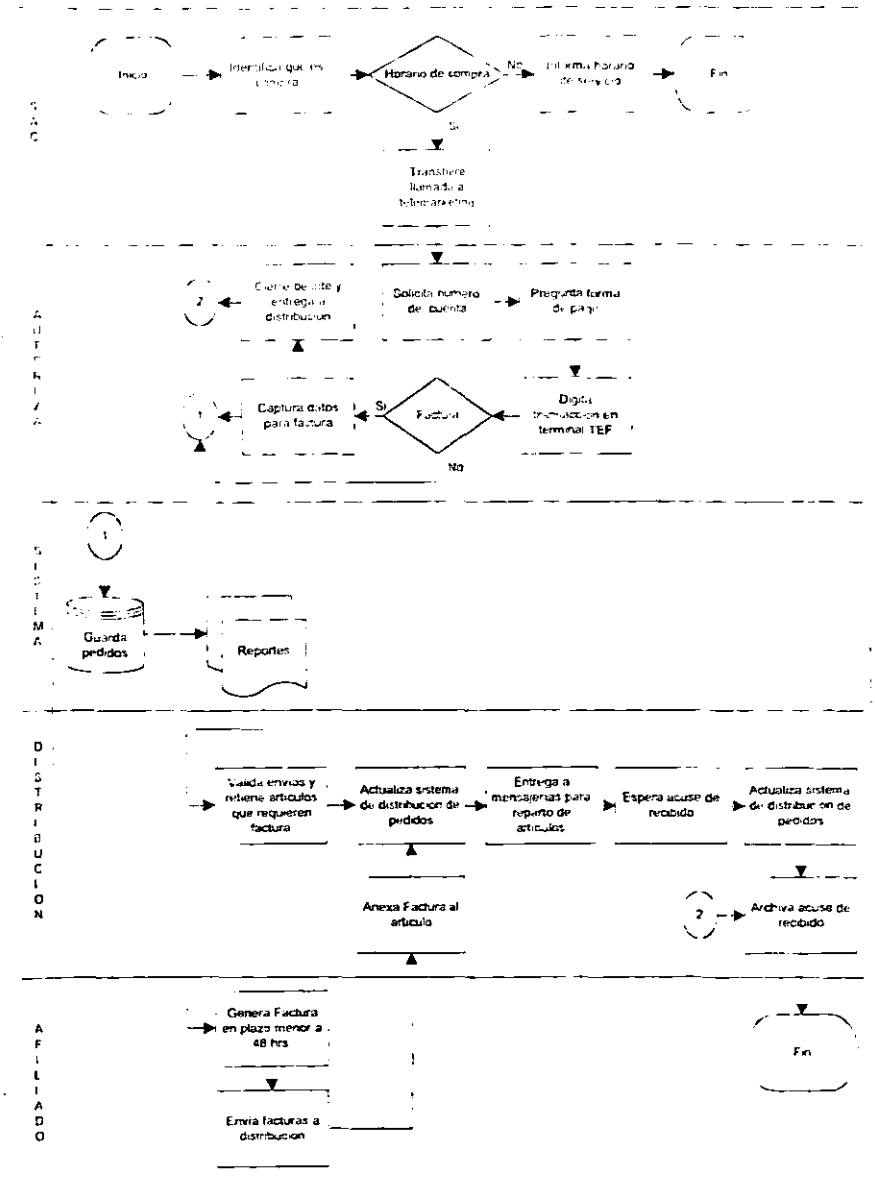
1. SAC no cuenta con una infraestructura de venta telefónica.
2. No existe una opción dentro de la IG (Interfaz Gráfica) que permita capturar la información de los clientes por concepto de venta en el área de SAC.
3. No existe aplicación que permita manejar la situación para el control de pedidos tanto en el área de SAC como en el área de Distribución.

4.3 Planteamiento de la solución

Es importante resaltar que en el presente trabajo únicamente me abocaré a plantear la solución que permitirá contar con un sistema de control de pedidos para el área de distribución. De acuerdo a esa definición del problema se ha determinado que es necesario llevar a cabo las siguientes actividades.

1. Anexar una pantalla dentro de la IG que permita capturar la información de los clientes al momento de realizar un pedido de cualquier tipo de artículo
2. Desarrollar una aplicación que proporcione la información correspondiente para conocer la situación del pedido.
3. Diseñar e implementar la base de datos en la que se llevará el registro de los pedidos realizados, para su adecuado control y consulta.
4. Proponer la infraestructura necesaria para la adecuada operación de las áreas mencionadas.

Tomando en cuenta la forma en que actualmente labora SAC también se ha determinado que el flujo que deberá cubrirse para el proceso operativo de ventas y que tendrá que incluirse dentro del sistema que actualmente manejan así como el flujo para la distribución del pedido será el siguiente:

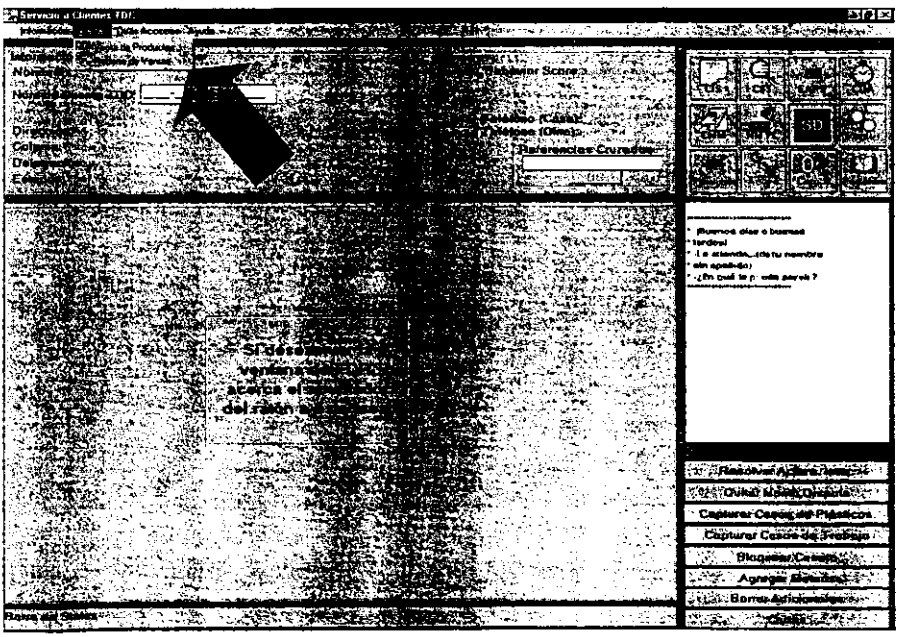

CLIENTE
AUTOFRIVA
SISTEMA
DISTRIBUCION
AFILIADO
Inicio
Identificar qué es lo que compra
Horario de compra
No
Fin
Si
Transfiere llamada a telemarketing
Solicita numero de cuenta
Pregunta forma de pago
Digita transacción en terminal TEF
Factura
Si
No
Captura datos para factura
Guarda pedidos
Reportes
Sacan envíos y rellenan artículos que requieren factura
Actualiza sistema de distribucion de pedidos
Entrega a mensajerías para reparto de artículos
Espera acuse de recibido
Actualiza sistema de distribucion de pedidos
Anexa Factura al artículo
Archiva acuse de recibido
Genera Factura en plazo menor a 48 hrs
Envía facturas a distribucion
Fin

Propuesta para el desarrollo de las aplicaciones (Prototipos)

Una vez concluido el análisis se obtuvo que será necesario realizar dos desarrollos. El primero que es sólo una adecuación a la IG existente y en la cuál no se implementará la base de datos. El segundo que servirá como complemento de la IG y en el cual se podrá controlar adecuadamente el control de pedidos, a diferencia del primero en este si se planteará el diseño de la base de datos que este sistema ocupara para su adecuado funcionamiento. Ambos desarrollos deberán realizarse con Visual Basic 3.0 y SQL Server 6.5.

