7
2EJ

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

ANALISIS, DISEÑO Y CONSTRUCCION DE UN SISTEMA CLIENTE/SERVIDOR PARA EL AREA DE VENTAS DE UNA EMPRESA INDUSTRIAL

SEMINARIO DE INVESTIGACION INFORMATICA

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN INFORMATICA

P R E S E N T A

MANUEL ALEJANDRO CARDONA SANDOVAL

ASESOR DEL SEMINARIO:
M. A. LUIS EDUARDO LOPEZ CASTRO

MEXICO, D. F 1995

FALLA DE ORIGEN

TESIS CON
FALLA DE ORIGEN

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Dedicatoria

A Mis Padres

Gracias a ustedes que me dieron la mejor de las herencia, la cultura del estudio. Que es la mayor riqueza que un ser humano pueda tener.

A Mis Hermanos

Por su apoyo y comprensión gracias y ser mis amigos en mi vida.

A Madrina Maria del Refugio

Gracias madrina por tus consejos y apoyo para poder terminar mis estudios y obtener mi titulo.

A Mi Padrino Carlos Cardona

Gracias padrino por tus consejos y animos para lograr mis objetivos

A Mis Compañeros y amigos de carrera

Por su apoyo y amistad les agradezco su ayuda y compañerismo.
Gracias a esta empresa por pulir mis conocimientos y experiencias en mi trabajo.

A la Gerencia de Informática de la División Quimicos de Servicos Industriales Peñoles y Compañeros de Trabajo

Gracias a las personas que me dieron la facilidad para completar este proyecto y me ayudaron a madurar profesionalmente.

Gracias a Dios que me dio Vida y Sabiduria para concluir mi Carrera Profesional

Indice

Indice

Indice

Indice

Indice

Indice

INTRODUCCION

La evolución de la humanidad se ha dado como consecuencia de nuevos paradigmas en las actividades humanas, ya que podemos ver como paises del llamado bloque socialista trasforman su forma de gobierno a economías capitalistas, o como los paises capitalistas están adoptando modalidades de alianzas económicas con otras naciones, lo que trae consigo que las empresas que integran estos bloque cambien sus políticas financieras, económicas, de recursos humanos, etc., así como sus tecnologías aplicables a todos sus proceso, sean productivos o administrativos, todo esto para poder ser más competitivos, y dar productos y servicios de calidad en un ámbito tan cambiante y competitivo.

La tecnología como muchas otras actividades humanas, también ha sufrido cambios, ya que podemos conceptualizar a esta en tres fases: la tecnología manual en la cual los métodos de hacer las cosas eran utilizando formas manuales, tecnología mecánica en donde por medios mecánicos realizaban las tareas y tecnología automatizada que utiliza medios como computadoras, programas, chips, etc.,. y la cual estamos viviendo en la actualidad. Ya que podemos ver su aplicación, en la producción con robots controlados por programas, en la astronomía con telescopios y equipos computarizados o en la medicina con equipos totalmente computarizados, en fin podríamos listas un numero considerable de actividades en donde la tecnología computarizada se aplica, pero en este trabajo el tipo de tecnología que nos interesa estudiar es la tecnología informática, en los procesos de una empresa y/o entidad, en donde ha habido una gran revolución de las computadoras y de los paradigmas software, para poder hacer los procesos más eficientes.

A través del desarrollo de la computación, han existido diversas generaciones de computadoras, en donde el salto de una generación a otra radica en la velocidad, los componentes, tamaño, capacidades, etc., de los equipos de computo, en las comunicación, capacidad y manejo de los programas de software. Pero sin duda el objetivo de todas las generaciones ha sido hacer más eficientes los procesos y el generar información oportuna para poder tomar mejores decisiones, y hacer a las empresas más productivas, para poder competir y sobrevivir en el ámbito económico. Por lo cual la informática como herramienta para proveer información y eficientar procesos, ha tenido en los últimos años un gran auge.

El desarrollo de sistemas, el cual es uno de los principales productos de la Informática, también ha tenido su evolución, tanto a nivel de arquitectura, como a nivel de desarrollo, esta evolución ha sido influida por los cambios en el hardware, software, en la administración de las empresas y en sus necesidades, ya que en décadas pasadas las empresas basaban sus sistemas, en arquitecturas de tipo centralizadas, bajo mainframes y lenguajes de segunda o tercera generación, en la actualidad aun cuando se siguen utilizando esa filosofía, surge nuevas concepciones para el desarrollo de software, las cuales nacen como producto del desarrollo de equipos más poderosos, comunicaciones, lenguajes más inteligentes y amigables, etc., esto conjuntado con las nuevas necesidades de las empresas, como la descentralización de funciones, la diversificación de actividades, las expansiones de los mercados, etc.
Todo esto ha traído que los sistemas también cambien, tal es el caso de las aquitecturas distribuidas en datos y procesos, como son los sistemas cliente/servidor, que surgen como una tecnología nueva para las necesidades actuales del procesamiento de la información en las empresas.

Es por lo cual esta tesis presenta una estrategia para el desarrollo de un sistema cliente/servidor, la cual define una arquitectura y un método de análisis y diseño, aplicados a una empresa de la vida real, como es el Sistema de Ventas de la División Químicos, de la compañía Servicios Industriales Peñoles, S.A.
La estrategia que se presenta es factor de mejora continua, debido a las características de los sistemas cliente/servidor, nos proporciona una serie de ventajas y beneficios sobre las arquitecturas de los sistemas existentes, tanto a nivel desarrollo en la Gerencia de Informática, como a nivel de requerimientos en las áreas usuarias.

La tesis se compone de seis capítulos:

Primer Capitulo Sistemas de Información.

En este apartado se presenta un marco de referencia sobre los sistemas de información, así como los diferentes modelos para representar los datos(bases de datos de administración de archivos, jerárquicas, de red y relaciona) y las arquitecturas de procesos(centralizada y distribuida).

En una segunda sección se muestra la definición de la arquitectura cliente/servidor, como una forma de procesamiento de información distribuida, formado por tres elementos: el cliente, la red y el servidor, los cuales tienen funciones bien definidas como: la interface de control y despliegue(cliente), el almacenamiento de datos y ejecución procesos pesados (servidor) y la comunicación entre estos dos por medio de la red. Así mismo se enuncian las características de los sistemas cliente/servidor en comparación con los sistemas centralizados en términos de:

- El ambiente, en el cual se desarrolla el sistema
- La infraestructura
- Las operaciones de soporte

Por último se describen las ventajas y desventajas de contar con el desarrollo y producción de un sistemas cliente/servidor.

Segundo Capitulo. Entorno de Aplicación.

En este capitulo se describe la empresa en la cual fue aplicado este método, así como la descripción del área donde se implanto el sistema y los beneficios del sistema cliente/servidor, en términos del área usuaria y de la Gerencia de Informática. En este capitulo se describe el origen de esta tesis, ya que en esta empresa, se requería del desarrollo de un nuevo sistema, el cual sustituyera a un sistema de tipo centralizado, el cual tenia dos problemas básicos: no cumplia con todas las necesidades del área de ventas y su mantenimiento era muy costoso y llevaba mucho tiempo, para la Gerencia de Informática. Esto propicio definir la nueva plataforma, para el sistema. Esta nueva plataforma fue la arquitectura cliente/servidor, pero esto trajo un nuevo problema, ya que debido a que es un nuevo concepto de desarrollo, ahora se debía de buscar un método de análisis y diseño del sistema, por lo cual esta tesis presenta la forma para desarrollar un sistema cliente/servidor.

Tercer Capitulo. Estrategia.

En este capitulo se presenta la estrategia del nuevo sistema, en cuanto a la arquitectura y el método de análisis y diseño para el desarrollo del sistema. Esta estrategia se basa en el diseño de un servidor basado en el modelo relacional, el cual estará construido bajo tablas con campos y relaciones entre estas, así mismo la base de datos, debe de cumplir ciertos requisitos, como es el caso de seguridad e integridad de la información, con estos elementos podemos contar con un servidor de datos de tipo relacional.

Mientras que para el cliente, este se basa su codificación en un lenguaje orientado a objetos, en donde la interface de despliegue gráfico es construida basada en clases y objetos, y el control de los procesos del sistema, basado en eventos que afectan a objetos, los cuales realizan algún procesos o método.

Una vez construidos el servidor y el cliente, el control del sistema, se realizará bajo la interface del cliente, en la cual por medio de mensajes realizará ciertos eventos, los cuales ejecutaran proceso, ya sea en el cliente o en el servidor, en el caso que sea al servidor, este lanzará clausuras SQL o mandará ejecutar procedimientos almacenados en el servidor, todo esto mediante la comunicación vía red, una vez terminada la ejecución en el servidor, se devuelve el resultado al cliente, el cual lo manipula y lo despliega.

Cuarto Capitulo. Análisis de Sistema.

En este capitulo se describe la forma de hacer un análisis para un sistema de este tipo, el cual se basa en el desarrollo de tres modelos; entidad/objetos, dinámico y funcional. Los cuales servirán de base para el diseño del sistema. El modelo entidad/objeto, describe a los elementos que intervienen en el sistema, como son las entidades u objetos, una vez identificados se construye un diagrama entidad/relación, en el cual se describen todas las relaciones entre entidades y las características o atributos de las mismas. El modelo dinámico describir el ciclo de estados de esas entidades, como producto de afectaciones de eventos por medio de diagramas de estados. Por último el modelo funcional nos presenta los procesos y flujo de datos que se realizan en el contexto del sistema, a través de un diagrama de flujo de datos. Todos estos modelos se integran en uno solo, el modelo de análisis, el cual por medio de un diagrama de descomposición de desglosan los procesos principales del sistema y a partir de cada nodo del diagrama, se descomponen el diagrama entidad/relación, el diagrama de estados y el diagrama de flujo de datos

Quinto Capitulo Diseño de Sistema.

En este capitulo se describe como a partir de los modelos desarrollados en análisis, se procede a diseñar un servidor de tipo relacional y un cliente orientado a objetos.
Del modelo entidad/objeto, se procede a diseñar y generar al mismo tiempo la base de datos en el DBMS, definiendo para esta las estructuras de almacenamiento de datos, como son las tablas a partir de las entidades de análisis, los mecanismos para conservar la integridad de datos como son tipos de datos, reglas, restricciones de datos y defaults, así mismo la definición de mecanismos para conservar la integridad referencial como son disparadores y restricciones referenciales. Todos estos mecanismos de definen a partir de las reglas del negocio.
Posteriormente al diseño del servidor, se procede a diseñar un prototipo GUI, así como liberias y funciones de objetos y estándares de programación. Todo esto se tomará de base para la programación de todas las aplicaciones del Cliente, en la fase de construcción.

Sexto Capitulo Construcción de las Aplicaciones

Una vez que contamos con la generación de la base de datos y del prototipo, como se menciono anteriormente se procede a programar las aplicaciones GUI del sistema, a partir de prototipo de diseño. Las aplicaciones deberán soportar los requerimientos del cliente final, las cuales exploten los datos en el servidor.
Por lo que este capitulo define la forma de codificar las aplicaciones, a partir de los modelo dinámico y funcional de cada entidad y proceso del sistema. En donde un proceso automatizado, tendrá una aplicación GUI, la cual explotará los datos por medio de store procedure o un estatuto SQL vía red al servidor, por lo que es necesario programar estos elementos .

Septimo Capitulo. Sistema de Ventas.

En este último capitulo se muestra como fue codificado el Sistema de Ventas de Servicios Industriales Peñoles, en términos del servidor y el cliente, mostrando cuantas tablas tienen el back-end y cuantas interfaces o pantallas el cliente. Así mismo se presenta a fondo una de las funciones del sistema automatizado, la función de los Certificados de Análisis, analizando los componentes del Cliente y del Servidor, y la interacción entre los dos.

La filosofía cliente/servidor, como tal, se basa principalmente en el funcionamiento de elementos autónomos e independientes que trabajan coordinadamente entre si, y realizan una distribución del trabajo. Estos elementos son hardware y software, los cuales tienen una actividad definida en el procesamiento de la información. Los elementos que conforman esta tecnología, son el cliente y el servidor.

El cliente es un sistema el cual es operado desde una PC, el cual bajo un ambiente gráfico y amigable, proporciona una interface gráfica de control y despliegue de los datos. Mientras que por parte del servidor, este es un sistema ejecutado desde una estación de trabajo o una minicomputadora, el cual se encarga de almacenar los datos del sistema en cuestión, proporcionando seguridad e integridad de esa información. Tanto el cliente como el servidor se comunican a través de una red, en donde el cliente requiere una petición al servidor y este le envía la respuesta, por medio de la red.

Esta nueva arquitectura de procesamiento de información, cambia el paradigma de los sistemas, ya que las aplicaciones anteriores, se basaban en sistemas centralizados en procesos y datos. Ahora el paradigma cambia, ya que estas aplicaciones son migradas a plataformas distribuidas en datos y procesos y en sistemas de hardware más pequeños, pero más poderosos, además de aprovechar el compartir recursos como información, hardware y software por medio de las redes.
Los sistemas Cliente/Servidor, así mismo son sistemas más amigables, debido a su interface gráfica y sus ayudas en líneas, así mismo, una de sus grandes ventajas es la reducción del mantenimiento de los sistemas, y además que son muy fáciles para adaptar nuevos componentes.
En la Facultad de Contaduría y Administración, en la carrera de Informática, se han desarrollado diversas tesis a cerca de Cliente/servidor, pero en realidad el enfoque de esta investigación es diferente a las demás investigaciones de las cuales he tenido noticia.
Tal es el caso de la tesis con titulo ***"Cliente/servidor, un autentico cambio en tecnología informática y su aplicación en un cajero bancario"***, cuyos autores son Blanca Graciela Lopez Jimenez, Marco Antonio Camacho Moreno y Rene Saul de la Rosa Castilla, del año de 1994, esta tesis básicamente en el Capitulo 1 habla a cerca de los ambientes distribuidos en forma teorica, mientras que en el capitulo 2 y 3 se refiere a la teoria de la Arquitectura cliente/servidor, así mismo en el capitulo 4 habla en forma muy escueta y teorica de la implementación de un sistema cliente/servidor y por último el capitulo 5 habla de la aplicación de un sistema cliente/servidor en un ***cajero bancario***, en el cual presenta la forma de trabajar del cliente y del servidor y como es la comunicación entre estos. La diferencia con esta tesis, radica en que se habla a cerca de un método provado y definido para hacer análsis, diseño y construcción de un sistema cliente/servidor, aplicado a una empresa industrial (quimica) y aun cuando en los primeros capitulos se da una serie de antecedentes teoricos de la arquitectura Cliente/Servidor, en los demás se habla de un método de desarrollo cliente/servidor, otra diferencia radica en la aplicación de la tesis, ya que mientras en esta tesis se habla de un sistema bancario, aqui la aplicación es un sistema de ventas de una empresa industrial en la rama de quimicos.

Otra de las tesis que se han escrito a cerca de cliente/servidor, es la que lleva el titulo de ***"Arquitectura Cliente/Servidor un enfoque evolutivo empresarial"***, desarrollado por Carolina Flores Zamora en el año de 1994, esta tesis la cual cuanta con tres capitulo, habla solo elementos teoricos de la arquitectura cliente/servidor, ya que su primer capitulo se refiere a antecedentes de los sistemas de información, mientras que en capitulo 2 y 3 solo se refiere a conceptos como conceptos, caracteristicas, ventajas, desventajas, modelos integración, etc. de sistemas cliente/servidor, pero no da un ejemplo real de esta tecnologia, mientras que esta investigación, se basa en la aplicación real de un método para desarrollar un sistema cliente/servidor.