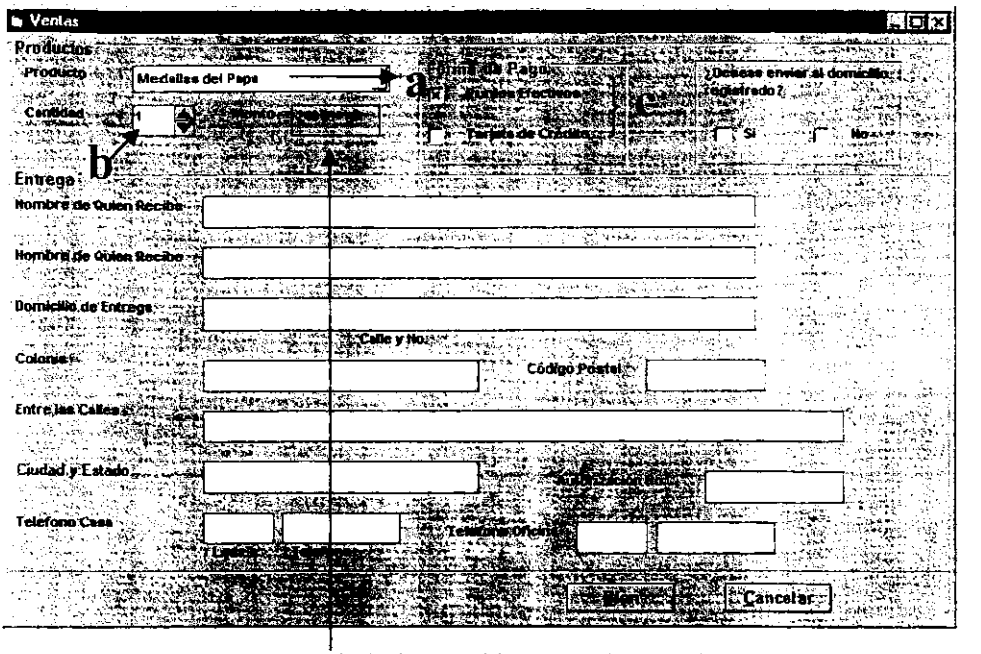
I Adecuación a la IG

Incluir en el sistema actual el menú ventas, con opciones de venta de productos y reporte de ventas.



Al seleccionar la opción de ventas deberá desplegarse una ventana que cuente con los datos necesarios para levantar el pedido, en donde el asesor deberá seleccionar los siguientes datos

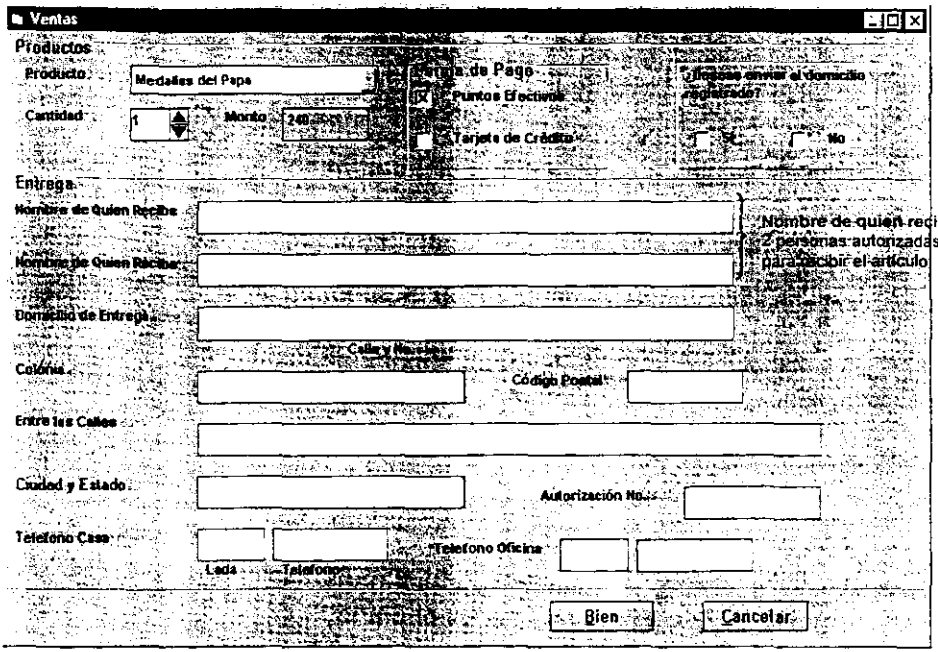
a. Producto
b. Cantidad
c. Forma de pago



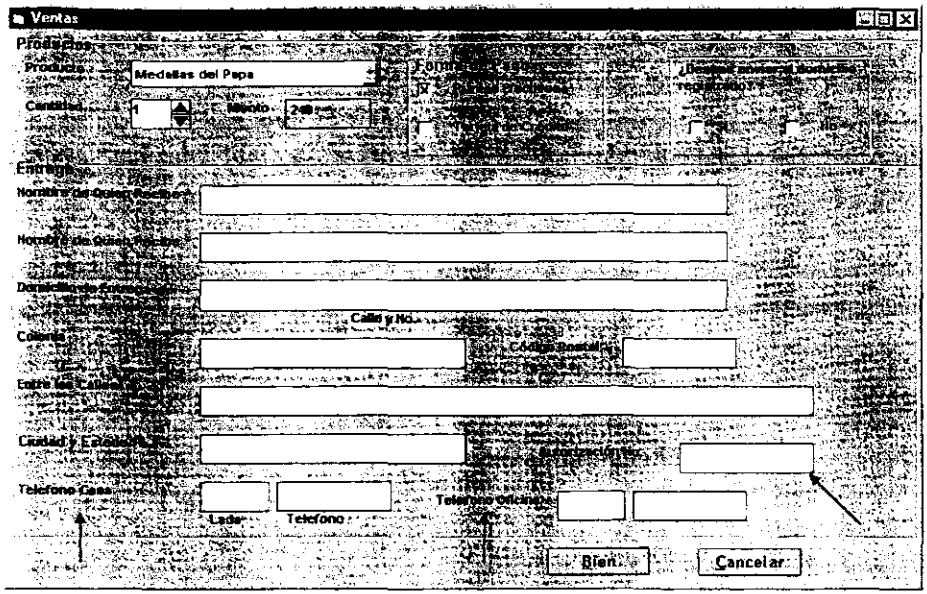
El campo de monto se calculará automáticamente de acuerdo al número de artículos y forma de pago

El campo de domicilio aparecerá prellenado con el domicilio del TH. En caso de haber seleccionado envío al domicilio registrado, de lo contrario los espacios asignados para el domicilio aparecerán en blanco para ser capturados por el asesor.

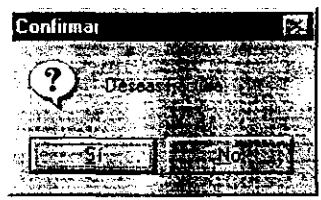
Los nombres de las personas facultadas para recibir el artículo deben ser capturados por el asesor.



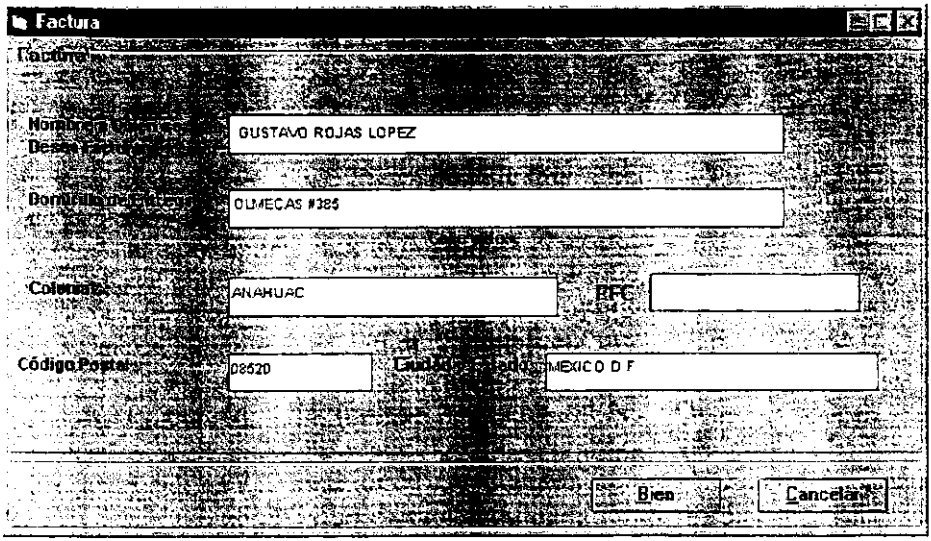
También será necesario que se valide que se encuentren correctos los números telefónicos que aparecen en la pantalla y que es asesor capture el número de autorización que le dio la terminal TEF al pedido y oprimir el botón Bien:



Una vez que se ha aceptado el levantamiento del pedido aparecerá la pantalla de confirmación de factura:



Al responde al mensaje de confirmación con cualquiera de los dos botones se deberá proporcionar al TH el número de folio, confirmar el número de artículos e importe del cargo ya sea a crédito o puntos y únicamente al presionar el botón Si, se desplegará la siguiente pantalla para capturar los datos de la factura.