Otro seminario de investigación a cerca de este tema, es el que tiene el titulo de ***"Aplicación del Concepto Cliente/servidor en un sistema de control escolar en la UNAM"***, cuyos autores son Julio César Roldán Campos y Rutth Elizabeth Enríquez Mountaunt del año de 1995, esta tesis como la anterior en sus primeros capitulos hablan sobre conceptos teoricos de este tema, ya que en su primer capitulo se habla de los antecedentes de los sistemas manejadores de base de datos, mientras que en el segundo capitulo, se refiere a conceptos de la arquitecura cliente/servidor. Esta tesis quiza sea la que tenga mayor parecido con mi investigación, ya que los capitulos restantes hablan del lugar de aplicación, el análsis, y diseño y programación del sistema y mi tesis también en sus capitulo habla de lo mismo, pero la diferencia radica en que la tesis del sistema de control escolar hace solo referencia desde el análsis hasta la programación solo al servidor y no al cliente para el caso de ese sistema, y tampoco explica como es la comunicación entre estos dos y solo se habla en forma especifica para el sistema de controle escolar, mientra que mi trabajo explica como visualizar desde análsis el servidor y al cliente, hasta su construcción, para el sistema de ventas y en forma general, y como es la comunicación entre el cliente y el servidor , ademas que la tesis del sistema de control escolar se aplica en la UNAM y esta se aplica a un sistema de ventas de una empresa del sector privado.

La última tesis de la cual conozco que fue desarrollada sobre cliente/servidor, es la de ***" La administración pública y la Arquitectura Cliente/Servior un caso practico"***, desarrollada por Saul Rodriguez Saud, del año de 1994, la cual solo presenta un caso de estudio de la evaluación para la compra de un equipo de computo, con sistema operativo abierto y la instalación de un sistema manejador de base de datos relacional en una Secretaria de Estado del Gobiernmo Federal, el cual contaba con sistemas centralizados en sus aplicaciones y ahora cuenta con sistemas abiertos, por lo cual es más que nada un caso de downsizing de las aplicaciones centralizadas a aplicaciones abiertas. Por lo cual esta tesis difiere mucho con la investigación en cuestión, debido a la naturaeleza del contenido y de la entidad en la cual es la aplicación.

Por lo anterior podemos justificar que no existe una tesis, la cual nos hable de como hacer un análsis, diseño y construcción de un sistema cliente/servidor, ya que otras investigaciones solo presentan elementos teoricos de funcionamiento, pero no presentan una alternativa real para desarrollar un sistema de este tipo, aun cuando algunas presetan algunos casos, solo son ejemplo, pero no indica una guía para el desarrollo.

CAPITULO 1
SISTEMAS DE INFORMACION

El mundo, en el cual vivimos, esta formado por sistemas, como el sistema monetario de nuestro país, el sistema eléctrico de la ciudad, el sistema digestivo de nuestro cuerpo, entre otros más. Pero cual es el significado real de un sistema, y de su funcionamiento, mucha gente no lo sabe. Pero en realidad los sistemas nos rodean y están presentes en todas las actividades de la humanidad, por lo cual hablaremos un poco de ellos.

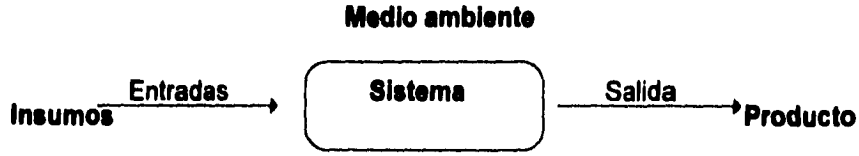
TEORIA DE SISTEMAS

Concepto De Sistemas

Para definir un sistema, existen muchos conceptos, pero en lo particular la siguiente definición es muy clara y sencilla; *"conjunto u ordenación de cosas relacionadas de tal manera que forman una unidad o un todo orgánico"*[1] un ejemplo es el sistema motriz de un automóvil, formado por válvula, pistones, cables, etc. los cuales están integrados y su objetivo es el generar fuerza para hacer girar los ejes del automóvil y así moverse.

Funcionamiento De Un Sistema

El objetivo primordial de un sistema es el captar estímulos del medio ambiente, para procesarlos y obtener algún producto.



Proceso de un sistema

Un sistema forma parte de un medio ambiente, el cual se le llama **macrosistema**, y a su vez el sistema esta compuesto por elementos a los cuales se les llama **subsistemas** . Esto nos hace pensar en jerarquías, pongamos un ejemplo; una ***empresa*** (zapatos el alce) es un sistema organizacional, esta formada por ***direcciones*** (ventas, informática, contraloría, etc.), las direcciones están formadas por *gerencias* (ventas nacional, ventas exportación), las ***gerencias*** están formadas por *departamentos o áreas* (investigación de mercados, publicidad, promoción, ventas, etc.), las ***áreas*** están formadas por *puestos* (facturista, vendedor, investigador, etc.). Ahora hacia arriba, una *empresa* forma parte de una ***industria*** (industria del calzado), la *industria* forma parte un ***sector económico*** (sector secundario), el *sector* forma parte de ***sistema económico de un país*** (México).

[1] Ingenieria de Software, Roger S. Pressman p.37

FALLA DE ORIGEN

CAPITULO 1
SISTEMAS DE INFORMACION

El mundo, en el cual vivimos, esta formado por sistemas, como el sistema monetario de nuestro país, el sistema eléctrico de la ciudad, el sistema digestivo de nuestro cuerpo, entre otros más. Pero cual es el significado real de un sistema, y de su funcionamiento, mucha gente no lo sabe. Pero en realidad los sistemas nos rodean y están presentes en todas las actividades de la humanidad, por lo cual hablaremos un poco de ellos.

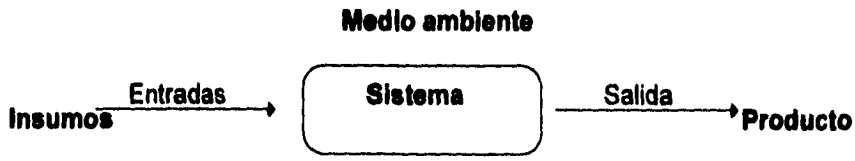
TEORIA DE SISTEMAS

Concepto De Sistemas

Para definir un sistema, existen muchos conceptos, pero en lo particular la siguiente definición es muy clara y sencilla; *"conjunto u ordenación de cosas relacionadas de tal manera que forman una unidad o un todo orgánico"*[1] un ejemplo es el sistema motriz de un automóvil, formado por válvula, pistones, cables, etc. los cuales están integrados y su objetivo es el generar fuerza para hacer girar los ejes del automóvil y así moverse.

Funcionamiento De Un Sistema

El objetivo primordial de un sistema es el captar estímulos del medio ambiente, para procesarlos y obtener algún producto.

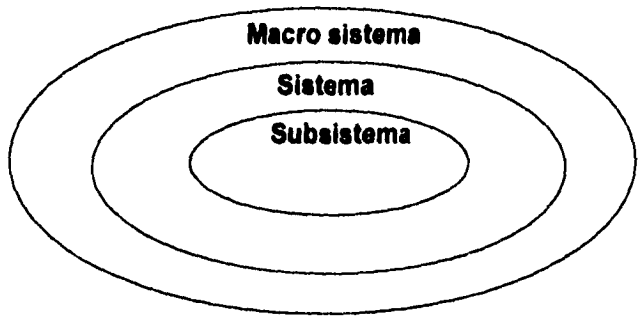


Proceso de un sistema

Un sistema forma parte de un medio ambiente, el cual se le llama **macrosistema,** y a su vez el sistema esta compuesto por elementos a los cuales se les llama **subsistemas** . Esto nos hace pensar en jerarquías, pongamos un ejemplo; una ***empresa*** (zapatos el alce) es un sistema organizacional, esta formada por ***direcciones*** (ventas, informática, contraloría, etc.), las direcciones están formadas por *gerencias* (ventas nacional, ventas exportación), las ***gerencias*** están formadas por *departamentos o áreas* (investigación de mercados, publicidad, promoción, ventas, etc.), las ***áreas*** están formadas por *puestos* (facturista, vendedor, investigador, etc.). Ahora hacia arriba, una *empresa* forma parte de una ***industria*** (industria del calzado), la *industria* forma parte un ***sector económico*** (sector secundario), el *sector* forma parte de ***sistema económico de un país*** (México).

[1] Ingenieria de Software, Roger S. Pressman p.37

FALLA DE ORIGEN



Jerarquía de los sistemas

Tipos De Sistemas

Como en muchas cosas existen diferentes tipos o clasificaciones, los sistemas no son la exepción, ya que hay diferentes tipos de sistemas:

- Sistemas Naturales
- Sistemas Artificiales
- Sistemas Cerrados
- Sistemas Abiertos

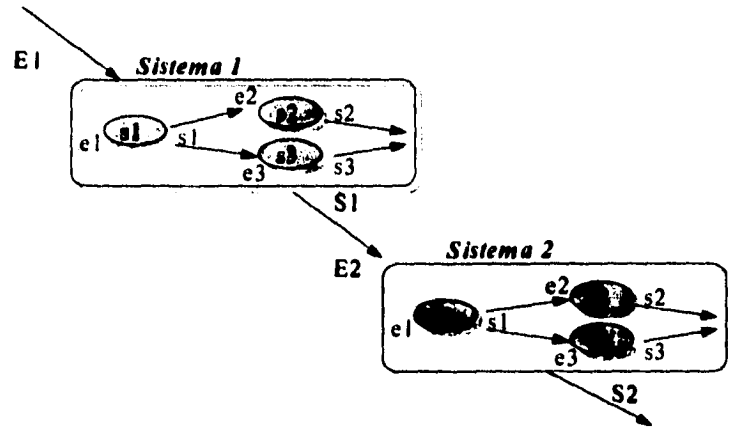
De estos cuatro tipos de sistemas, los de mayor interés para nuestro estudio, son los sistemas cerrados y los abiertos.

Sistemas Cerrados

Los sistemas cerrados son aquellos que no tienen entradas ni salidas, es decir no hay insumos que capte, ni productos que se emitan, y por lo cual no interactuan con su medio, pero este tienen un funcionamiento interno, y esta compuesto de elementos.

Sistemas Abiertos

Son aquellos que tienen una interacción con su medio ambiente, los cuales tienen entradas, las procesan y se obtienen productos. Las entradas son tomadas del medio ambiente, las cuales fueron productos de otro sistema de la misma jerarquía, este ciclo se repite hasta obtener un producto final para un sistema de mayor jerarquía.



Interrelación de los sistemas abiertos

Después de estudiar algunos elementos de la teoría de sistemas, tenemos una idea clara de lo que significa un sistema, y como debemos de visualizarlos, en relación a su entorno. Ya que cada tipo de sistema, sean abiertos, naturales y artificiales(excepto los cerrados), se interrelacionan entre si.

SISTEMAS DE INFORMACION

En la sección anterior, se pudo observar como son los sistemas, y cuales son sus características, ahora hablaremos de un tipo especial de sistemas, los sistemas de información para una organización. La información en nuestra actualidad es de gran importancia, ya que esta debe ser concisa, veraz y oportuna, para poder tomar decisiones para el logro de nuestros objetivos. Este concepto es de vital importancia en las empresas, ya que en la actualidad la situación económica de los mercados, es de alta competitividad, y se requiere que las empresas sean de alta calidad en sus productos y servicios, para poder sobrevivir en este ámbito.

El estudio en la administración de la información, ha llevado al desarrollo de diversas formas y métodos para la recopilación, proceso y distribución de la información. En una organización, se llevan a cabo procesos, los cuales están continuamente captando información, la cual es transformada, y es emitida hacia otro proceso. La evolución de los métodos para la gestión de la información, ha hecho que los procesos sean más eficientes cada vez más y así poder:

- Reducir costos
- Disminuir cargas de trabajo
- Unificar actividades
- Mejorar los canales de comunicación
- Reducir tiempos
- Pensar en actividades que no se han desarrollado aun

Este estudio de la información, no solamente se ha centrado en la forma conceptual de llevar las cosas, sino en la forma practica de hacerlas, es por esto que la introducción de la tecnología en este ámbito es de vital importancia, como es el caso de la tecnología de la computación, como herramienta para la administración de la información, ya que la computadora debido a sus características, como son velocidad, capacidad de almacenaje, operabilidad de funciones aritméticas, etc. ha contribuido al mejor manejo de la información, por lo cual surge la informática, la cual realiza; *los procesos más eficientes y nos proporciona información oportuna, concisa y veraz para la toma decisiones*.

Ahora que entendemos por informática, *es la técnica que por medios manuales, mecánicos y automatizados, recopila datos los procesa y distribuye información veraz, concisa y oportuna para la toma de decisiones* . Por la definición anterior podemos decir que la informática tiene como función primordial, la creación de sistemas de información para las entidad u organizaciones, los cuales tengan como característica principal eficientar procesos y generar información para la toma de decisiones, estos sistemas pueden ser llevados a cabo por medios manuales, mecánicos o computarizados, según sea la factibilidad del proceso.

Concepto De Sistema De Información.

Es el conjunto de datos, herramientas (hojas, maquina mecánica o computadora), métodos, procedimientos, procesos y actividades, los cuales interactuan mutuamente para la obtención de información.

Tipos De Sistemas Automatizados

Una manera de ordenar por categorías los sistemas automatizados es por su aplicación: sistemas de manufactura, sistemas de contabilidad, sistemas de defensa militar, sistemas de control de procesos, etc. Sin embargo, esto no resulta muy útil, ya que para cuestiones de nuestro estudio, es más conveniente y más general la siguiente:

- Sistemas en línea y batch
- Sistemas de tiempo real
- Sistemas de apoyo a decisiones
- Sistemas basados en el conocimiento

Sistemas En Línea Y Batch

El sistema en línea *"Es aquel que acepta material de entrada directamente del área donde se creó. Además que el material de salida, o el resultado de la computación, se devuelve directamente a donde es requerido."*[2] Y existe una interacción directa y continua con el usuario. Así mismo los datos almacenados, como archivos o bases de datos, se organizan de tal manera que los componentes individuales de información puedan ser recuperados o modificados rápidamente, sin tener necesidad que efectuar acceso a otros componentes.
Ejemplos de este tipo de sistemas tenemos: sistemas de reservación en líneas aéreas, sistemas de facturación, sistemas de inventarios, sistemas de registro, sistemas bancarios, sistemas de consulta, entre otros más.

[2] Análsis estructurado Moderno, Edward Yourdon, p25

Los sistemas en batch son aquellos que ejecutan una serie de instrucciones en forma continua, sin necesidad de estar interactuando con el usuario, sino solo para ejecutar ordenes de largo tiempo.
Ejemplos de estos tenemos: sistemas de nomina, sistemas de cierre de ventas mensuales, sistemas de contabilidad, entre otros más

Sistemas De Tiempo Real

"Un sistema de tiempo real, es considerado como una variante de un sistema en línea, pero este controla un ambiente recibiendo datos, procesándolos y devolviéndolos con la suficiente rapidez como para influir en dicho ambiente en ese momento"[3] . Una de las características que hacen llamar a estos sistemas de tiempo real, es la velocidad de respuesta ante un estimulo de entrada, para poder actuar en ese medio.

Sistemas De Apoyo A Decisiones Y Sistemas De Planeación Estratégica

Son aquellos sistemas los cuales ayudan a tomar decisiones inteligentes y documentadas a los administradores de una organización, este tipo de sistemas son pasivos, ya que no operan en forma regular , sino que se utilizan cuando se necesitan. Este tipo de sistemas por lo general son de análisis estadístico, de pronóstico, etc. De hecho una característica en común de los sistemas de apoyo a las decisiones, no sólo recuperan y exhiben datos, sino que también realizan varios tipos de análisis matemáticos y estadísticos de los mismos. Así mismo tienen la capacidad de presentar información en una variedad de reportes, gráficas , etc.

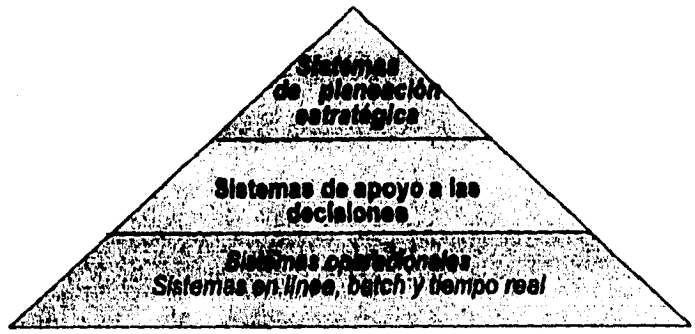
Los *sistemas de planeación estratégica* , son utilizados por lo gerentes en jefe de una organización, para evaluar y analizar la misión de la empresa. En lugar de dar consejos acerca de alguna decisión de negocios aislada, estos sistemas ofrecen consejos más amplios y generales acerca de la naturaleza del mercado, las preferencias de los consumidores o el comportamiento de la competencia.

Sistemas Basados En El Conocimiento

Estos sistemas, mejor conocidos como sistemas expertos en el campo de la inteligencia artificial, el cual se define de la siguiente manera; es un programa de computadoras que contiene el conocimiento y la capacidad necesaria para desempeñar en un nivel de experto en una actividad, estos sistemas por lo general se construyen de tal manera que sean capaces de explicar las líneas de razonamiento que llevaron a las decisiones que tomaron.

Una vez que conocimos los diferentes tipos de sistemas, podemos decir que en una organización, todos estos tipos de sistemas deben de existen y se relacionan entre si, lo cual nos hace pensar en la integración de los sistemas y de las organizaciones, en uno solo.

[3] idem p27



La jerarquía de los sistemas de información[4]

ARQUITECTURA DE LOS SISTEMAS DE INFORMACION

Los sistemas de información computacionales, desde su surgimiento se han organizado de diferentes maneras, para el procesamiento de la información. Pero en todas estas formas han existido dos elementos de principal importancia:

- Base de Datos
- Arquitectura de Procesos

Bases de Datos

Las Bases de datos, es el conjunto de datos que guardan una relación entre si y están almacenados con una determinada estructura. Para el manejo de estos datos existe un software llamado DBMS (Data Base Management System) el cual se encarga de la administración de la información almacenada en la base de datos. Un DBMS funciona basándose bajo un cierto modelo de manejo de la información.

Arquitectura De Procesos

Es la forma de como están organizados los elementos de un sistema de computo y de como se procesa la información.

[4] idem p37

Modelos De Manejo De Información

A través del desarrollo de la computación se han desarrollado diferentes formas de almacenar la información, en los sistemas computarizados. Cada método tiene característica diferentes, las cuales hacen que unos tengan mayores ventajas sobre otros. Estos modelo son:

- Modelo de Administración de Archivos
- Modelo de Jerárquico
- Modelo de Red
- Modelo Relacional

Modelo Manejador De Archivos

El *modelo manejador de archivos* (FMS File Management System) es muy fácil de entender, ya que almacena los datos en forma secuencial en el disco. En el modelo FMS, las relaciones se realizan a través de apuntadores los cuales hacen referencia a los elementos de la relación, y así poder localizarlo.

La ventaja de este modelo es la simplicidad, mientras que su principal desventaja es que es difícil implementar un numero grande de relaciones, además de que la secuencialidad lo hace lento en el almacenamiento y acceso de la información.

Modelo Jerárquico

El *modelo jerárquico* (HDS Hierarchical Database system) organiza sus datos a través de una estructura de árbol, en donde existe una raíz y cada dato se almacena en un nivel. A cada nivel de la estructura se le conoce como nodo.

Las relaciones que existen es de padres a hijos, es decir un nodo tendrá una o muchas ocurrencias, la gran ventaja de este modelo es la facilidad y velocidad de localización de un elemento de un nodo padre a un nodo hijo, mientras que su gran desventaja es la dificultad para definir una estructura de este tipo, además de que es muy difícil realizar cambios y dar mantenimiento a este tipo de bases de datos.

Modelo de red

El *Modelo de red* (NDS Network Database System) no tiene que ver nada con su forma física de almacenamiento, ya que su concepción de red es a nivel lógico, ya que describe relaciones muchos a muchos, las cuales están limitadamente definidas.

La ventaja de este modelo es la flexibilidad para el manejo de relaciones muchos a muchos, lo que hace de este modelo muy rápido para localizar información, pero estas relaciones son difíciles de definir y una vez realizadas es difícil cambiarlas y darles mantenimiento al igual que en el modelo jerárquico.

Modelo relacional

El *modelo relacional* (RM Relational Model) fue descrito por primera vez en 1969, en una publicación por el Dr. E: F. Codd, un científico de bases de datos de IBM. Este modelo es el único que se basa en una teoría, la relacional y la de conjuntos. La teoría relacional nos marca 12 reglas las cuales un sistema debe de cumplir para ser de tipo relacional.

El modelo relacional se basa en el almacenamiento de la información en tablas, formadas por renglones y columnas. En donde los renglones son registros y las columnas son campos.

La principal ventaja de este modelo es la facilidad para definir las relaciones entre tabla y tabla, ya que estas se definen a través de columnas iguales en cada tabla, por ejemplo la relación entre la tabla LIBRO y AUTOR es la columna autor_id en las dos tablas, las cuales son iguales.

Arquitectura De Procesos

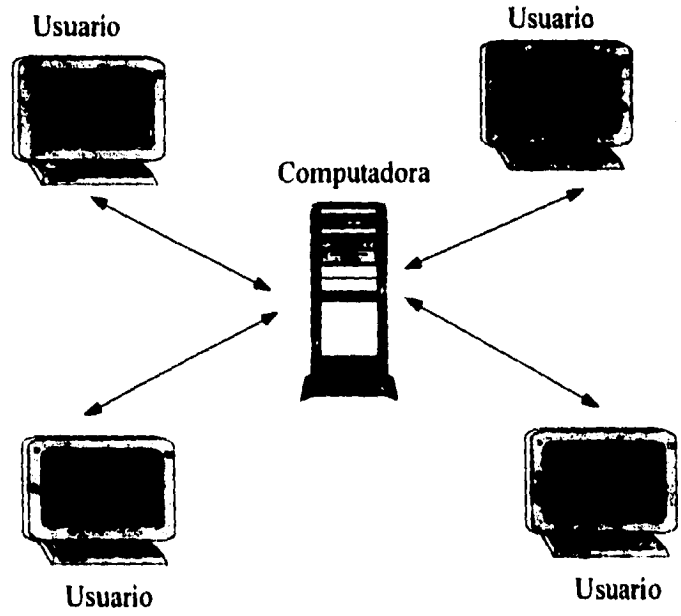
Una vez que hablamos de los diferentes modelos de almacenamiento de información, ahora nos toca explicar las diferentes arquitectura de los procesos. Entre los cuales se encuentran:

- Arquitectura centralizados
- Arquitectura distribuidos
- Arquitectura cliente-servidor

Arquitectura centralizados

En el inicio del procesamiento de información, los sistemas eran desarrollados para trabajar bajo un mainframe, este un equipo grande y voluminoso, que se calentaba rápidamente, estos eran equipos de tipo monolíticos, ya que se constituían de un procesador central y dispositivos de entrada/salida, como terminales, impresoras, lectores, etc. El procesamiento de información y las bases de datos era centralizado.

Sistema Centralizado



Centralizado

Este tipo de sistemas surgieron con el nacimiento de diferentes compañías de computadoras, las cuales cada una de ellas en forma independiente, desarrollaron sus propias tecnologías, tales como sistemas operativos, lenguajes, hardware, protocolos de comunicación, etc. en la lucha por ganar clientes para sus marcas. Los sistemas que soportaban esta arquitectura no se podían comunicar con otro de diferente compañía, por lo cual surgió una falta de estandarización y se pueden considerar a estos sistemas como *cerrados*.

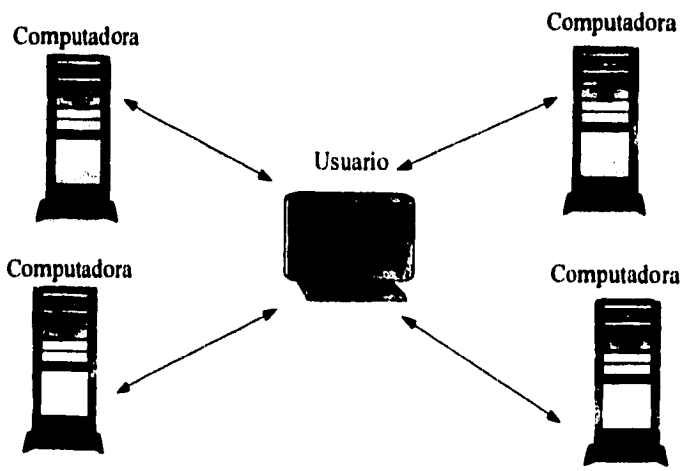
En la evolución de las tecnologías de computo, los sistemas han cambiado, así como la forma de desarrollarlos, por lo que en la actualidad surgen los sistemas de tipo distribuido, lo contrario a los centralizados. Esto debido a el desarrollo de equipos más pequeños que los mainframes, pero más rápidos y eficientes, de redes de comunicaciones, de software, y de bases de datos.

Arquitectura Distribuidos

La arquitectura distribuida esta representada por medio de los Sistemas de Distribuidos, los cual distribuyen físicamente los componentes de un sistema, como son ***procesos y datos.***

Proceso de Datos Distribuidos

Este termino ha ido evolucionando a través del tiempo, ya que en los procesamientos de información centralizados, el procesador (CPU) y los dispositivos de entrada y salida (I/O) son considerados como un proceso distribuido, ya que están separados. Pero en la actualidad este concepto esta más desarrollado, y lo podemos definir como; *una serie de procesadores autónomos (no necesariamente homogéneos) los cuales están interconectados a través de un red de computación y los cuales trabajan en forma coordinada y cooperativa en la ejecución de un trabajo, en donde cada procesador ejecuta una tarea.* Estos procesadores distribuidos, tienen cada uno el software y hardware necesarios para controlarse independientemente, y no existe un control centralizado.



Distribuido

Que es un sistema de bases de datos distribuidos

Podemos definir a un sistema de base de datos distribuida *como una colección múltiple de datos, los cuales están almacenados en diferentes bases de datos, distribuidas en diferentes sitios geográficos, las cuales se unen a través de una red de computadoras.* Ahora un elemento importante en un sistema de base de datos distribuida, es el software que coordina a esas bases de datos, este se le llama DBMS Data Base Managament System (Administrador de sistemas de bases de datos) y es definió como; *el software que permite la administración de un DDBS y hace que la distribución de los datos sea transparente para el usuario.*

Una de las características de los términos anteriores, que cabe resaltar en *la Interrelación lógica* y *la distribución a través de una red.* Primero, un DDBS no es una colección de archivos, sino un conjunto de tablas que están almacenados en cada nodo de una red. Para un DDBS las tablas están relacionadas lógicamente, unas con otras, aun cuando estén en diferentes lugares, los datos pueden ser accesados en cualquier lugar por medio de una interface.

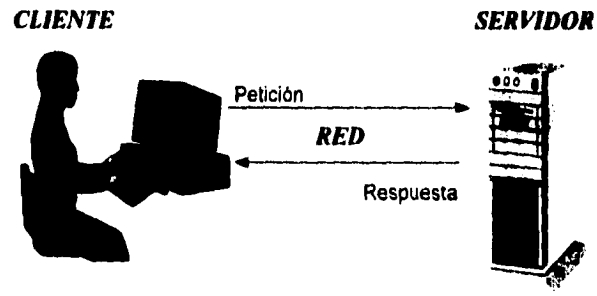
Así mismo en esta arquitectura no solo los datos se encuentran distribuidos, sino los procesos también, en los cuales podemos repartir las tareas y realizar procesos en paralelo de diferentes lugares geográficos.

Arquitectura cliente-servidor

Entre una de las arquitecturas distribuidas que existen, tenemos a la arquitectura Cliente-Servidor la cual se refiere a dos aplicación de software, las cuales se ejecutan en diferentes computadoras, comunicadas por medio de una red. Cada computadora trabaja en forma coordinada, la una con la otra a través de una red.

Los softwares son llamados, cliente y servidor, las cuales cooperan juntos en un trabajo conjunto. El ***cliente*** maneja la interface interactiva con el humano, a través de un ambiente gráfico, en el cual se introducen datos, se validan y se ejecutan pequeñas computaciones, este tipo de software se almacena en *Computadoras Personales,* mientras que el ***servidor*** es el encargado de hacer los procesos intensos y del almacenamiento, seguridad, integridad y entrada/salida de la información en una base de datos, generalmente este tipo de software se almacena en una computadora con procesador RISC o en una minicomputadora. Las dos maquinas se encuentran unidas en una ***red***, la cual proporciona la interface de comunicación, para poder intercambiar **mensajes**.

Cliente/Servidor



La arquitectura Cliente-Servidor, es un paradigma de procesamiento de información distribuido, ya que los procesos se llevan a cabo en lugares diferentes, lo cual ha hecho que se incremente su popularidad, además de que este tipo de aplicaciones son las llamadas de *downsize* , en donde las compañías que antes procesaban sus aplicaciones en maquinas grandes, como macrocomputadoras, que eran grandes y voluminosas, a equipos más pequeños en volumen, pero con mayor capacidad de proceso, almacenamiento y rendimiento.

Los proveedores actuales, tanto de software y hardware han desarrollado sus productos, en base a la comunicación con terceros, esto ha provocado que podamos interconectar sistemas de computo heterogéneos, y así poder conformar lo que conocemos ***como sistemas abiertos*** . Por lo cual debemos decir que una arquitectura cliente/servidor es un sistema abierto.

ARQUITECTURA CLIENTE/SERVIDOR

El término Cliente-Servidor , es un concepto nuevo, el cual se ha desarrollado gracias a los cambios en la industria de la computación, particularmente en las computadoras personales, las cuales son cada vez más poderosas y más económicas, así como del desarrollo de las comunicaciones. Además de las características propias de esta arquitectura tales como:

- Bajo costo en términos reales de desarrollo, hardware y operación
- Acceso fácil a la información
- Interface amigable
- Control total de los procesos
- Velocidad de procesamiento
- Integración de ambientes y sistemas

Los elementos que componen esta arquitectura de procesamiento de información son :

- ***Cliente***
- ***Servidor***
- ***Red***

Además de estos tres elementos existe uno más, el cual por lo general esta inmerso dentro de los anteriores, pero debe hacerse referencia a este, el cual es el ***middelware***.

Cliente

El cliente es software y hardware, en donde el software instalado en una estación de trabajo inteligente, es operada por el usuario. Esta computadora tiene su propio sistema operativo, así como la capacidad de ejecutar otros programas, como procesador de palabras u hoja de calculo, así como la aplicación cliente. El cliente trabaja en cooperación con el servidor. Por ejemplo, en una orden de entrada, en una aplicación, el cliente presenta una pantalla de captura al usuario, el cliente acepta la entrada de datos, este envía un mensaje al servidor, el servidor procesa el mensaje y este envía una respuesta al cliente, y este la despliega en la pantalla. El cliente provee una interface entre el usuario y el servidor.

"El cliente realiza servicios de presentación, tales como edición de campos, despliega gráficas, reportes, consultas y ayudas en línea, etc. y hace llamados al servidor, a través de un formato llamado RPC (Remote Procedure Call Procedimiento Remoto de Llamadas)"[5].

La interface gráfica se le conoce como GUI (Graphical User Interface, Interface Gráfica al Usuario) orientada a objetos, la cual consiste en objetos gráficos tales como:

- Ventanas (Windows)
- Menús
- Iconos
- Apuntadores (Pointers)

[5] Testing Client/server Applications, Patricia A. Goglia, p3

- Barras de Control (Scroll Bars)
- Controles
  - *Botones de radio (Radio Buttons)*
  - *Botones de comando*
  - *Listas*
  - *Cajas de chequeo (Check Boxes)*
  - *Sliders*
- Ventanas de Dialogo

Entre las plataformas más conocidas, basadas en GUIs son:

- Apple Macintosh
- Microsoft Windows,
- X-Windows,
- OS/2, Windows 95
- Unix Motif, entre otros.