Respecto a la opción Reporte de ventas, en aspectos generales su emisión debe permitir los siguientes puntos:

- Validar la transmisión de las TEF contra la captura realizada en el sistema de levantamiento de pedidos.
- Informar a la afiliación que emite los artículos las solicitudes de facturación por artículos.
- Informar al área de distribución los artículos que deberán esperar la factura para ser enviados a los clientes y aquellos que podrán enviarse desde el mismo día del levantamiento del pedido.

Además cumplirán con las siguientes especificaciones:

- Su emisión debe ser diaria y contener todas las compras efectuadas con puntos y con TDC.
- Contener por separado las compras que requieren factura y especificar en él:

a) RFC
b) Nombre
c) Dirección

d) Número de artículos comprados
e) Importe de la| factura
f) Forma de pago

- Se debe contar con dos impresiones. Una impresión a un archivo texto que deberá enviarse por e-mail a las afiliaciones que surten el producto y la otra al área de distribución.
- Se contará con reportes por cada una de las siguientes categorías debiendo contener cada uno de ellos los siguientes campos:

a) Por levantamiento de pedido
   1. Fecha en que se realizó la compra del artículo
   2. Monto total de las compras efectuadas con puntos y con TDC por separado y ordenadas por número de cuenta
   3. Clave del asesor
   4. Número de autorización
   5. Número de artículos adquiridos por tipo de compra
   6. Número de cuenta
   7. Forma de pago
   8. Total de autorizaciones efectuadas en el día

b) Por Facturas
   1. Fecha en que se realizó la compra del artículo
   2. Monto total de las compras efectuadas con puntos y con TDC por separado y ordenadas por número de cuenta
   3. Número de cuenta
   4. Forma de pago
   5. RFC
   6. Nombre al que deberá generarse la factura
   7. Domicilio al que se factura
   8. Número de artículos adquiridos por tipo de compra

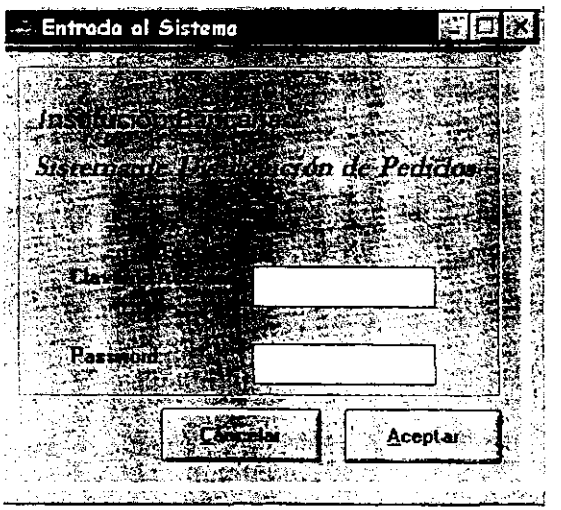
c) Por Distribución
   1. Fecha en que se realizó la compra del artículo
   2. Número de cuenta

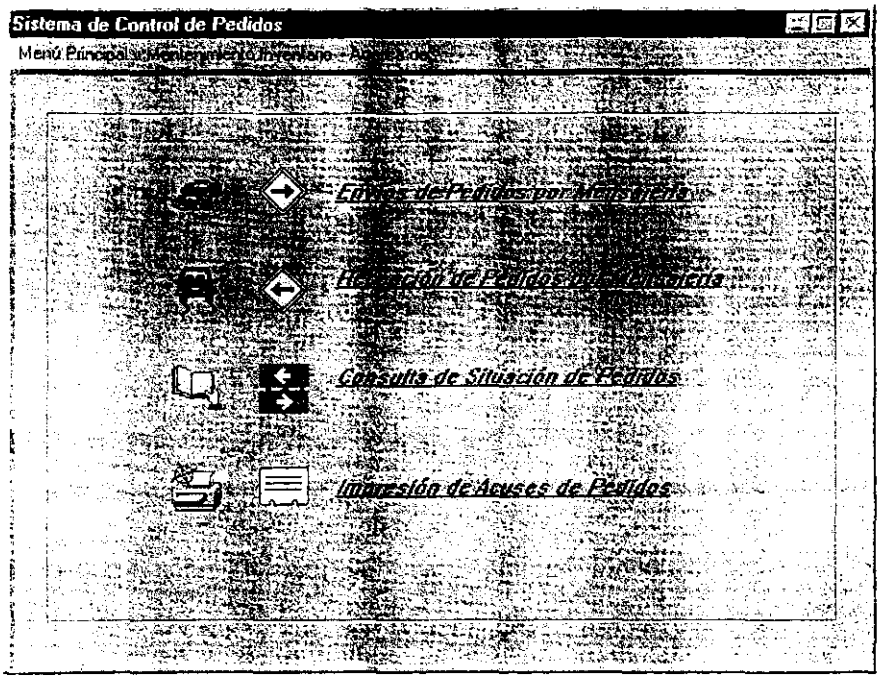
3 Forma de pago
4 Número de artículos adquiridos por tipo de compra
5 Separar las compras que requieren factura e incluir RFC, nombre al que deberá generarse la factura y domicilio al que se factura
6 Número de artículos comprados
7 Importe de la factura
8 Forma de pago (puntos o TDC)

ii. Desarrollo para la distribución de pedidos

Las pantallas prototipo y características que deberán tener para desarrollar el sistema de control de pedidos son las siguientes:

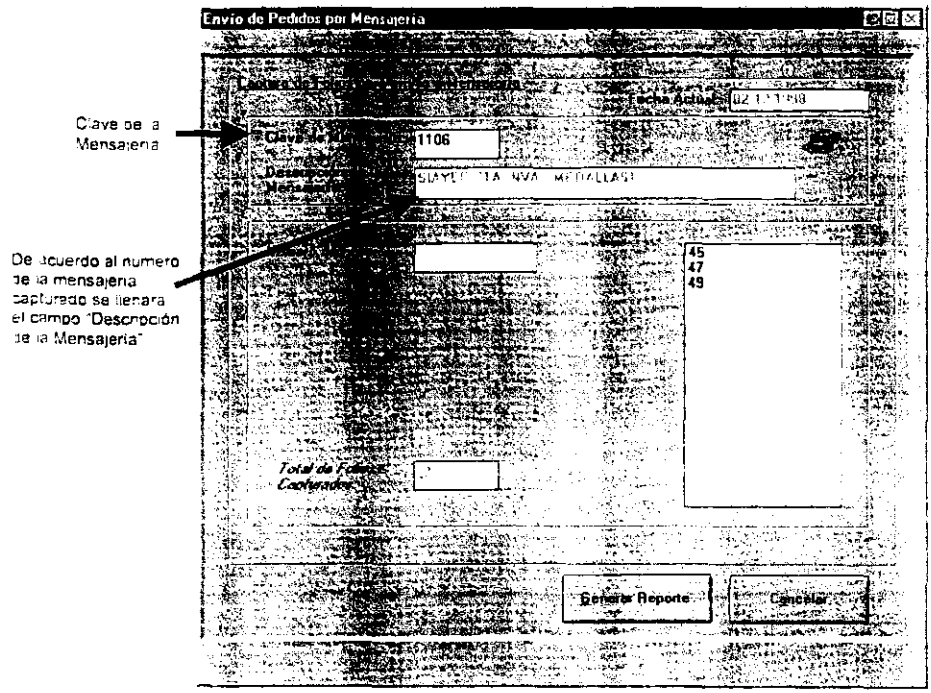
Pantalla principal y entrada al sistema





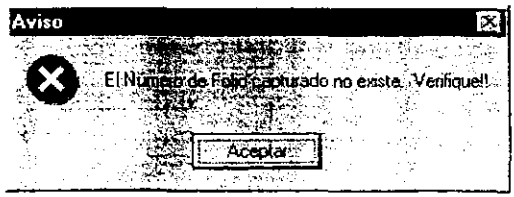
Envio de Pedidos por Mensajerias

Para mantener un control de las entregas que se han hecho a mensajerias se debe capturar la clave de la mensajeria y el número de folio

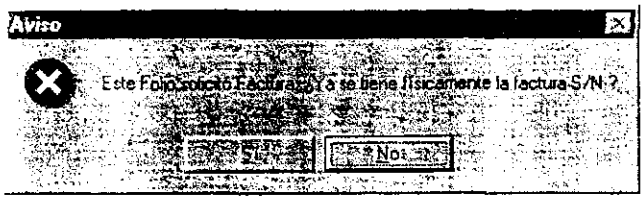


El sistema hará las siguientes validaciones al realizar las capturas.