Las herramientas para desarrollo de GUIs, comúnmente conocidas como 4GL, lenguajes de cuarta generación o semejantes, nos proporcionan una plataforma de desarrollo para software de cliente.
Entre las características para considerar este tipo de productos están:

- Programación orientada a eventos, programación gráfica y/o programación orientada a objetos.
- Soporte el uso de SQL (Structured Query Lenguage)
- Soporte las interfaces gráficas GUI
- Portabilidad, es decir que puedan generar código portable en algún lenguaje como C++, para ser ejecutado en otros equipos
- Que cuente con herramientas gráficas para la programación de: pantallas, reportes, gráficas, etiquetas, etc.
- Trabaje bajo las plataformas de Windows, Windows para grupos de trabajo, Windows NT,
  X-Windows, etc.
- Soporte las interfaces OLE2 y MDI

Entre algunos productos tenemos:

Gain Momentun	Sybase	Windows, Os/2 y Motif Unix
Microsoft Visual Basic	Microsoft	Windows
SQL Windows	Gupta Technologies Inc	Windows y OS/2
PowerBuilder Enterprise	Power Corp	Windows y OS/2
Object Vision	Borland International Inc	Windows y OS/2
Object/1	DataBase System, Inc	Windows

Servidor

El servidor es software y hardware, en donde el software almacenado en una computadora en Red de Area Local (LAN), que provee de servicios a otras computadoras, que están en la red. El servidor no realiza una interface directa con el usuario, sino que este recibe peticiones por parte del cliente, tal como un procesamiento de información, y regresa cierta información al cliente. En una aplicación cliente-servidor, existen diferentes servidores, en donde cada uno realiza un servicio especifico.

"Los siguientes son diferentes tipos de servidores."[6]

- **Servidores de Datos (Data Server)**. Soporta el almacenamiento y extracción de información de archivos y bases de datos.
- **Servidores de Proceso (Process Server)**: Servidores de procesamiento de información
- **Servidores de Impresión (Printer Server)**. Soporta funciones de impresión, para las aplicaciones.
- **Servidores de Fax (Fax Server)**. Soporta las funciones de envio y recepción de documentos
- **Servidores de Comunicaciones (Communications Server)**. Soporta las comunicaciones de tipo WAN (Wide Area Network)

[5] idem p9

- **Servidores de Bases de Datos (Database Server).** Soporta las peticiones de datos que requiere el cliente.

De los servidores mencionados anteriormente, los que en esta investigación nos interesan son los de datos y procesos, los cuales son soportados por software de tipo relacional, y operados por SQL (Structured Query Language). Este software se conoce como RDBMS (Relational Database Management System), el cual se encarga del control de los procesos, a través de procedimientos y triggers, los cuales corren en el servidor.

Entre las plataformas que se pueden utilizar para el manejo de servidores, tenemos:

- Estaciones de trabajo RISC, Unix
- Equipos Dell
- Equipos AS400 de IBM
- Mainframes

Entre algunos manejadores de bases de datos, que funcionan como servidores de datos, procesos e impresión tenemos :

System 10 SQL Server	Sybase, Inc	Unix
Oracle	Oracle, Inc	Unix
Informix	Informix, Inc	Unix
SQLBase	Gupta, Inc	DOS, Windows95
SQLServer	Microsoft	Windows95

Procedimientos

Son procedimientos de código procedural, que están residente en el servidor. Los cuales utilizan comandos de SQL, para el acceso a la base de datos. Estos procedimientos proveen a las aplicaciones cliente/servidor de un mayor redimiento e integridad de los datos debido a:

- El código esta precompilado, de tal forma que la ejecución de la petición del cliente es rápida, debido a que la instrucción SQL esta empotrada.
- Están residentes en el servidor, lo cual reduce el trafico en la red. Ya que el cliente no envía cada instrucción SQL sobre la red. Sino que envía solo llamadas a esos procedimientos.
- Proveen de integridad de los datos debido a la centralización del código

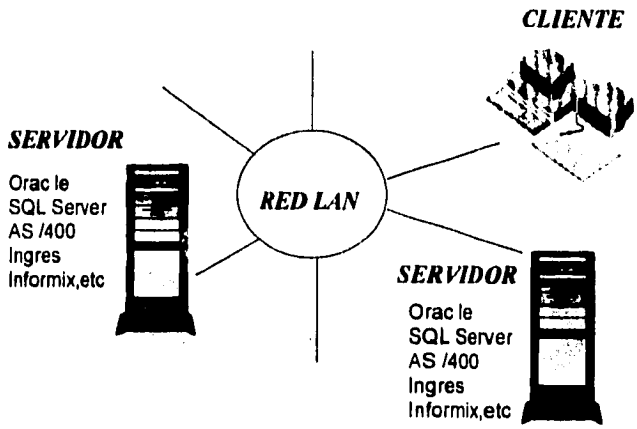
Triggers

Estos son un tipo especifico de procedimientos de almacenamiento . Estos se almacenan en el servidor y se ejecutan automáticamente cuando ocurre un evento predefinido. Este evento puede ser la inserción, actualización o borrado del registro en una tabla.

LA RED

La red provee la conectividad del sistema. El cliente y el servidor están conectados uno al otro a través de una red LAN o WAN. El cliente puede comunicarse a cualquier servidor en la red y el servidor puede dar servicio a cualquier cliente en la red.

Arqutectura Cliente/Servidor



La comunicación en la red, se realiza por medio de un protocolo de comunicaciones, ya que por medio de estos es como se envían los mensajes, entre el cliente y el servidor. Entre los protocolos más conocidos para las comunicaciones tenemos:

- NetBios
- Named Pipes
- DCE-RPC
- APPC de IBM
- TCP/IP

Las Interface gráfica del cliente GUI, es un ambiente fácil de operación y comprensión, ya que por medio de
objetos gráficos podemos operar cierta aplicación. Para el desarrollo y funcionamiento de una GUI, es necesario contar con APIs (Application Programa Interface - Programa de Interface de Aplicaciones) , las cuales son funciones que presentan servicios de presentación al desarrollador de GUIs, estas funciones manejan la creación de ventanas, sonidos, campos de edición, menús, etc. Estas APIs son desarrolladas bajo lenguajes nativos de SDK de Microsoft, lenguaje ensamblador o C.

Middlware

El término "*middleware*" es muy importante de entender al hablar de arquitectura cliente/servidor. Debido a que las aplicaciones cliente/servidor incorporan productos de software y hardware que no provienen del mismo distribuidor, se necesitan herramientas que habiliten a las aplicaciones a trabajar independientemente de los datos, lugares y estructuras de los sistemas.

El término *middleware* puede ser un tanto confuso porque puede ser un software, una especificación de software o una combinación de software y hardware. Esto se debe a que no "*todo el middleware es creado igual, y muchos expertos distinguen tres categorías: middleware de red, de bases de datos, y middleware de administración de copias/servidor de respuestas, en los cuales se ofrecen diferentes niveles de funcionalidad*"[7].

Por lo que el middleware se considera un conjunto de servicios, que hacen posible que las aplicaciones GUI o APIs, puedan comunicarse en una red con un servidor.

Entre los servicios con los cuales podemos contar están:

- Servicios de aplicaciones de la red, tales como: servicio de mensajes, servicio de acceso a bases de datos, servicio de directorios y servicios de seguridad.
- Las aplicaciones desarrolladas con middlware son portables, hacia otras plataformas
- Integración con otras aplicaciones

[7] Karen Watterson, Data onDemand, Window sourde, febrero 1994, pp212

Lógica de la Arquitectura Cliente/Servidor

Un sistema cliente/servidor, como ya vimos se compone de tres elementos básicamente, el cliente, la red y el servidor, estos realizan ciertas funciones, los cuales se basan en cierta lógica, como es:

Lógica de Presentación.

Es aquella la cual se encarga del control de los medios físicos y lógicos de entrada y salida de datos, como son un teclado, un mouse, un monitor, el GUI, etc. Por lo general siempre esta la lleva el cliente.

Lógica de Procesamiento

Es el conjunto de computaciones, cálculos, procesos o transacciones que realiza alguna de las partes del sistema. Esta es realizada por el cliente y/o el servidor, dependiendo de la proceso.

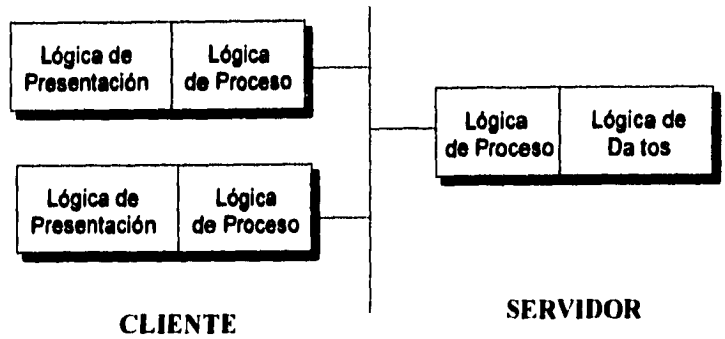
Lógica de Datos

Es la parte que realiza el almacenamiento y administración de los datos en el sistema. Esta es llevada a cabo por el servidor de datos.

Servicios de comunicación.

Son aquellos que se encargan del envío y recepción de paquetes, entre el cliente y el servidor. Estos servicios los lleva a cabo la red por medio de un protocolo.

Arquitectura Cliente/Servidor



Servicios de Red

Categoría de los Sistemas Cliente/Servidor

La arquitectura física de la distribución de los componentes de los sistemas cliente/servidor, la podemos dividir en dos categorías; Arquitectura en dos niveles y en tres niveles.

Arquitectura de dos niveles.

Esta es la forma base de un sistema cliente/servidor, en la cual solo existen dos elementos el cliente y el servidor. El primero se encarga de la interface de despliegue y del control de los procesos, mientras que el segundo, ejecuta las peticiones de almacenamiento y proceso de datos del cliente, por lo tanto las tareas se dividen en dos partes.

Arquitectura de tres niveles.

Esta forma de cliente/servidor, distribuye en más de dos parte física los procesos, ya que el cliente se encarga de la interface con el usuario, el cual solicita servicios de procesamientos, de almacenamiento, de impresión, de comunicación remota, etc. a diferentes servidores, los cuales están hechos para una tarea en especial, lo cual hace que los procesos se distribuyan en varios servidores.

CARACTERÍSTICAS CLIENTE/SERVIDOR

La arquitectura cliente/servidor introduce fundamentales cambios en la forma en que las empresas, administran los sistemas de información. Ya que provee a los usuarios y desarrolladores, un ambiente de autonomía, ya que no son controlados por los procedimientos y estándares de los mainframes. El software de desarrollo tiene el control total sobre el desarrollo, prueba y mantenimiento de las aplicaciones. Después de que el sistema es instalado, los usuarios son dueños de su propio sistema. Ellos pueden accesar y manejar su propia información. Este tipo de libertad requiere una serie de responsabilidades. Un proyecto cliente/servidor nos debe de proveer de una disciplina, para su mantenimiento y soporte de las aplicaciones.

En un ambiente de mainframe, el software es almacenado en una computadora, la cual debe de tener una seguridad física, en el cuarto donde este. En una aplicación cliente/servidor, el software reside en el cliente y en el servidor. Esto es un ambiente distribuido , ya que en cada site se ejecuta una aplicación, con su propia información.

"Las mayores diferencias entre un sistema cliente/servidor y un sistema basado en un mainframe son:"[8]

- ***El ambiente, en el cual se desarrolla el sistema***
- ***La infraestructura***
- ***Las operaciones de soporte***

[8] Testing Client/server Applications, Patricia A. Goglia, p17

El Ambiente , En El Cual Se Desarrolla El Sistema

Los procesos cliente/servidor, utilizan nuevas tecnologías, entre las cuales se incluyen:

- La arquitectura cooperativa de proceso
- Los procesos distribuidos
- Las herramientas para el desarrollo de GUI
- Los productos de bases de datos
- Las herramientas de prueba
- Las herramientas de software para administración
- Los sistemas operativos de red.

Esta nueva tecnología hace que algunos elementos cambien, en relación de un desarrollo tradicional, tales elementos son:

- El personal del área de desarrollo
- Las GUI
- Los servidores de bases de datos
- Los estándares

El personal del área de desarrollo

En un departamento de Informática no existe la experiencia necesaria para el desarrollo de soluciones cliente/servidor , ya que es un concepto nuevo, ya que se requiere de experiencia en bases de datos relaciónales y herramientas de software para la creación de GUIs. Y la formación del programador tradicional, es bajo un ambiente de mainframe, con lenguajes como Cobol, Algol, RPG, etc. o bases de datos jerárquicas y de red. Además de que no tienen la disciplina para el desarrollo de sistemas, bajo una red, ya que estos implican mucho control. Por lo cual este momento, es la oportunidad de integrar gente joven que pueda aprender este tipo de software, ya que seria más difícil que cambien el paradigma de programación de un programador tradicional.

Las GUI

Las GUI programadas en lenguajes de cuarta generación orientados a objetos, genera un cambio en la forma de programar, debido a que estos sistemas de basan en el paradigma de la orientación a objetos.

El servidor de la base de datos

El servidor como elemento independiente, de esta arquitectura, es controlado bajo un software llamado DBMS, este software nos provee de integridad de datos, de concurrencia y de referencias integrales. Existen una gran variedad de productos de este tipo. Los cuales a través de procedimientos y triggers bien definidos, aumentan el rendimiento de una bases de datos.

Para que un servidor sea totalmente funcional, este debe de ser probado bajo todos los eventos que sean controlados por los procedimientos y triggers, para evitar que algún evento no considerado, llegue a afectar a la bases de datos, y esto controlado directamente por el servidor y no dejar nada a los clientes, que controlen, este tipo de eventos.

Los estándares

Por lo general, este paradigma de sistemas, es nuevo, por lo que no existen estándares para el desarrollo de estos sistemas, y si existen, en una organización, habrá los estándares que soporten el antiguo paradigma de la programación.

Por lo que hay que definir nuevos estándares para checar la seguridad, variables, objetos, pantallas, ayudas en línea, etc. esto también se debe de aplicar a la creación de librerías de programas u objetos, con el objetivo de tener código reusable. Todo esto requiere de una formación y cambio de cultura, para sacar el mayor provecho de este paradigma.

En la actualidad existen softwares, los cuales ayudan al control de este tipo de proyectos, ya que ante todo es una forma nueva de integración, en el desarrollo de sistemas, bajo equipos de programación, en donde todos cooperan con todos, este tipo de software se les conoce como ***software team***.

Estos softwares, nos ayudan a definir estándares y librerías de código, esto en organizaciones donde se desarrollan aplicaciones distribuidas en diferentes sitios, en donde cada sitios, puede existir un equipo de programación, los cuales, accesan a un repositorio central, para accesar a librerías de código, o GUIs previamente creadas. Esto debe de ser normado bajo un comite de desarrollo.

La Infraestructura

El tipo de infraestructura que se utiliza en este paradigma, abarca aspectos físicos y de estructura procedimental que soportan el sistema. Esto incluye :

- Soporte de Hardware y Software
- Seguridad, privacidad y prevención de virus
- Un repositorio central
- Herramientas de prueba
- Control de Versiones
- Control del Proyecto
- Monitoreo del sistema

En un ambiente/cliente servidor, estos elementos son totalmente diferentes, en un ambiente de mainframe.

Soporte de hardware y software

En un ambiente de mainframe, los desarrolladores de software, tienen la única responsabilidad del desarrollo de las aplicaciones. Mientras que el hardware es operado y mantenido por otra parte del área de informática, así como el trabajo de instalación y soporte del software, de los compiladores y de la administración de la base de datos, es realizado por otra parte del área.

Seguridad

En un proyecto cliente/servidor, las fases de desarrollo y prueba, deben de estar protegidas. La seguridad física en el cliente y el servidor deben estar implementadas, para las funciones que cada elemento realiza.
Para garantizar que un sistema de este tipo es seguro, debe de haber un ***equipo de prueba*** , el cual realice ejecuciones del sistema a gran escala, además de contar con las medidas de seguridad suficientes para que no accesen al sistema, gente externa, ya sea a través de cliente o del servidor .

En el ámbito de seguridad, también se debe de contar son mecanismos para la prevención, detección y eliminación de un virus, pero será siempre mejor la prevención.

Repositorio central

En un ambiente de mainframe, el área de Informática, provela a un desarrollo de sistemas, los servicios de desarrollo, prueba y producción de un sistema, en forma separada. Mientras que en cliente/servidor, todo esto se centraliza y se controla, en un repositorio de datos, a través de un equipo de desarrollo, que siguen una serie de estándares y utiliza una serie de recursos y librerías de programación estándares, código reusable para la implantación de un sistema.

Control de versiones

Es necesario tener un control de cada nodo o elemento del grupo de programación, en aplicaciones cliente-servidor, así mismo como el control de los módulos o programas alterados o modificados, para llevar el control de las versiones y actualizaciones. Esto se realiza a través de software de grupos de desarrollo, en los cuales se registran los accesos al repositorio de programas, así como la actualización de una bitácora de los eventos que ocurren en los repositorios y de las versiones de cada grupo de programación.

Control del proyecto

El control de un proyecto cliente/servidor es necesariamente requerido, ya que debido a las características de esta arquitectura, es necesario llevar el control de cada fase de un proyecto, para poder controlar; las funciones del cliente, del servidor y de la red.

Monitoreo del sistema

Durante la fase de implantación de una aplicación, un sistema debe de estar monitoreado en el servidor y los clientes, para poder asegurar el rendimiento de este.

Soporte Operacional

El soporte operacional en una arquitectura cliente/servidor es un factor importante. Esto funciona de forma diferente, que en una aplicación mainframe. Ya que una aplicación cliente/servidor, no se ejecuta en un lugar centralizado, como una aplicación mainframe. Por lo tanto este sistema debe de contar con soporte en todos los lugares, que se ejecute una aplicación.

El soporte operacional incluye los siguientes puntos:

- Administración de site físico
- Implantación de planes de contingencia
- Proveer de ayudas sencillas en línea.
- Soporte a la distribución de software
- Minimización del mantenimiento.

Administración física del site

En un ambiente mainframe, las computadoras están en un cuarto, en el cual esta especialmente diseñado para soportar la integridad y seguridad de un equipo. Estos cuartos son conocidos como centros de computo, los cuales tienen los inconvenientes de ser:

- Grandes
- De tener entradas restringidas y solo para personal autorizado
- Temperatura y humedad controlada
- Piso falso, para el cableado
- Alarmas contra fuego y robo
- Y su existe un desperfecto en el cuarto, se necesita llamar aun especialista

Todo lo anterior genera muchos costos para una organización. Mientras que en sistemas cliente/servidor, estos problemas disminuyen, ya que las aplicaciones no están centralizadas, sino distribuidas y los equipos son cada vez menos delicados y más personales, por lo que únicamente, se requiere de estándares para la administración de los equipos, y de conciencia para llevarlos a cabo.

Planes de contingencia.

Este punto tiene gran relación con el anterior, ya que en las aplicaciones centralizadas, existen planes de contingencia para el software instalado, en caso de algún siniestro. Pero estos son muy rígidos y complicados. En cambio en un sistema cliente/servidor deben de existir estos planes, pero son más sencillos y flexibles, pero se requiere de estándares que se cumplan, para poder llevar a cabo los planes de contingencia, que son mínimos. En tales planes encontramos los backup's y restore's de la información, que ahora son en línea.

Ayudas en línea

La naturaleza, de estos sistemas, con interfaces GUIs, nos proporciona, ayudas en línea, para cada opción del sistema, esta característica es necesaria en estos sistemas.

Distribución de software

Se debe de contemplar planes de actualización, para lograr distribuir en cada sitio, las nuevas versiones de software.

Minimización del mantenimiento

En el desarrollo de estos sistemas, una vez que este queda liberado, se minimiza el mantenimiento correctivo, debido a la modularidad e independencia del software y a los beneficios de la programación orientada a objetos.

Aumento de la velocidad de desarrollo en siguientes sistemas

El proceso de aprendizaje en el primer desarrollo cliente/servidor, es muy lento y difícil, debido a la falta de experiencia. Pero una vez logrado el objetivo, la curva de aprendizaje disminuye, y en los siguientes desarrollos el proceso de construcción se vuelve automático, debido a la ventaja de la reutilización del código de los objetos programados.

VENTAJAS Y DESVENTAJAS DE LOS SISTEMAS DISTRIBUIDOS

Ventajas

Cómputo distribuido a través de la red.

Esto se traduce en una distribución más efectiva del poder computacional, ya que en este modelo ninguna máquina es sobrecargada.
El mayor beneficio de negocios en un modelo cliente/servidor, es la división de tareas en dispositivos *front-end* y *back-end.*

Plataformas computacionales de menor costo

Con la introducción de un modelo cliente/servidor, los dispositivos mainframes, pueden ser utilizados como administradores de los datos corporativos o sistemas de almacenamiento, moviendo de esta forma las actividades diarias a plataformas de menor costo.

Menores costos de la administración de la red.

Los servidores son mucho más fáciles de hacerles servicio, mantenerlos, expanderlos y migrarlos que las *mainframes* debido a su función modular en la red, lo que provoca una reducción en los costos de mantenimiento.

Menor volumen de tráfico en la red.

En el modelo cliente/servidor se reduce la cantidad de datos moviéndose a través de la red, lo que trae importantes ventajas a redes grandes y con muchos usuarios, debido a que en la red solo viajan RPC

Incremento en la seguridad.

El modelo cliente/servidor mejora la seguridad permitiendo al servidor esforzarse con las reglas de seguridad. Los requerimientos de seguridad en las aplicaciones, se pueden programar en el software de operación del servidor y no en cada aplicación. Así los desarrolladores pueden concentrarse en maximizar el *performance* y las opciones de sus aplicaciones, y dejar que la seguridad sea una función del servidor.

Además el modelo cliente/servidor incrementa significativamente la seguridad, reduciendo el movimiento de información crítica a través de la red.

Incremento en la flexibilidad.

Los componentes front-end y back-end son independientes y por lo tanto requieren técnicas diferentes para ser creados y mantenidos. También esta flexibilidad existe en los diseñadores de software que no por ser muy eficientes para realizar sorprendentes interfaces de usuario, necesariamente lo serán para diseñar un sistema de manejo de bases de datos. La separación del trabajo de desarrollo significa que ambos componentes, front-end y back-end, pueden ser mantenidos, renovados y migrados transparentemente hacia otros.

Lo anterior se debe a que el modelo cliente/servidor está basado en una segmentación de roles entre los servidores, redes y clientes. Así la red puede ser cambiada por una más rápida, el sistema operativo del cliente puede ser cambiado por una nueva versión y el servidor puede ser reemplazado por un modelo más poderoso, todo esto de forma independiente y sin interrupciones significativas de los servicios.

Autonomía local

La distribución que nos ofrece esta tecnología, hace que cada elemento de una arquitectura de este tipo, sea responsable de sus recursos, tenga sus propias políticas de su funcionamiento, etc. Lo cual nos proporciona un descentralización de las funciones y de la autoridad, a nivel organización de una empresa.
Mientras que en las arquitectura monolíticas, las funciones se centralizan en un solo sitio, trayendo consigo un mayor numero de problemas y haciendo más difícil el control.

Minimización del mantenimiento correctivo

Los sistemas desarrollados en lenguajes como Cobol, RPG, etc. son difíciles y costosos en su mantenimiento, debido a la complejidad de su código, en cambio un sistema cliente/servidor bien analizado y diseñado minimiza el mantenimiento correctivo de los sistemas, debido a su estructura modular de los componentes del software.

Aumento de la calidad en los sistemas

Las posibilidades de accesar a diferentes bases de datos y recursos, hace que los sistemas sean más flexibles y con gran gama de funciones, para realizar.

Economía

Este punto se puede ver desde dos puntos de vista. La primera en términos de costos de comunicación. Si las bases de datos están en diferentes lugares geográficos, las aplicaciones y los recursos, hacen que en organizaciones grandes, esto sea muy económico. El segundo punto de vista radica en que el costo es menor cuando se invierte en maquinas pequeñas, pero con mayor poderío que un mainframe.
Estos dos punto anteriores son costos que en un principio son muy altos, pero a través del tiempo el costo se justifica y se reduce, ya que, no hay mantenimiento de equipos grandes.

Naturaleza de las organizaciones.

En la actualidad las organizaciones distribuyen sus operaciones en diversos lugares geográficos, desde una ciudad hasta un continente a otro. Así mismo la descentralización de las operaciones a diversas regiones, o la integración de diversos mercados, lo que propicia la unión de empresas de diversas regiones. Todo lo anterior en el ámbito organizacional es descentralización de las operaciones.

Aumento del rendimiento.

En este tipo de sistemas el rendimiento de las aplicaciones, aumenta considerablemente, debido a que:

- Los procesos en paralelo.
- El compartir recursos en diferentes sitios
- La forma en que trabajan los sistemas cliente/servidor, a través de los RPC

Lo anterior nos permite hacer transacciones en numerosos sitios, y que los procesos se hagan en
cooperación de diferentes procesadores o hosts.

Modularidad

El desarrollo de los sistemas cliente/servidor es a través de herramientas, que permiten la integración de estos, con otro sistemas de su mismo tipo, pero desarrollados con otras herramientas de diferente marca. Lo cual nos da la ventaja de ser un sistema abierto, y poder comunicar sistemas de diferentes organizaciones y así llegar a una integridad de los sistemas y de la información de las organizaciones.

Expandibilidad

En sistemas distribuidos, es más fácil incrementar el tamaño de las bases de datos y los recursos. Así mismo esta característica hace que los sistemas, se puedan comunicar de manera muy fácil, aun cuando sean sistemas homogéneos.

Compartir recursos.

Los sistemas distribuidos, por naturaleza comparten recursos o información, a través de la red.

Heterogeneidad

Esta arquitectura nos permite la comunicación con sistemas heterogéneos, definiendo así una arquitectura abierta.

Mayor a velocidad de desarrollo en siguientes sistemas

Una vez que se desarrollo el primer sistema cliente/servidor, el proceso de desarrollo de nuevos sistemas, se vuelve más rápido, debido a la facilidad de uso de librerías de código de objetos para nuevos sistemas.

Desventajas

Mal Funcionamiento

Si no se definen en forma clara las funciones tanto del cliente, como del servidor, el sistema puede tener un mal funcionamiento y un rendimiento bajo.

Peligro de trafico en la red

El desarrollo de un sistema cliente/servidor, requiere de una evaluación y planeación de la arquitectura, para minimizar el tráfico y evitar la saturación del medio de comunicación

Carencia de experiencia

Debido a que esta tecnología es nueva, no existe mucha gente experimentada, para el desarrollo de estos sistemas, pero la curva de aprendizaje de estos sistemas, se incrementan.

Complejidad

En los sistemas distribuidos, debido a que son más complejos que los sistemas centralizados, se incrementan el numero de problemas y variables que controlar. Por lo que se necesita de una cultura y de métodos de administración de estas aplicaciones.

Control distribuido

En estos tipos de sistemas, la ventaja de la distribución de tareas, es un factor vital, pero para llegar a garantizar esa distribución, existen muchos problemas a solucionar, tales como la sincronización y la coordinación. Ya que deben existir una serie de politicas, las cuales se deben de cumplir, para que se debe la distribución.

Miedo al cambio.

Muchos de los negocios y empresas, las cuales han invertido en sistemas de computo, que no son distribuidos. No invierten en el desarrollo de nuevas tecnologías debido a que son desconocidas para ellos, a pesar de que son más eficientes que las centralizadas, debido a que tienen miedo al cambio.

RESUMEN

En este capitulo logramos ver que los sistemas están presentes en cualquier ámbito de la actividad humana, que el mundo es gobernado por estos y el desarrollo de la humanidad se debe al mejoramiento e implantación de nuevos tipos de sistemas, tanto económicos, sociales, organizativos, científicos, etc.

En el caso de las organizaciones sociales, empresas, instituciones, entidades publicas, los sistemas de información y de procesos no son la exepción, sino que son un rubro de gran prioridad para el desarrollo de sistemas bien definidos, ya que estas necesitan información oportuna, así como el mejoramiento de sus procesos, para ser más eficientes. Por lo que la Informática surge como una herramienta para lograr los objetivos mencionados, ya que en forma sistemática define estos sistemas para ser llevados en forma electrónica, utilizando los recursos computacionales existentes y formas de administración de los procesos, para implementar sistemas de gran calidad y que resuelvan las necesidades de las organizaciones.

En el desarrollo de un sistema de información electrónico, principalmente se automatizan datos y procesos, los cuales se pueden implantar con diferentes estrategias, tales como; para datos: modelo de manejador de archivos, modelo jerárquico, de red o relacional, para procesos; arquitectura centralizada o distribuida(cliente/servidor).

Las estrategias para el desarrollo de sistemas electrónicos, se basan en las tecnologia de software y hardware, por lo cual en este trabajo se pretende hablar a cerca de una nueva estrategia para la creación de sistemas, esta es la filosofía cliente/servidor, la cual trata de una arquitectura distribuida en los elementos de computo. Estos elementos son el cliente y el servidor, los cuales son sistemas independientes, pero que trabajan en forma conjunta a través de una red de computo, en donde cada elemento realiza un trabajo especifico en el sistema, para satisfacer las necesidades del usuario.

El cliente se encarga la presentación de los datos y el control de los programas del sistema, por medio de una interface gráfica para el usuario, mientras que el servidor realiza la administración y almacenamiento de los datos, además de proveer al cliente de las solicitudes que este se le envie. Todo este trabajo se realiza en una red de computo.

CAPITULO 2
ENTORNO DE APLICACION

En la actualidad, la tecnología en computo ha evolucionado rápidamente, tanto en el área del hardware y del software, así como en las redes y comunicaciones, el surgimiento de nuevas concepciones del mundo de la informática como es el caso del paradigma de la orientación a objetos. Todo esto ha originado la comercialización de equipos y programas que pretenden hacer las cosas más sencillas, que la tecnología anterior en computo.

Pero en realidad, no necesariamente lo nuevo es mejor o más eficiente que lo viejo, ya que en el mundo de la informática, la misión de los profesionistas de esta área es ***" aplicar los recursos informáticos para hacer más productiva una entidad"***, lo cual implica entregar productos que al cliente final realmente le sirvan y que se aplique a sus necesidades, sin importar si es nueva o vieja la tecnología en computo.

Un ejemplo claro de esta situación se muestra en los Estados Unidos, en donde miles de empresas de gran tamaño, sus sistemas de información están basados en tecnologías de computo de los años setentas u ochentas, en donde utilizan arquitecturas centralizadas, en equipos mainframes y utilizando lenguajes como COBOL, RPG, etc. Ya que estos sistemas realmente satisfacen sus necesidades de proceso e información. Estas empresas en sus áreas de informática no pretenden cambiar estos sistemas, por nuevas tecnologías, ya que no requieren un a cambio de algo que en realidad les sirve. Por lo cual no siempre lo más novedoso es lo mejor, sino lo que en realidad funciona es lo que se utiliza.

En cambio en las empresa en las cuales no existen procesos automatizados o se cuenta con sistemas que no cumplen con las necesidades de estas y que son difíciles de mantener. Es necesario hacer un análisis de cambio, pero de cambio hacia la productividad, ya sea utilizando nuevas tecnologías que en realidad funcionen, para hacer más eficiente a esa empresa. En ese análisis de cambio debemos de ver que tecnología es la más apropiada.