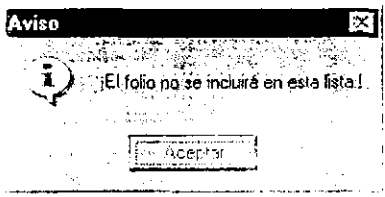
1 Si el número de folio existe como levantamiento de pedido. En caso de no existir se presentará el siguiente mensaje:



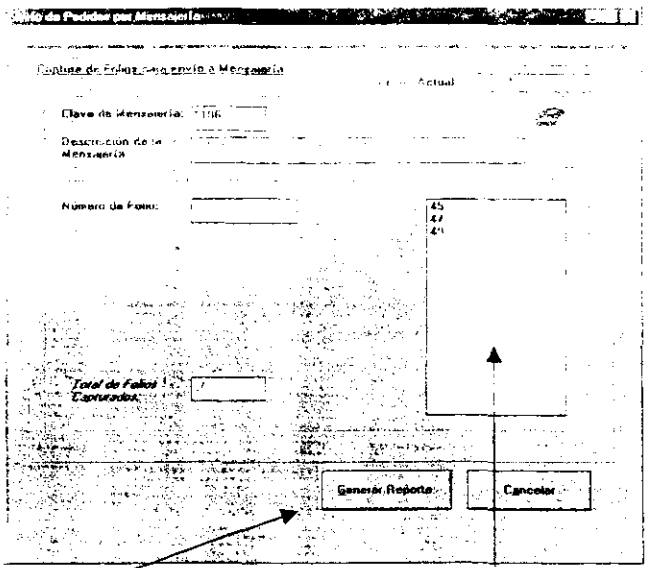
2 Si el folio existe pero en el levantamiento del pedido se solicito factura, deberá presentarse el siguiente mensaje de validación:



Si la respuesta es no, el folio no se incluirá en la lista entregada a las mensajerías para la distribución de los pedidos.



Si la respuesta es sí, procede a capturar el siguiente número de folio.



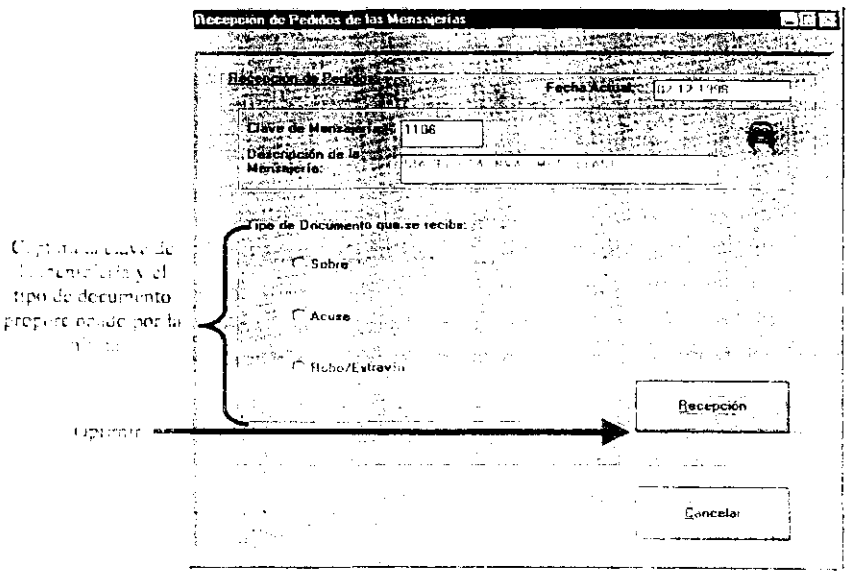
Terminada la captura se genera el reporte de distribución para las mensajerías

Los números de folios aceptados serán listados en el espacio

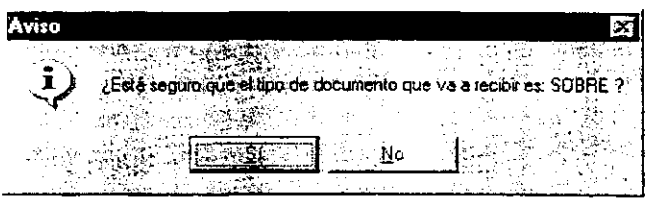
Recepción de Pedidos por Mensajeria

Esta opción del menú se usará para registrar la información de entrega a mensajerias por tipo de documentos como son: sobre, acuse, información de robo/extravio

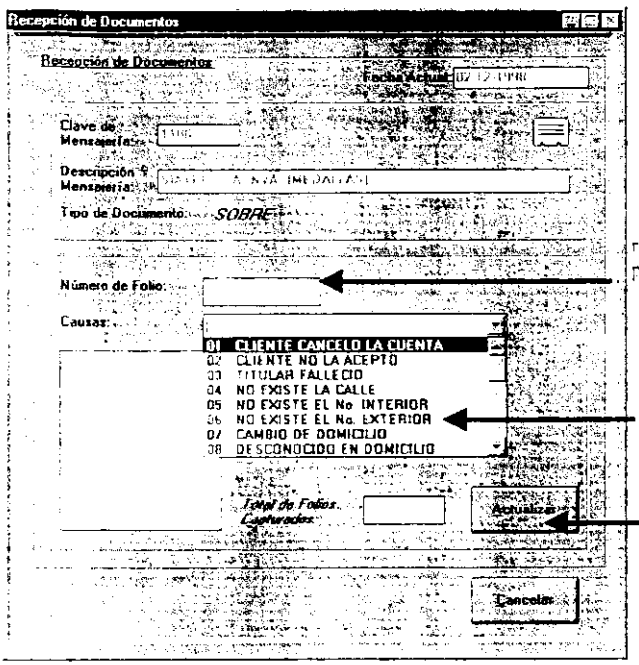
Recepción de Robo/Extravio por mensajeria



Si la selección de tipo de documento es sobre aparecerá el siguiente mensaje de confirmación:



Si la respuesta es no, regresará a la ventana principal de recepción de pedidos por mensajeria. Si la respuesta es si, se presentará:



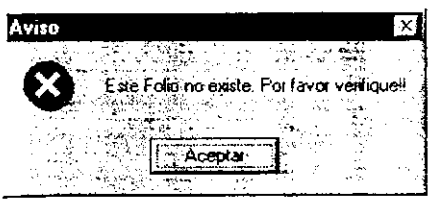
Para registrar las respuestas, se captura primero el numero de folio del envío

Seleccionar la causa de regreso de la mensajería.

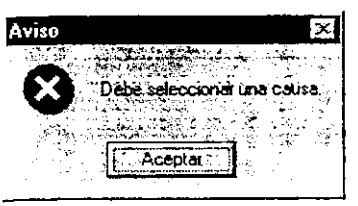
Presionar el botón actualizar.