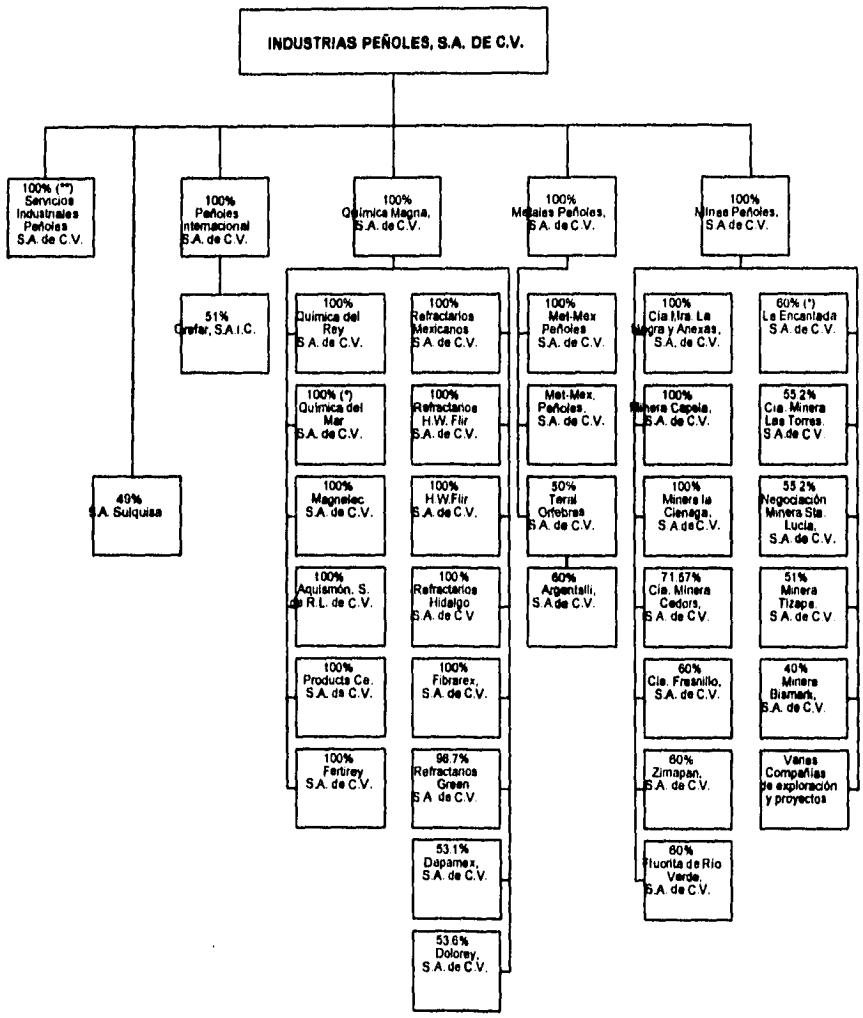
Por lo anterior este trabajo pretende dar auge a la utilización de una nueva tecnología, la cual hace más eficiente el procesamiento de la información. Esta es la arquitectura cliente/servidor, la cual trae un cambio en la forma de automatizar los procesos, así como en la forma de construcción de los sistemas, ya que implica la utilización de nuevos equipos de computo y de software para desarrollo y producción.

El origen de esta tesis se presenta a partir de una metodología para al análisis, diseño y construcción de un sistema cliente/servidor, como nueva tecnología para un cambio de mejora, basado en un caso real, de una empresa Mexicana. La cual en su área de ventas, cuenta con un sistema de computo de hace 10 años, el cual era muy caro su mantenimiento, además de que no cubría todas las necesidades reales del negocio, por lo cual se planteo un cambio de mejora, esto por medio del desarrollo de un nuevos sistema, el cual reducirá el costo de su mantenimiento, además de que cubra todas las necesidades reales del negocio. Esta empresa de la cual hablamos es Servicios Industriales Peñoles. S.A., en su División Químicos Industriales. La cual pertenece a un Industrias Peñoles, que esta conformada por una serie de empresas las cuales se dedican al ramo: minero, metalúrgico y químico. Servicios Industriales Peñoles, S.A. (SIPSA) su labor es otorgar servicios administrativos y tecnológicos a las empresas integrantes de Industrias Peñoles. SIPSA se encuentra segmentada en Divisiones, cada una de estas otorga esos servicios contables, de ventas y informáticos a cada compañía por ramo. Una de estas divisiones en Químicos Industriales, la cual soporta a todas las empresas que se dedican a la producción de materiales químicos.

INDUSTRIAS PEÑOLES

La Compañía minera de Peñoles se constituyo el primero de marzo de 1887, gracias a un grupo de inversionistas mexicanos, encabezados por el Sr. Raúl Balléres, quienes se dedicaron a la tarea de explotar unas pequeñas minas en la región de Peñoles, Durango, trabajando en yacimientos de plomo y planta. Lo que dio lugar a lo que en la actualidad se conoce como Industrias Peñoles.

Industriales Peñoles, S.A. de C.V. , es un grupo de empresas las cuales se dedican al ramo minero, químico, refractario y cerámica, Industrias Peñoles la cual formada parte del Grupo Bal, en donde el accionista mayoritario es el Lic. Alberto Balléres, este grupo esta constituido por empresas como Grupo Nacional Provincial, Palacio de Hierro, Bancomer, el ITAM, entre otras más.



Organigrama Industrias Peñoles[1]

[1] Informe Ejecutivo Industrias Peñoles, p2

Servicios Industriales Peñoles, S.A.

Servicios Industriales Peñoles, S.A. de C.V. es una empresa prestadora de servicios administrativos, de planeación y desarrollo, de asesoría técnica, de comercialización, capacitación y de todo aquello que contribuya al mejoramiento de las actividades productivas de cada una de las diferentes empresas de Industrias Peñoles, que a su vez se conjuntan en Grupos y Divisiones.
Sus oficinas Corporativas se encuentran en la Cd. de México, en el Centro Corporativo Plata ubicado en la Calle Río de la Plata 48 Col Cuahutemoc, Delegación Cuahutemoc.

Esta empresa, controlan a más de 35 unidades productivas en por lo menos 30 localidades diferentes del país, además de oficinas comercializadoras en Brasil, Venezuela y Estados Unidos; así como la participación en sociedades con empresas en Argentina y España, y la distribución en Europa a través de una bodega en Amberes, un Centro de Investigaciones Técnicas, en Monterrey Nuevo León, así como de Investigaciones Geológicas en Chihuahua, Chihuahua. Además se consolida como una de las industrias de mayor importancia que cotizan en la Bolsa Mexicana de Valores.

GRUPO MINAS

-División Minas

Formada por 20 unidades instaladas en distintas partes de la República, como son Coahuila, Colima, Chihuahua, Durango, Edo. de México, Guanajuato, Guerrero, Hidalgo, Jalisco, Michoacán, Querétaro, Sonora y Zacatecas, se dedica a extraer: plata, plomo, cobre, zinc, fluorita, barita y oro. Estos minerales (a excepción de la fluorita se venden a otros compradores) se concentran y transportan a la Planta Metalúrgica (MET-MEX Peñoles) para procesarse.

-División Exploración

Esta División realiza estudios geológicos para la localización de yacimientos económicamente explotables.

DIVISION METALES

Ubicado en Torreón Coahuila, Met-Mex Peñoles, S.A. de C.V., la cual se dedica a transformar los minerales (procedentes de las empresas mineras de Peñoles) de minerales crudos a metales afinados, como son: Plata, plomo, zinc; cadmio, óxido de cadmio, bismuto; sulfato de amonio , trióxido de antimonio, ácido sulfúrico y plomos antimoniales.

DIVISION QUIMICOS

Constituida por las empresas: Química del Rey, S.A. de C.V.(Coahuila), Química del Mar, S.A. de C.V.(Tamaulipas) y General Products Co (Edo. de México) , produce principalmente: óxido de magnesio, sulfato de sodio, óxido de zinc, hidrosulfito de sodio y zinc, sulfoxilato de sodio y zinc, cloruro de sodio, cloruro de zinc y amonio.

DIVISION DE LOGISTICA Y ABASTECIMIENTOS

Es responsable de la adquisición de bienes y servicios, resguardo y custodia de los materiales en cuestión, para asegurar la continuidad de las operaciones de Peñoles.

DIVISION DE FINANZAS

Se dedica principalmente a la administración de recursos financieros, así como a la realización de inversiones para la construcción ampliación, modernización, reposición y desarrollo de las Plantas y Unidades de Peñoles. Con lo anterior apoya el crecimiento y la consolidación de las Empresas y operaciones del Grupo.

DIVISION DE RELACIONES

Tiene por objeto mantener relaciones obrero-patronal, en el marco de cooperaciones y respeto; tiene también como misión, mejorar la productividad y calidad fortaleciendo los
equipos de trabajo por medio de programas de capacitación, de seguridad e higiene, de desarrollo gerencial y de mandos intermedios, de reclutamiento en las Universidades, etc.

DIVISION DE PLANEACION Y DESARROLLO

Este grupo se encarga de dar soporte técnico y apoyo en las operaciones, tanto para la optimización de los procesos, tanto como los desarrollo de nuevos productos. También tiene a su cargo el desarrollo de proyectos de expansión, diversificación y adquisición de empresas; estudio de costos de inversión, operación, revisión de planos de ingeniería y proceso. Además tiene a su cargo las obras de ampliación, instalación y montaje de las diversas Unidades y Plantas del Grupo.

DIVISION QUIMICOS

La División Químicos, la cual nos interesa conocer en especifico, soporta a 3 compañías Química del Rey, S.A. de C.V. , Química del Mar, S.A. de C.V. y General Products Company, S.A. de C.V. , las cuales se dedican a la producción de químicos industriales, para proveer a diferentes industrias de insumos. Entre sus principales productos se encuentran los siguientes:

Química del Rey, S.A. de C.V. o Quirey, ubica en Laguna del Rey Coahuila, a unos 150 kilómetros de la Cd. de Torreón:

Sulfato de sodio anhídro
Oxido de magnesio grado refractario
Oxido de magnesio grado cáustico

Química del Mar, S.A. de C.V. o Quimar, ubicada en Tampico Tamaulipas:

Oxido de magnesio grado refractario
Oxido de magnesio grado cáustico
Oxido de Calcio

General Products Company o GPC ubicada en Santa Clara, Estado de México.

Hidrosulfito de sodio
Hidrosulfito de zincSulfexilato de sodio
Oxido de zinc proceso Americano
Oxido de zinc proceso Fraces
Polvo de zinc
Cloruro de zinc y amonio
Azufre grado hulero

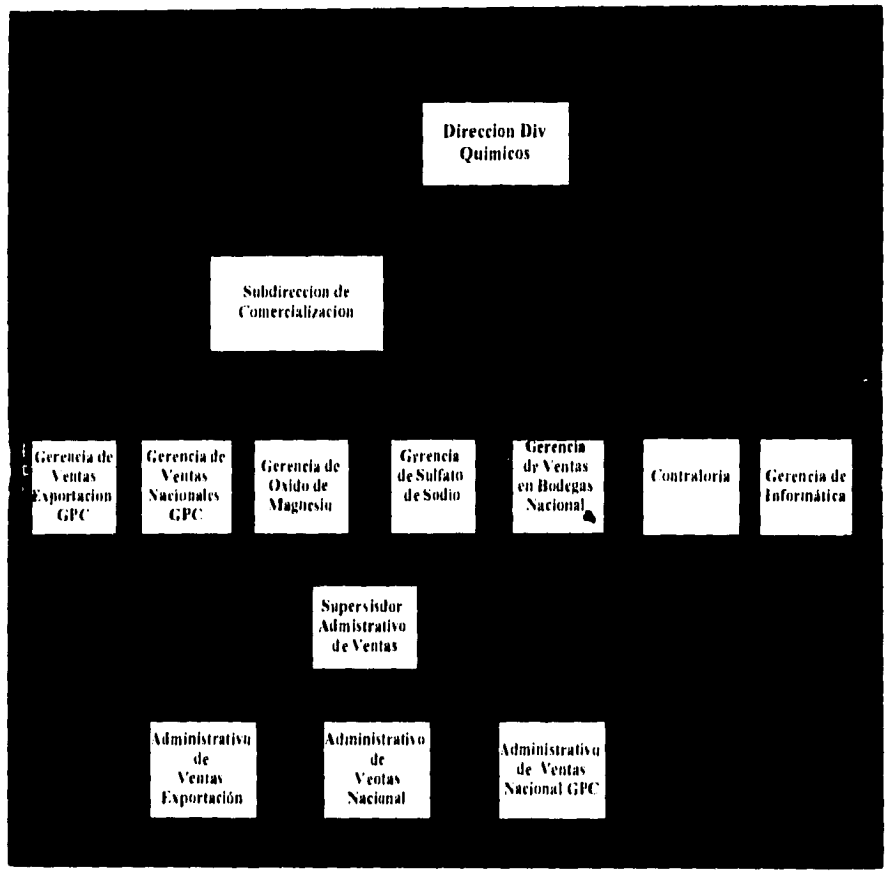
Los productos mencionados anteriormente abarcan una gran variedad de industrias, además de que existen una nula competencia en América, aun que en el ámbito mundial existen países como China y Australia, las cuales desarrollan estos tipos de productos, y empiezan a dar competencia a Peñoles, por lo que la División debe de ofrecer productos de calidad, además de servicios que apoyen a la venta de los productos, tales como créditos, transportación, etc. Para la comercialización de los productos antes mencionados la División cuenta con oficinas y Bodegas comercializadoras en diferentes localidades del mundo, tales como:

Bodegas Nacionales	Bodega Ciprés CD de México Bodega Monterrey
Bodegas y Oficinas Internacionales	**Norteamérica** Oficina Peñoles Metals and Chemical en New York(PEMECHE)
	Sudamérica Oficina Peñoles Venezuela Bodega y Oficina Quirey Du Brasil Bodega Costa Rica
	Europa Bodega Amberes Bodega España

Bodegas y Compañías en el mundo

Como podemos observar la compañía abarca los mercados de Norte, Centro y Sur América y Europa, además de tener exportaciones directas desde México hacia Japón.
Así mismo también cuenta con compañías consignatarias de sus productos, en Europa y América Latina.

Organizacion De La Division Quimicos



Como se puede observar en el organigrama; existen tres áreas en esta división; Comercialización, Contraloría e Informática. Estas áreas son las encargadas de dar servicios a las compañías Quimar, Quirey y GPC, de los demás servicios administrativos se encargan otras Divisiones, tal es el caso de la División de Relaciones(Recursos humanos), División Logística(Embarques de materiales), etc. Para nuestro caso nos interesa conocer el área de Comercialización.

Comercializacion

La organización del área comercial podemos dividirla en dos grandes mercados Nacional y Exportación, y una subdivisión de estos por productos, ya que existen ventas nacionales para oxido de magnesio y ventas exportación para sulfato de sodio. Ya que para cada producto existe un gerente de ventas, el cual se encarga de vender tanto en el país como en el exterior esos productos.

Subdirección de Comercialización: Esta se encarga de definir los planes estratégicos y de penetración de mercado para los productos de la División Químicos, así como del control de las ventas de todos los productos y de las bodegas y oficinas de ventas en la república y el extranjero.

Gerencia de Ventas Exportación GPC: Se encarga de la comercialización y venta de los productos de la compañía General Products Company en el extranjero.

Gerencia de Ventas Nacionales GPC: Su función es la venta de comercialización y venta de los productos de General Products Company en el mercado Nacional.

Gerencia de Oxido de Magnesio: Se encarga de la comercialización y venta del Oxido de Magnesio producido en Quimar y Quirey , tanto en el mercado Nacional y Extranjero.

Gerencia de Sulfato de Sodio : Gerencia la cual tiene por función la comercialización y venta , del Sulfato de sodio de Quirey , en el mercado nacional y exportación

Gerencia de Ventas de Bodega Nacional: Se encarga de la comercialización de los productos de Quimar y Quirey en las bodegas nacionales .

Una vez conocido cuales son las funciones que cada gerencia , nos falta definir una parte importante de la comercialización de los productos, esta es una unidad de tipo staff , la cual otorga servicios a todas las gerencias de ventas, ya que entre sus funciones están; recibir pedidos, facturar, hacer reportes , etc., esta unidad la conforman los siguientes puestos:

Supervisor Administrativo de Ventas (SAV) : Su función es la administración de las ventas que realiza la división , así como el de la recopilación de información de ventas, gastos de mercado, embarques, etc. en una venta.

Administrativo de Ventas Exportación (AVE): Entre sus funciones están el de recibir pedidos, facturar, controlar embarques y hacer cierres de mes, para las ventas de Exportación de los productos de Quimar, Quirey y GPC.

Administrativo de Ventas Nacional (AVN) : Se encarga de recibir pedidos, facturar y hacer cierres de mes, para las ventas Nacionales de los productos de Quimar y Quirey, incluyendo las bodegas.

Administrativo de Ventas Nacional GPC(AVNGPC): Se encarga de recibir pedidos, facturar y hacer cierres de mes, para las ventas Nacionales de los productos de GPC.

Una vez que el gerente de ventas, contacto un cliente para surtirle un pedido, estos pasan la estafeta a las área administrativa de ventas nacionales o de exportación dependiendo el caso de la venta, para continuar con todo el ciclo de ventas.