A lo cual la aplicación hará las siguientes validaciones:

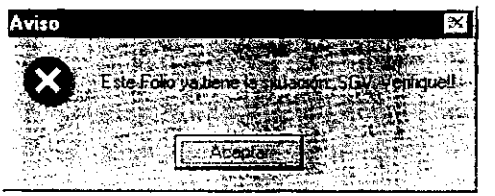
Si el número de folio es incorrecto



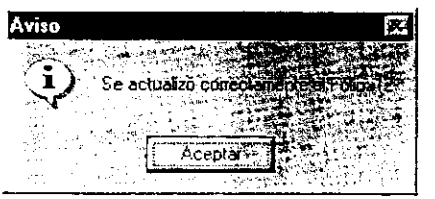
Si no se seleccionó una causa



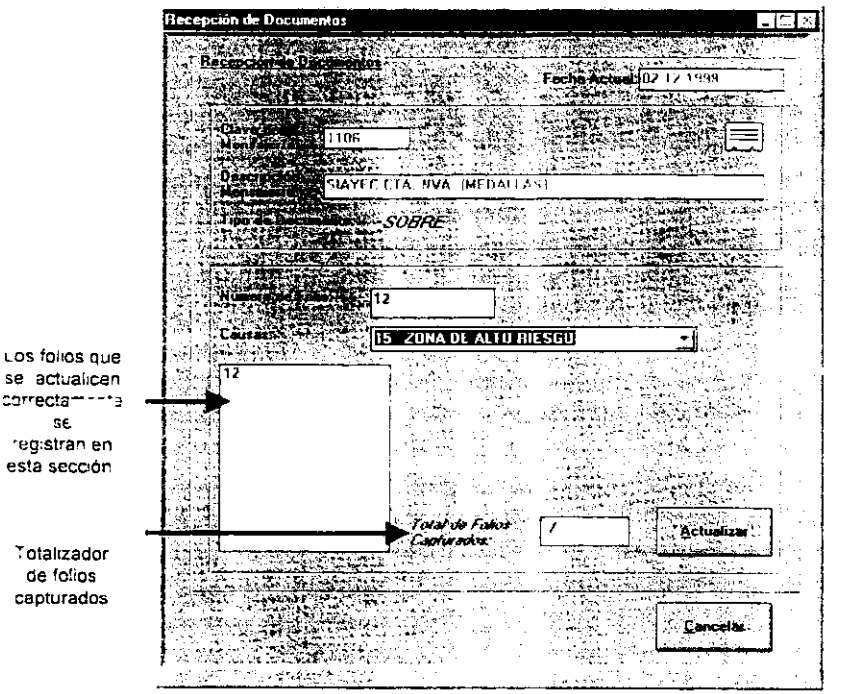
Si el folio ya estaba registrado dentro del sistema



Si el folio se actualizó correctamente



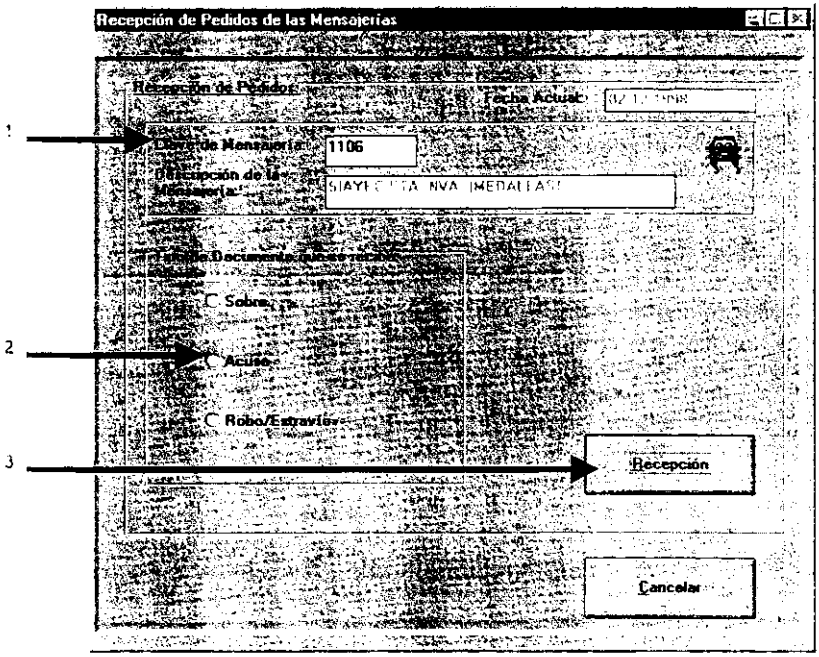
Recepción de Sobres por Mensajería



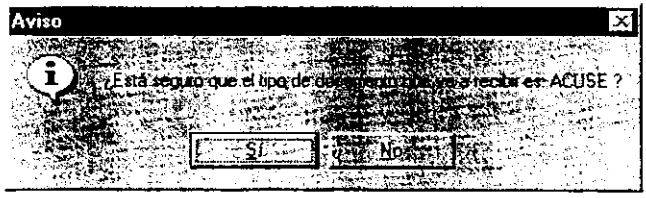
Recepción de Acuses por Mensajería

Desde la ventana de Recepción de pedidos por mensajería se realizan los siguientes pasos

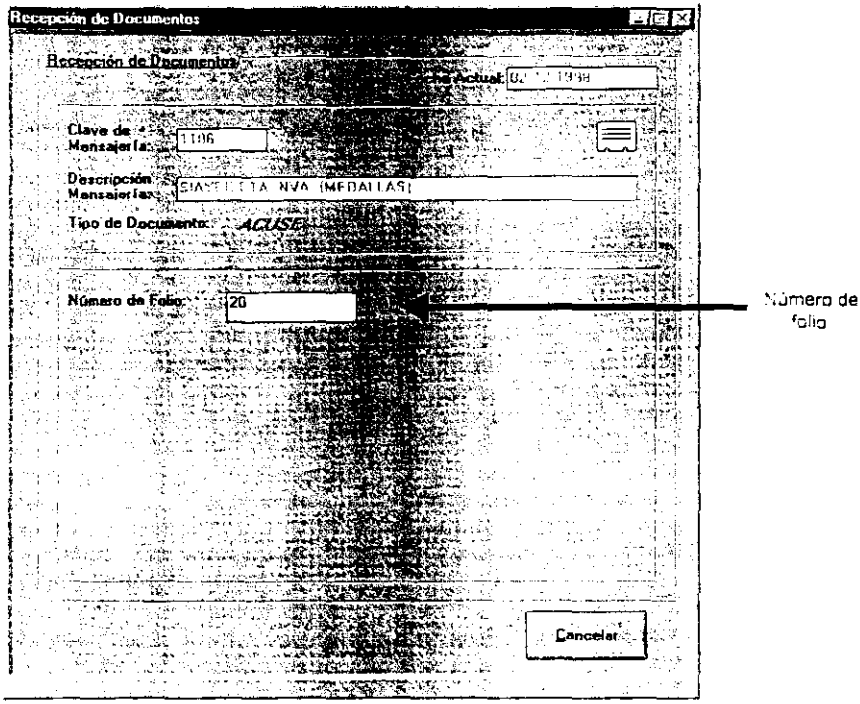
1. Captura la clave de la mensajería y el tipo de documento proporcionado por la misma si la clave es correcta la descripción aparecerá de forma automática.
2. Selecciona la opción Acuse
3. Presionar el botón Aceptar



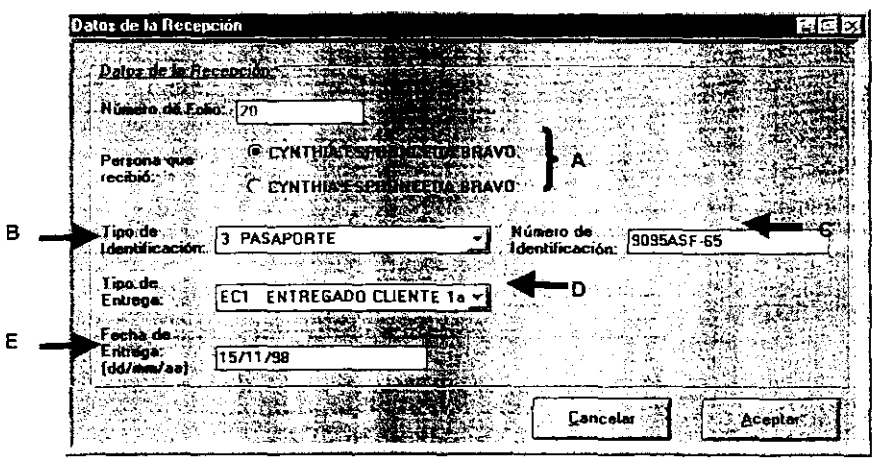
Seleccionado el tipo de documento se presentará el siguiente mensaje de confirmación:



La aplicación presentará la ventana de Recepción de Documentos y se deberá capturar el número de folio:



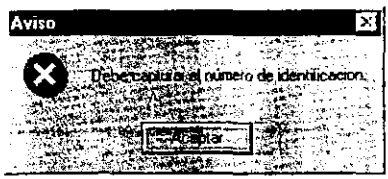
Si el número de folio es válido aparecerá la ventana de captura para datos de recepción del acuse, en donde se tendrán que realizar las siguientes acciones.