Proceso De Ventas De La Division Quimicos

El proceso actual de ventas, que realiza la División Químicos, abarca desde la recepción de Pedidos, hasta la facturación de estos Pedidos, de la siguiente forma:

Ventas Nacionales

- Pedidos Nacional
  - Pedidos de Bodega-Cliente
  - Pedidos de Planta-Cliente
- Embarques de Planta
  - Embarques de Planta a Bodegas
  - Embarques de Planta a Cliente
- Facturación
  - Ventas de Bodega
  - Ventas de Planta a Cliente
- Póliza de Ventas

Ventas Exportación

- Pedidos Exportación
  - Pedidos para Bodegas en el extranjero
  - Pedidos para Bodegas en el extrajero a consignación
  - Pedidos de México a Clientes
- Tramite de documentación para la exportación
- Embarques al Extranjero
  - Embarques Planta a Bodegas en el extranjero
  - Embarques de Planta a Bodegas en consignación
  - Embarques de Planta a Clientes en el Extranjero
- Facturación de Pedidos del Cliente
  - Facturación de Ventas de Bodega
  - Facturación de Ventas de Bodega a Consignación
  - Facturación de Ventas desde México
- Pólizas de Ventas
- Gastos de Mercado

PROBLEMATICA ACTUAL

En la actualidad este proceso, es llevado a cabo en su gran mayoría en forma manual, mientras que solo la parte de pedidos y facturación de ventas nacional y exportación se lleva a cabo en un sistema computarizado. Este sistema desarrollado hace 10 años bajo la arquitectura centralizada de mainframes (HP3000), en un lenguaje llamado SISINF, el cual cuenta con una base de datos jerárquica.

Ventajas Del Sistema Actual

- Velocidad de respuesta en el sistema actual, debido a que cuanta con una base de datos jerárquica.
- El sistema automatizado satisface el 100% de las necesidades de pedidos y facturación para ventas nacional y exportación.

Desventajas

- Problemas de mantenimiento del sistema, ya que no existen manuales de usuario, técnicos y de operación, su mantenimiento es a base de conocimientos empíricos, además que el mantenimiento correctivo y adaptativo es muy costoso, ya que es necesario tener a programadores cien porciento dedicados a esta tarea.
- Tecnología en hardware y software obsoleta
- Falta de control en los procesos que se necesitan implementar, como son:

Ventas Nacionales
- Embarques desde Planta a Bodegas
- Embarques desde Planta al Cliente
- Entregas de Embarques a Clientes
- Entregas de Embarques a Bodegas de Venta
- Facturación desde Bodegas de Ventas
- Control de Inventarios
- Estadisticas

Ventas Exportación
- Embarques desde Planta a Bodegas en el Extranjero
- Embarques desde Planta al Cliente al Extranjero
- Embarques desde Planta a Bodegas en Consignación
- Control de Embarques
- Entregas de Embarques a Clientes en el Extranjero
- Entregas de Embarques a Bodegas de Venta en el Extranjero
- Entregas de Embarques a Bodegas en Consignación
- Control de Inventarios de Bodegas del Extranjero
- Control de Inventarios de Bodegas a Consignación
- Entrega de Pedidos de Bodega en el Extranjero
- Control de Gastos de Ventas

- Estadísticas

- La implementación de estos procesos en el sistema automatizado actual sería muy costoso y laborioso, debido a la plataforma actual
- La falta de un verdadero sistema de información, el cual proporcione información a todos los niveles de la organización, desde el operativo hasta el directivo.

OBJETIVO DE LA TESIS

Por el marco de referencia anterior, se tomo la decisión de construir un nuevos sistema de ventas, el cual incluya los procesos actuales tanto manuales y computarizados, como los nuevos requerimientos, pero bajo un esquema de mejora de todos los procesos implementados. Por este motivo se tomo la decisión de seleccionar la arquitectura cliente/servidor como plataforma para el nuevo sistema, debido a sus ventajas y beneficios.

La decisión de llevar a cabo un sistema bajo esta arquitectura, rompe la forma tradicional de hacer un análisis, diseño y codificación de los sistemas en esta gerencia, por lo que era necesario definir una método de trabajo. Por lo cual el objetivo de la tesis es " Presentar un método para el análisis, diseño y construcción de un sistema cliente/servidor " basado en una serie de investigaciones bibliográficas y de asesorías con despachos de software, la cual se fuera la más eficiente para el desarrollo del Sistema de Ventas de la División Químicos de Servicios Industriales Peñoles, S.A:. de C.V.

Los beneficios los cuales se obtendrán del desarrollo de un sistema cliente/servidor, son propios para el sistema de ventas y para la Gerencia de informática.

Beneficios Para El Sistema De Ventas

- Mejoramiento de los procesos automatizados
- Aumento de los procesos automatizados
- Mayor dominio y control de la información
- Aumento en la productividad del área
- Un sistema que otorgue información oportuna y veraz a todos los niveles de la organización.

Beneficios De La Gerencia De Informática

- Creación de una metodología y estándares para este tipo de sistemas
- Creación de sistemas más amigables
- Generación de código para ser reutilizado en siguientes sistemas
- Disminución del costo de desarrollo y mantenimiento de los sistemas
- Aumento en la productividad en el desarrollo de sistemas
- Integración con los Sistemas Abiertos

RESUMEN

En la actualidad existen en México una gran cantidad de empresas las cuales, están acostumbradas a desarrollar y mantener sistemas de tecnología anterior, los cuales se basan en arquitecturas centralizadas y en software de tipo procedural. Estos tipos de sistemas son muy costosos de mantener, debido a la naturaleza de los lenguajes de programación, otros realmente no satisfacen las necesidades de los negocios. Un caso de este tipo de empresas es Servicios Industriales Peñoles, en su División Químicos, en donde sus sistemas son de tipo centralizados y costoso de mantener, en forma especial su sistema de ventas. Esta empresa la cual se dedica a la comercialización de productos químicos, en el mercado nacional e internacional, requiere de sistemas mas eficientes, que solucionen las necesidades de información y procesos, y además de sean sencillos de mantener.

Por lo cual el profesional en Informática, debe de dar soluciones eficientes a las organizaciones, es por esto que se propuso el hacer un cambio de sistemas en el área de comercialización de la División Químicos, tanto a nivel técnico como a nivel funcional, el cual mejorara los proceso de información y fuera más fácil su desarrollo y mantenimiento. Por lo se propuso la arquitectura cliente/servidor como plataforma del sistema, pero esto trajo consigo un cambio en la forma de desarrollo de los sistemas, ya que en la gerencia de informática se habían desarrollado sistemas centralizados y procedurales. Y era necesario contar con una metodología de análisis, diseño y construcción para un sistema cliente/servidor, lo cual me permitió llevar una investigación y proponer un método de desarrollo para este tipo de plataformas, que es lo que se describe en esta tesis.

CAPITULO 3
ESTRATEGIA A SEGUIR

A través de la historia de la computación, se han creado diversas metodologías para el desarrollo de sistemas, las cuales proveen un marco de referencia para la arquitectura, la infraestructura en hardware y software, la planeación, el análisis, el diseño, la programación y el mantenimiento de los sistemas de información. Así mismo incluye métodos de trabajo, tareas, estándares entre otras cosas más.

El desarrollo de un sistema, en forma tradicional ha consistido en seguir las etapas y técnicas que nos proporcionan las metodologías, para construir sistemas basados en las arquitecturas centralizadas y/o en software de tipo procedimental, tales es el caso de metodologías como: Ingeniería de Información, Jackson, BSP, etc. La evolución de la tecnología en computo, las redes de comunicación, las nuevas necesidades de las empresas y los nuevos paradigma para desarrollar sistemas, han originado el surgimiento de nuevas plataformas para la construcción de los sistemas, así como nuevas metodologías para su desarrollo. Un ejemplo claro de esto es la tecnología cliente/servidor, la cual surge con motivo del desarrollo de nuevos equipos de computo, más pequeños y poderosos que los mainframe, como es el caso de las PC y estaciones de trabajo, así como el surgimiento de las redes de comunicación y nuevos software como lo son las bases de datos relacionales y los lenguajes de cuarta generación.

La tecnología cliente/servidor como plataforma nueva de procesamiento de información, debe ser desarrollada siguiendo ciertos métodos y herramientas, así como un sistema en Cobol o en RPG, debe ser desarrollado siguiendo cierta metodología, también un sistema cliente/servidor. Por lo cual existen diversos enfoques o paradigmas de hacer las cosas.

METODOLOGIAS CLIENTE/SERVIDOR

Las metodología o métodos para desarrollos cliente/servidor, se basan en la concepción o análisis de los datos y procesos de un contexto, lo cual en metodologías anteriores también se daba, pero ahora se busca que desde el análisis los elementos de un sistema se conceptualizan como objetos o entidades, los cuales son de interés al negocio. Para que a partir de ese tipo de análisis, pasemos al diseño de dos elementos; el cliente y el servidor

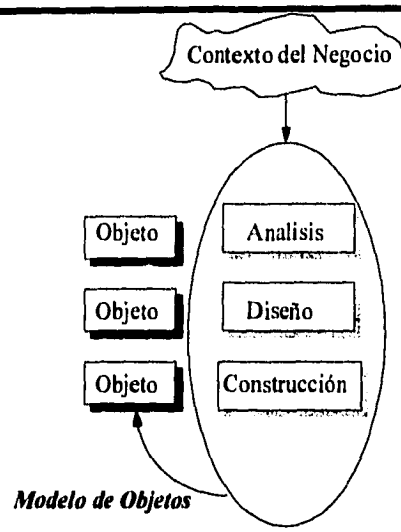
"Las metodologías actuales para este tipo de sistemas, las podemos clasificar en tres categorías:"[1]

1. Metodología para migrar sistemas a cliente/servidor.
2. Metodologías de análisis y diseño orientado a objetos.
3. Metodologías para sistemas cliente/servidor, bajo herramientas GUI/RDBMS.

La primera clasificación, se basa en el concepto de reingeniería del software, en donde aplicaciones centralizadas bajo lenguajes como Cobol o RPG, son rediseñados para cliente/servidor, tomando el código fuente de esos lenguajes para ser rediseñados por medio de herramientas CASE, para obtener un modelo conceptual de ese sistema, es decir pasamos de la codificación de un sistema hasta el análisis, por lo cual se le llama reingeniería, para poder partir hacia una nuevo diseño del sistema, basado en cliente/servidor.

La segunda se basa en la teoría de la orientación a objetos, en donde desde análisis se plasman en modelos las necesidades del negocio en forma de objetos que sufren una serie de transformaciones, cada objetos es un elemento del sistema analizado, los cuales se componen de datos y procesos. Una vez que se definió el modelo de objetos de análisis, en diseño se procede a realizar la creación de clases y eventos, los cuales son la implantación de los elementos del análisis en código fuente para estructuras de datos orientadas a objetos y en lenguajes basados en eventos. Entre algunas de estas metodologías tenemos a RAD, OO Methodology, Análisis y Diseño orientado a objetos, OMT , etc.

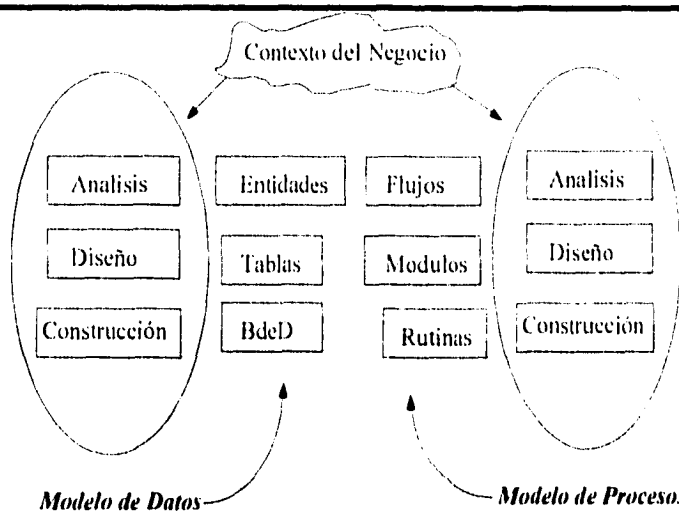
[1] DBMS Database Client/Server Solutions, , Following a Client/Server Database Methodology, p50



Modelo orientado a objetos[2]

La última categoría, se basa en la construcción de un sistema cliente/servidor, basado en un back-end de tipo relacional y el un front-end GUI orientado a eventos. En esta categoría se utilizan bajo metodologías como Ingeniería de Información, Yourdon/Demarco, Gane/Sarson, Análisis y Diseño Orientado a Objetos, RAD, etc. Por lo cual es considerada una categoría híbrida.

[2] idem p50



Modelo hibrido[3]

La selección de una metodologías para el desarrollo de un sistema cliente/servidor, depende la tecnología actual que soporte a ese paradigma y de la capacidad del personal de informática para realizar un cambio de plataforma y de los conocimientos en estas. Ya que si nuestro análisis y diseño es totalmente orientado a objetos, en la actualidad no hay aun herramientas de software que soporten estructuras de datos orientados a objetos, como lo sería una base de datos de objetos por lo cual sería difícil su implantación. Pero lo importante de una metodología cliente/servidor es que se pueda construir un sistema el cual siga los objetivos y características de la filosofía cliente/servidor, como lo es la autonomía y cooperación de los elementos.

[3] idem p50

ESTRATEGIA

La estrategia para el análisis y diseño de un sistema cliente/servidor, la cual se llevó a cabo en Servicios Industriales Peñoles, S.A. fue una metodología la cual se tomo a partir de la tecnología en software y hardware actual. Por lo que se opto por seguir la tercera categoría de las antes mencionadas. Utilizando un enfoque orientado a objetos, debido a que el cliente debe ser la interface de control de los datos y procesos, mientras que para el servidor se siguió un análisis y diseño relacional.
La estructuración de los pasos a seguir para el desarrollo del sistema, partió de una serie de investigaciones en bibliografía y revistas, así como asesoría con diversos despachos de informática como es el caso de CIE (Consultores en Informática Empresarial) representantes de GUPTA en México y de Factum Informática representantes de Knolegewre. Inc. Lo cual dio como origen el armado de una metodología propia para la Gerencia de Informática de la compañía, utilizando bases de datos relaciónales y lenguajes orientados a objetos.
La metodología de desarrollo, comprende las tradicionales etapas de análisis, diseño y construcción.

Análisis

En el análisis del contexto del sistema, para poder identificar los requerimientos del usuario, se parte del desarrollo de varios modelos como productos de esta etapa, los cuales son; modelo entidad/objeto, modelo dinámico y modelo funcional. En el modelo entidad/objeto podemos identificar aquellos elementos que son de interés para el sistema, definiendo sus características de cada una, en el modelo dinámico se describen los eventos y estados de los elementos del sistema y en el modelo funcional se describen los procesos y flujos de datos que intervienen en nuestro contexto. Con estos modelos podemos identificar la realidad del sistema y poder plasmarlo en un diseño computacional.

Los tres modelos anteriores se conjuntan en uno solo, en el modelo de análisis.

Insumos	Requerimientos del usuario
Actividades	
Modelo de Análisis	1)Diagrama de Descomposición
Modelo Entidad/Objeto	2) Diagrama Entidad-Relación (DER) 3) Diagrama de Descripción de Entidades (DDE) 4) Dominios de Datos
Modelo Dinámico	5) Diagrama de Transición de Estados (DTE) 6) Diagrama de Esquema de Eventos (DEE)
Modelo Funcional	7) Diagrama de Contexto 8) Diagramas de flujo de datos. (DFD) 9) Texto Estructurado
Productos	Modelo de Entidad/Objeto Modelo Dinámico Modelo de Procesos

Diseño

La fase de diseño, se divide en dos elementos; el diseño de la base de datos y del prototipo GUI. Para el diseño de la base de datos, la tarea a seguir, es la generación física de la misma, en el servidor, por lo que en esta etapa se mezcla el diseño con la construcción, por lo que al finalizar esta etapa, la base de datos deberá estar físicamente en el servidor, así como los mecanismos para garantizar la integridad de datos y la referencial, para esto tomaremos el modelo entidad/objeto de análisis. El diseño del prototipo, trata de la generación de una aplicación patrón, para que las demás aplicaciones del sistema funcionen igual, este prototipo incluye el diseño de librerías de objetos, estándares de programación y procedimientos.