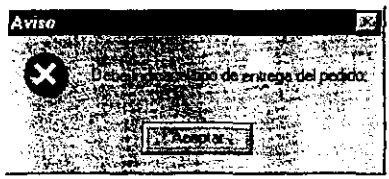
a Seleccionar la persona que recibió la mercancía (misma que el tarjetahabiente registro en el levantamiento del pedido.
b Elegir el tipo de identificación presentado por quien recibió la mercancía
c Registrar el número de identificación.
d Elegir el tipo de entrega.
e Registrar la fecha de entrega (dd/mm/aa)

El sistema deberá realizar las siguientes validaciones en caso de que no sea registrada la información.

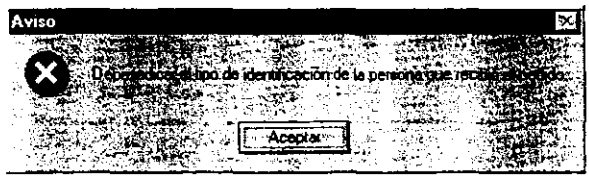
Captura número de identificación



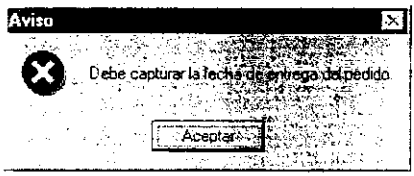
Tipo de entrega de pedido



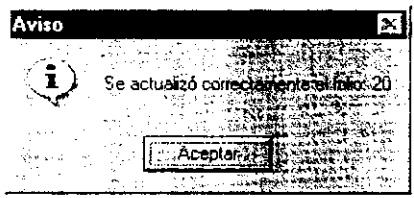
Tipo de identificación



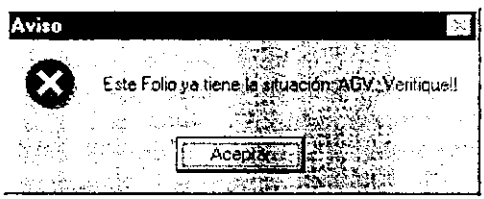
Fecha de entrega del pedido



Actualización correcta del folio

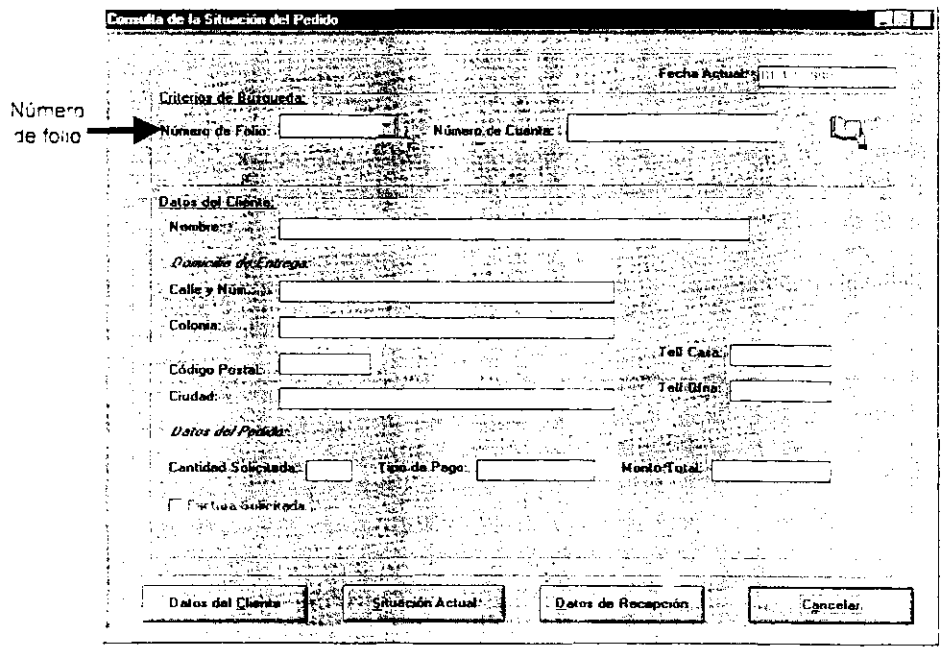


Número de folio registrado

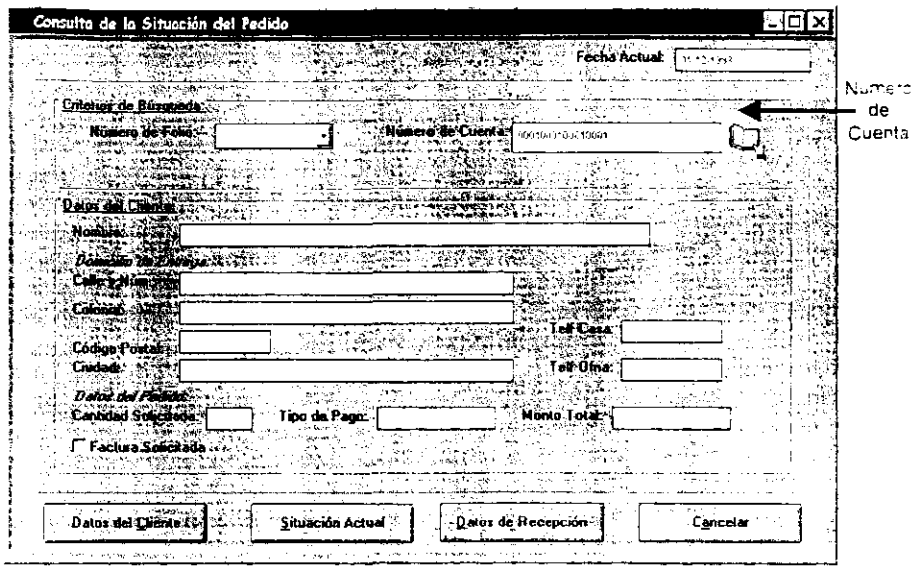


Consulta de Situación del Pedido

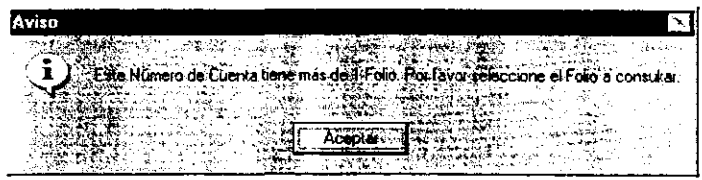
Para poder consultar la situación en la que se encuentra el pedido se podrá hacer por dos tipos por número de folio, al capturarlo en el campo correspondiente



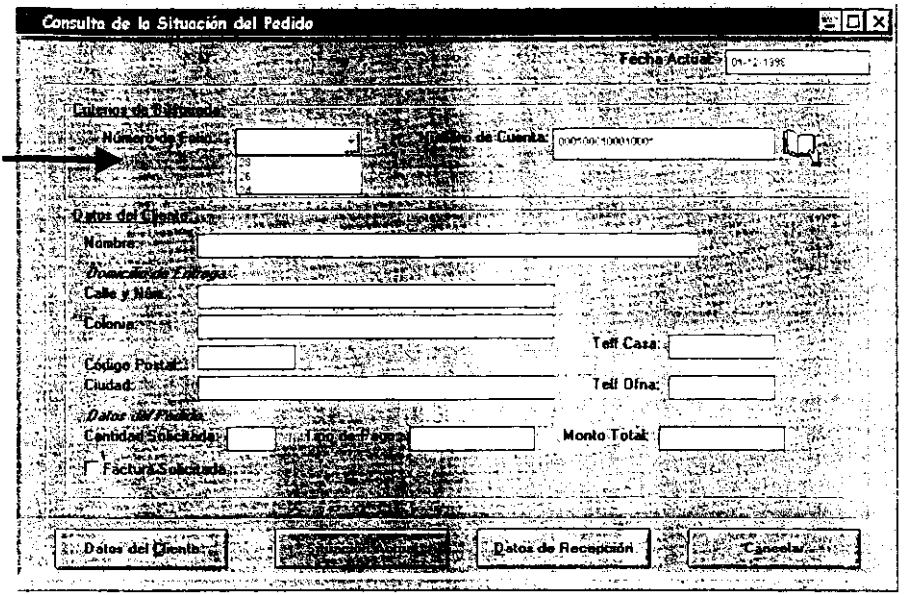
Por número de cuenta:



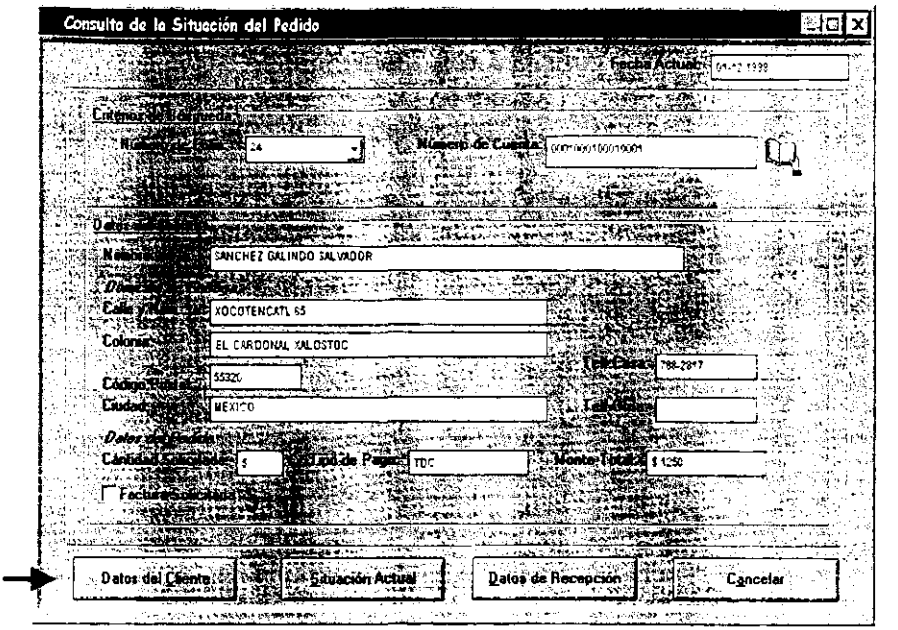
Si el sistema identifica que ese número de cuenta tiene más de un folio asociado, se presentará el siguiente mensaje:



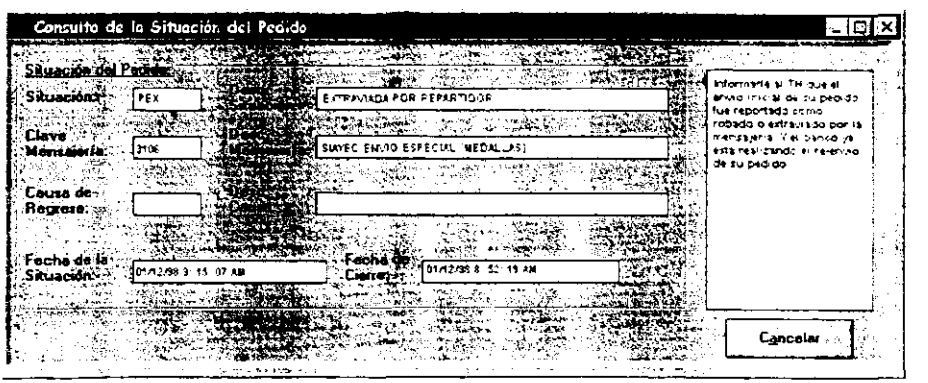
Y al oprimir aceptar se desplegará la siguiente pantalla en donde aparecerán los folios asociados para que el asesor seleccione sólo uno



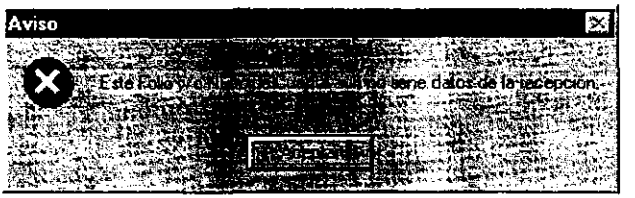
En ese momento deberá oprimirse el botón datos del cliente para que en la pantalla aparezca el grupo de campos del cliente con los datos capturados al momento de levantar el pedido.



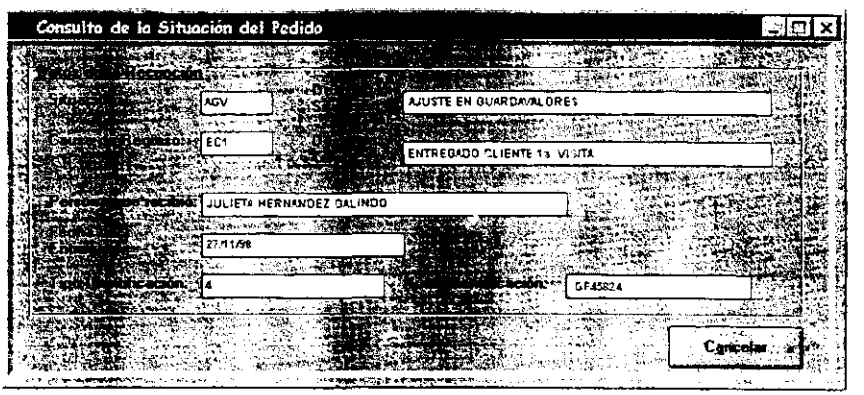
Al presionar el botón Situación Actual se detalla el envío a mensajería: situación, mensajería, causa de regreso, fecha de situación, fecha de cierre:



Para obtener la información sobre la recepción de los articulos se oprime el botón de datos de recepción, pero si el número de cuenta y/o folio no tiene una situación de Acuse y se presiona este botón aparecerá el siguiente mensaje.



En caso contrario, si el folio y/o cuenta si tiene una situación de Acuse (pedido ya entregado al cliente) se muestra la siguiente pantalla:



III Diseño de la Base de Datos para la distribución de pedidos

De acuerdo a la funcionalidad con que se debe contar en la aplicación que se desarrollo, se empleo SQL Server para implementar la base de datos, la cual cuenta con las siguientes relaciones:

1. Relación Cliente

Esta relación cuenta con los atributos necesarios para registrar todos los datos del cliente y su estructura es como a continuación se muestra:

Cliente	
No_Cuenta	Char 16
Folio	Float 8
Nombre_th	Varchar 80
Nombre1_Entrega	Varchar 80
Nombre2_Entrega	Varchar 80
Fecha_Vencimiento	Char 8
Miembro_Desde	Char 8
Calle_No	Varchar 50
Colonia	Varchar 50
c_p	Char 5
Ciudad	Char 35
Entre_Calles	Char 100
Telefono_Casa	Char 10
Telefono_Ofna	Char 10
Factura	Char 5
Llamada	Char 10

2. Relación Pedido

Esta relación cuenta con campos que permitan registrar los datos referentes al pedido como son:

Pedido	
No_Cuenta	Char 16
Folio	Float 8
Fecha	Char 10
Hora	Char 8
Tipo_Pago	Char 5
Autorizacion	Char 6
User_Name	Char 70
Monto	Char 9
Cantidad	Char 9
Entrega	Varchar 25

3. Relación Productos

Esta relación es un catálogo de artículos promocionales que pueden ser vendidos por el área. Con ella será fácil emplear esta aplicación para los nuevos productos que se presenten periódicamente, a lo cual sólo será necesario dar de alta el producto para poder empezar a comercializarlo con los tarjetahabientes. Su estructura es:

Productos	
Clave	Char 10
Descripcion	Char 30
Unidades	Char 9
Costo_Tc	Char 9
Costo_Pnts	Char 9

4. Relación Tipo_Identificacion

Esta relación de igual forma es un catálogo a través del cual el asesor puede seleccionar fácilmente el tipo de identificación que se presento al ser entregado el artículo y cuenta con los siguientes atributos:

Tipo_Identificacion	
Clave_Id	Int 4
Desc_Id	Varchar 25

5. Relación Usuario_Pedidos

Esta relación se conforma de atributos necesarios para poder registrar los datos del asesor telefónico que atiende la solicitud del pedido.

Usuario_Pedidos	
Usuario	Varchar 8
Nombre_Usuario	Varchar 45
Password	Varchar 8
Perfil	Varchar 10

6. Relación Factura

Es a través de la relación factura por medio de la cual se pueden registrar los datos referentes a las facturas solicitadas por los tarjetahabientes y cuenta con los siguientes atributos.

Factura	
No_Cuenta	Char 16
Folio	Float 8
Nombre_Facturar	Char 80
Calle_No	Char 50
Colonia	Char 50
c_p	Char 9
Ciudad	Char 35
RFC	Char 15

7. Relación Distribucion_Pedidos

Esta relación contendrá la información necesaria para poder consultar el status en que se encuentra el envío del pedido y poder ofrecer este servicio al tarjetahabiente, su estructura es la siguiente:

Distribucion_Pedidos	
No_Cuenta	Char 16
Folio	Float 8
Cp_Clamens_Men	Varchar 9
Situacion	Varchar 9
F_Situacion	Datetime 8
F_Cierre	Datetime 8
F_Regreso	Datetime 8
Causa_Regreso	Varchar 3
Quien_Recibio	Varchar 45
F_Entrega	Datetime 8
Tipo_Id	Int 4
Num_Id	Varchar 20
Reenvio	Char 2
Usuario	Varchar 10

8. Relación Actual_Stock

Sobre esta relación se vaciará el contenido de artículos existentes en la bóveda, para lo cuál cuenta con los siguientes atributos:

Actual_Stock	
Usuario	Varchar 8
Stock_Anterior	Varchar 9
Cantidad	Varchar 9
F_Actualización	Datetime 8
Tipo_Movto	Varchar 10

9. Relación Cp_Mensajeria

Aquí se tiene un catálogo de las mensajerías con que actualmente trabaja la institución financiera para distribuir las tarjetas de crédito y que serán las mismas para distribuir los artículos promocionales. Esta relación se conforma de los siguientes atributos:

Cp_Mensajeria	
Cp_Clamens_Mens	Varchar 8
Cp_Descrip_Men	Varchar 9

10 Relación Cp_Situaciones

Esta relación es un catálogo a través del cual el asesor puede saber la situación en la que se encuentra el pedido del tarjetahabiente y podrá proporcionarle esta información, para ello la relación cuenta con los siguientes atributos:

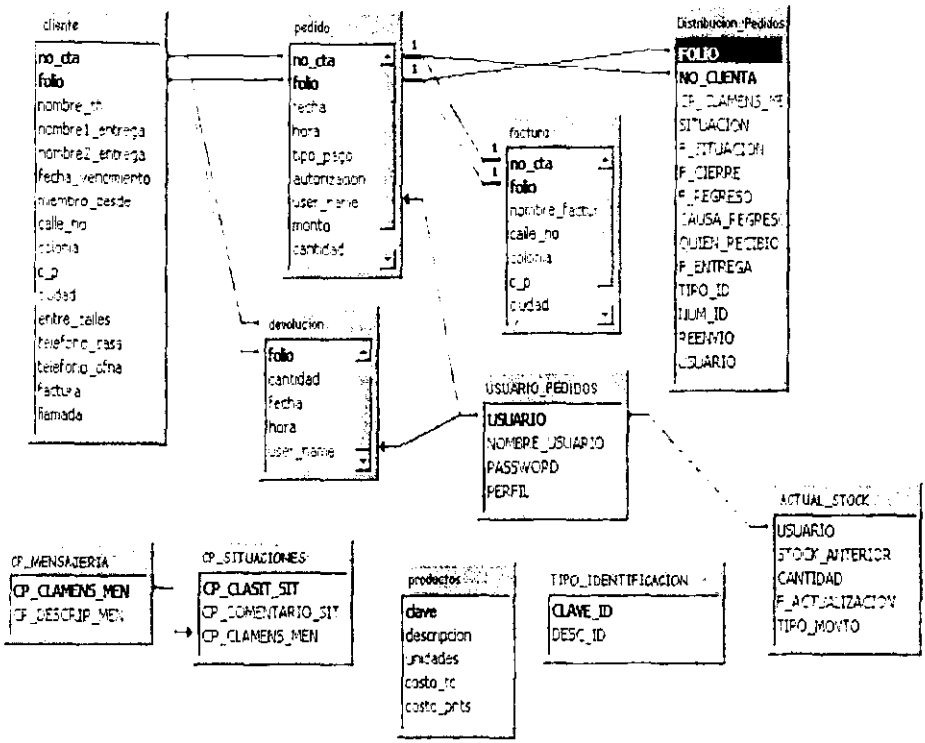
Cp_Situaciones	
Cp_Clasit_Sit	Varchar 8
Cp_Comentario_Sit	Varchar 9
Cp_Clamens_Men	Varchar 9

11. Relación Devolucion

Esta relación permite tener un control sobre las devoluciones realizadas por el cliente y a su vez permite mantener actualizado el stock de los artículos existentes en la bóveda. Sus atributos son:

Devolucion	
No_Cta	Char 16
Folio	Char 5
Cantidad	Char 9
Fecha	Char 10
Hora	Char 8
User_Name	Char 8

De acuerdo a esta estructura las relaciones quedarían como a continuación se muestra:



IV Diseño de la infraestructura

Los componentes necesarios tanto para el área de distribución como para el servicio de telemarketing en el edificio corporativo en donde se encuentra integrado el call center de esta institución son los siguientes:

1 Integrar a la red local del edificio un servidor aplicativo y de base de datos con las siguientes características: Compaq Proliant 3000, 32 RAM, NT4, SQL Server 6.5

2. Establecer una célula especial nombrada "Célula de Telemarketing", la cual tendrá un horario de atención de lunes a domingo de 8:00 a 20:00hrs. y contará con el siguiente equipo de trabajo:
   - PC: HP Pentium a 500Mhz, 32 RAM, 1G en disco duro, Office 97 y Windows 95.
   - ACD: Nortel Modelo M2216 con display
   - Diadema: Nortel Modelo Plantronics
   - Terminales TEF
   - Impresora: HP Láser Jet 4

3. El equipo de cómputo de la célula de telemarketing deberá integrarse a la red del dominio del edificio corporativo por medio del protocolo TCP/IP.

4. El equipo de telefonía deberá integrarse a un grupo generado dentro del conmutador del edificio para poder distribuir las cargas de trabajo entre los agentes que se encuentren firmados en la célula de telemarketing en ese momento.

5. El área de distribución contará con PC's pentium a 500Mhz, 32 RAM, 1G de DD, Office 97 y Windows 95 integradas al dominio del edifico por medio de TCP/IP.

CONCLUSIONES

Del trabajo realizado se puede concluir que las bases de datos relacionales cuentan con gran difusión en el ámbito informático y se pueden encontrar presentes en todos los sectores laborales existentes gracias a la contribución que tienen para establecer procesos de administración y control confiables dentro de las instituciones, además de ver que SQL es un lenguaje declarativo fácil de entender y de aprender, lo cual explica en parte, la aceptación que ha tenido como lenguaje de SGBDR.

El hecho de contar con una base de datos que permita registrar los pedidos realizados de cualquier tipo de producto así como el status que guarda, le ha permitido a la institución financiera cubrir dos de sus objetivos principales:

1° Ofrecer un servicio de telemarketing de gran calidad, lo que aumenta el nivel de servicio ofrecido al cliente y se traduce en mayor captación del mercado.

2° Reducir en forma considerable los gastos en que se incurre cada vez que se lanza un nuevo producto a la venta debido a la contratación de empresas externas.

Una institución como ésta deberá integrar constantemente de manera rápida y sin conflictos variedades de información, actividad que podrá cubrir fácilmente entre menos dispares se encuentren las bases de datos que emplea para su operación cotidiana.

BIBLIOGRAFIA

1. HANSEN, Gary W. y HANSEN JamesV. (1997) Diseño y Gestión de bases de datos. Madrid: Ed. Prentice Hall Segunda Edición.
2. PIATTINI, Mario G. (1998). Fundamentos y modelos de bases de datos. México: Ed. Alfaomega.
3. ATRE, Shakuntala. (1988). Técnicas de bases de datos: estructuración en diseño y administración. México: Ed. Trillas.
4 GONZALEZ, Alfons. (1999). SQL Server: programación y administración. México: Ed. Alfaomega.
5. MICROSOFT. (1998). Implementing a database design on Microsoft SQL Server 6.5. United States of America: Microsoft Corporation.
6. DELOBEL, Claude. (1987). Bases de datos y sistemas relacionales. Barcelona: Ed. Ediciones Omega.
7. KRUGLINSKI, David. (1985). Sistemas de administración de bases de datos. México: Ed. MacGraw-Hill
8 GARDARIN, Georges. (1994). Dominar las bases de datos: modelos y lenguajes Barcelona: Ed. Eyrolles.
9. CAMPDERRICH, Benet. (1984). Técnicas de Bases de datos. Barcelona: Ed. Editores técnicos y asociados S.A.
10 DEEN, S.M. (1987). Fundamentos de los sistemas de bases de datos. Barcelona: Ed. Gustavo Gill S.A.
11. DATE, C.J. (1996). Introducción a los sistemas de bases de datos. México: Ed. Addison Wesley Iberoamericana.