Insumos	Modelo Entidad/Objeto
Actividades	
Diseño para Servidor	1) Esquema Relacional de la Base de Datos 2) Mecanismos de integridad de datos 3) Mecanismo de integridad referencial
Diseño para Cliente	1) Diagrama de Flujo de Eventos 2) Librerías de Clases 3) Funciones de programación 4) Estándares de Programación 5) Prototipos de Aplicaciones
Productos	Esquema de la base de datos Librerías de clases Funciones de programación Prototipo de aplicaciones Diagrama de flujo de eventos

Para el desarrollo de esta fase, solo es necesario tomar el modelo entidad/objeto, para crear la base de datos.

Construcción

En la fase de diseño de la cual generamos la base de datos y un prototipo GUI, ahora partiremos a la construcción de las aplicaciones. Esta fase debe de realizarse en forma modular, partir de los procesos o aplicaciones, hasta llegar a terminar el sistema. Esta fase comienza con la programación en el servidor de disparadores, procedimientos almacenados y vistas que involucren a las tablas que intervienen en el proceso, para posteriormente construir la aplicación GUI del proceso en cuestión. El proceso de programación de disparadores, vistas y procesos almacenados se realiza para dar facilidades de rendimiento y velocidad a la explotación del cliente GUI de la base de datos.

Insumos	Modelo de Procesos (Análisis) Modelo de Eventos (Análisis) Esquema de la base de datos (Diseño) Librerías de clases (Diseño) Funciones de programación (Diseño) Prototipo de aplicaciones (Diseño) Diagrama de flujo de eventos (Diseño)
Actividades	1) Construcción de la aplicación GUI 2) Programación de Disparadores por proceso 3) Programación de vistas en el servidor por proceso 4) Programación de procedimientos almacenados por proceso 5) Construcción de Reportes Impresos
Productos	Sistema Cliente/Servidor

Una vez que terminamos la fase de construcción, contamos con un servidor de datos, como un ente independiente, capaz de responder a cualquier solicitud de un cliente propio o externo, el cual mantendrá los datos seguros e íntegros, mientras que por parte del cliente se tienen aplicaciones que satisfacen los requerimientos de información y proceso de los usuarios finales.

RESUMEN

De este capitulo podemos resumir que las metodología para el desarrollo de un sistema cliente/servidor, se categorizan en tres rangos; la primera se basa en la reingenieria de sistemas, la segunda en un análisis y diseño completamente orientado a objetos, mientras que la última se enfoca a un servidor relacional y a un cliente orientado a objetos. En la División Químicos se decidió emplear la tercer categoria, armando un método propia para el análisis, diseño y desarrollo, basándonos en investigaciones bibliográficas y en asesorias con despachos del área de Informática, por lo que se logro definir una forma para el desarrollo de un sistema cliente/servidor

En la metodología de desarrollo se contemplan tres fases, análisis, diseño y construcción. En la primera se desarrollan varios modelos: entidad/objeto, dinámico y funcional, con los cuales modelamos la realidad de un contexto, en el diseño partimos de la construcción de dos elementos, la base de datos y el prototipo. Para el primero nos basamos en el modelo entidad/objetos para generar una base de datos relacional, segura, integra y confiable, la cual debe de estar generada físicamente en el servidor al finalizar esta fase, mientras que el diseño del prototipo modelo, estándares, librerías de objetos y procedimientos servirán para la construcción de las demás aplicaciones. Por último la construcción de las aplicaciones GUI, parte de la explotación de los datos del servidor, siguiendo los requerimientos de cada proceso identificados en los modelos funcional y de eventos de análisis, basándonos en el prototipo y las librerías de funciones y objetos. Una vez realizados estas fases obtendremos dos elementos; el Cliente y el Servidor.

CAPITULO 4
ANALISIS DE SISTEMAS

El desarrollo de un sistema automatizado, el cual soporte las necesidades de una organización y más, es una labor muy delicada, ya que es necesario poder identificar los requerimientos de las áreas usuarias, las cuales intervienen dentro del sistema.

Para lograr lo anterior el análisis de sistemas tiene el propósito de estudiar las políticas del usuario y el esquema del sistema, en una especificación estructurada, para lograr *"(1) identificar las necesidades del cliente (usuario), (2)asignar funciones al software, hardware y gente, y (3)crear una definición del sistema"*[1].

La identificación de las necesidades es el punto de partida, en donde el analista tendrá que asistir al usuario, para definir la información que va producir el sistema y la información que se va suministrar.

La asignación de funciones, a los elementos que integran el sistema, se comienzan a definir, ya que la identificación de las necesidades, nos indica como las llevaremos a cabo y quien la llevará.

Por último tendremos que hacer una definición del sistema, en un modelo, el cual muestre los requerimientos del usuario a automatizar.

La tarea de análisis de los requerimientos es un proceso de descubrimiento y refinamiento, ya que se analiza y asigna a los distintos elementos del programa las soluciones y alternativas. En el desarrollo del software, el cliente (usuario) tendrá un papel activo en la especificación de los requerimientos, y este tendrá conceptos nebulosos de lo que se desea, por lo que el analista tienen en sus manos el saber guiar al usuario en lo que se desea y de como se deberá de plasmar.

El producto de la fase de análisis, para un sistema cliente/servidor, es la definición de tres modelos:

- Modelo de entidad/objeto
- Modelo dinámico
- Modelo funcional

En estos modelos se definen dos elementos esenciales, datos y procesos que intervienen en el contexto del sistema que nos interesa, estos modelos los desarrollaremos a partir de diversos diagramas que en su momento se mencionarán.

[1] Ingenieria de Software, Roger S. Pressman, p155

Modelo de Entidad/Objeto

Este modelo describe los elementos que intervienen en un contexto dado, definiendo sus atributos y relaciones entre cada uno de estos. En un sistema cliente/servidor, orientado hacia un back-end relacional, estos elementos se definen como entidades para la creación de tablas, mientras que en el front-end que funciona orientado a objetos, la conceptualización de estos elementos se conocen como objetos. Pero en realidad los elementos son los mismos, en sus atributos y relaciones, lo único diferente es su conceptualización. Por lo tanto los diagrama para definir este modelo son:

- Diagrama Entidad-Relación o Diagrama de Clases
- Diagrama de Descripción de Entidades

Modelo Dinámico

Un elemento que interviene en un mundo dinámico, sufre cambios, esto debido a que interactua con otros elementos. Para el análisis de cualquier sistema, debemos identificar los estados de un objeto, que sufre como consecuencia de los cambios de interactuar con el sistema. Por lo cual debemos definir el :

- Diagrama de Transición de Estados
- Diagrama de esquema de eventos

Estos diagramas nos muestra los eventos- estímulos y respuestas que tienen una entidad u objetos en un sistema dado.

Modelo Funcional

El modelo funcional nos muestra los flujos de datos y sus transformaciones a través de procesos y funciones en un escenario determinado, para definir este modelo contamos con el:

- Diagrama de Flujo de Datos
- Texto Estructurado

Para poder agrupar cada uno de estos modelos y ver su interacción, definiremos *el modelo de análisis* el cual se basa en el Diagrama de Descomposición, el cual define un marco de referencia para conjuntar el modelo de entidad-objeto, modelo dinámico y el modelo funcional, para el escenario al cual nos referimos.

ANALISIS DEL SISTEMA

Objetivo

"Determinar que procesos son ejecutados en un área de negocios especifico así como la Interrelación entre éstos y los datos necesitados" [2].

Características

- Requiere participación intensiva del usuario
- Define un modelado de los datos del sistema
- Define un modelado de los procesos del sistema
- Provoca replantear sistemas y procedimientos
- Crea un modelo detallado para el área de negocios
- Crea modelos para el diseño del cliente y del servidor

Productos

- Modelo de Entidad/Objeto
- Modelo Dinámico
- Modelo Funcional

DESCRIPCION DEL PROBLEMA

El punto de partida para comenzar el análisis de cualquier contexto, es realizando una descripción del problema, para poder comprender los requerimientos del usuario. Esta descripción debe ser clara y concisa, evitando ambigüedades y dudas. Ya que a partir de conocer el problema , podremos comprender los aspectos de la vida real para abstraerlos a los modelos de análisis, y después poder implementar esos modelos a elementos de software y hardware.

En la descripción del problema se incluye la definición de ciertos elementos, tales como :

- Definir los objetivos de la organización
- Alcance del proyecto
- Definir el objetivo del proyecto
- Recopilación de Información

Definir Los Objetivos De La Organización

Se debe de realizar una recapitulación de los objetivos de la organización y del área donde se vaya a aplicar el sistema, esto para poder visualizar hacia donde debemos de canalizar el desarrollo del sistema y que relación debe de existir entre nuestro proyecto y el cumplimiento de esos objetivos.

---

[2] Seminario, Análsis y Diseño de Sistemas, Factum Infomática Mancera, p6

Alcance Del Proyecto

Delimitar el área de acción del proyecto, es una tarea de vital importancia, ya que a partir de la limitación del alcance del proyecto, podremos expresar los objetivos y productos del sistema. En la limitación del proyecto se deben definir los siguientes elementos:

1. Departamentos involucrados en el sistema
2. Procesos y Funciones para cada departamento
3. Las personas o puestos que intervienen en los procesos

Definir El Objetivo Del Sistema

Una vez que conocimos los objetivos de la organización y el alcance del proyecto, debemos definir el objetivo del sistema a desarrollar, en términos de los beneficios que otorgará el sistema en el área en la cual se aplique. Así como la definición de subobjetivos o productos que se obtendrán del desarrollo del sistema.

Recopilación De Información

Para poder determinar los puntos anteriores y además estudiar el sistema actual, debemos de realizar una recolección de la información, por alguno de los siguientes medios:

- Manuales (organización, políticas, procedimientos, etc.)
- Reglamentos
- Procedimientos de procesos
- Documentación generada por los procesos (facturas, notas, memorándums, etc.)
- Entrevistas
- Observación directa

Todas estas fuentes de información son necesarias para la concepción del contexto por parte de los analistas.

Cuando en la tarea de análisis, nos hemos documentado acerca del comportamiento de un escenario, y además conocemos su funcionamiento actual y de los temas de los cuales se tratan. Ahora podemos comenzar a plasmar esa realidad en modelos abstractos. El plasmar en estos modelos la realidad, no implica copiarla tal cual, ya que el analista como agente de cambio, puede identificar posibles mejoras en cuanto a procesos y datos, los cuales pueden ser modelados para el nuevo sistema, pero esto siempre y cuando el dueño del proceso lo acepte, lo cual es una labor de venta por parte del analista para la mejora de ese proceso.

MODELO DE ANALISIS

El modelo de análisis, como lo definimos anteriormente, será en marco de integración de los modelos de entidad-objetos, dinámico y funcional de una realidad, mediante la construcción del Diagrama de Descomposición del sistema, y a partir de este, se descomponen los demás modelos.

Diagrama De Descomposición (DD)

Es una herramienta la cual nos permite descomponer un objeto en sus componentes o eventos, los cuales puedan suceder en el sistema, este diagrama se asemeja a un organigrama, ya que muestra una arquitectura de tipo jerárquica.

El diagrama se diseña a partir del método top-down, hasta llegar al nivel de detalle que se quiera, cada nivel estará formado por módulos, por lo que existen tres tipos de módulos;

Modulo padre: Es aquel que da nombre al diagrama o el nivel mayor del sistema, y del cual se descomponen los demás módulos.

Modulo Intermedio: Es aquel modulo que fue descompuesto de un nivel superior y el cual es susceptible de ser descompuesto.

Módulo terminal: Es el módulo que fue descompuesto de un nivel superior, pero ya no es susceptible de ser descompuesto.

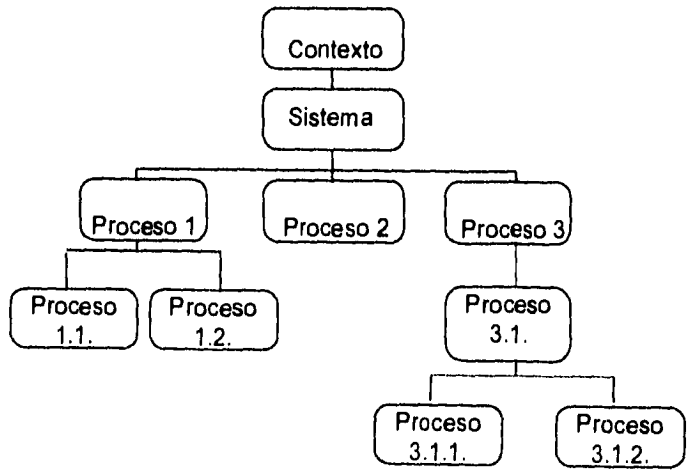


Diagrama de descomposición

En el diagrama anterior podemos ver dos módulos, con el nombre de contexto y sistema, estos son módulos padre y siempre deben dibujarse de esa manera, ya que a partir del modulo contexto se define el diagrama de contexto del sistema y a partir del modulo sistema se desglosan todos los procesos internos

El criterio para descomponer el sistema en un DD, es a partir de cada función que se realiza, como puede ser pedidos, embarques, inventarios, facturación, etc., en donde los subprocesos para esas funciones, serán de alta, baja, cambios y consultas, que son las actividades primordiales de los sistemas de información para las organizaciones.

MODELO DE ENTIDAD/OBJETO

En este modelo, identificamos los elementos que intervienen en el sistema que analizáremos. Una entidad u objeto en el mundo real puede ser una persona, lugar, objeto o concepto, el cual interviene en nuestro contexto. Para cada elemento que identifiquemos en nuestro contexto, debemos definir: características, relaciones y el tipo de datos que estos generan.

Diagrama Entidad-Relación (DER)

Es un modelo de datos, el cual representa las relaciones de información entre los diferentes elementos de un sistema, a través de los siguientes elementos:

Entidad. Es aquel elemento tal como una persona, lugar, objeto, concepto, actividad o evento que genera información de interés a una organización.

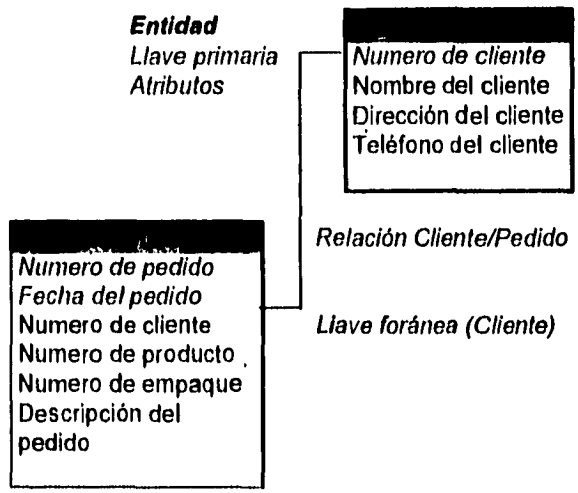
Relación. Es una asociación entre dos entidades

Atributos. Son las propiedades descriptivas de una entidad o relación.

Llave primaria (PK). Es el identificador único que distingue a una instancia de atributos, de una entidad. La llave primaria es un atributo de la entidad.

Llave foránea. Es el atributo de la entidad, que es llave primaria de otra entidad.

Las relaciones entre entidades se llevan a cabo a través de las llaves primarias, foráneas y de atributos, que sean comunes entre ellas.



Con el diagrama entidad relación definimos los datos que intervienen en el contexto del sistema, ya que este se desarrolla con el método bottom-up, es decir para cada nodo terminal del DD definiremos un DER, y continuaremos con los procesos padres de los nodos terminales hasta llegar al nivel superior del DD.

Reglas Para La Construcción De Un Diagrama Entidad-Relación

El DER se desarrolla por medio del método bottom-up, es decir, a partir del DD, identificamos para cada nodo terminal las entidades, relaciones y atributos que intervienen en el proceso, así sucesivamente hasta llegar al nivel superior.

Definición de entidades por cada proceso de DD

Debemos de identificar cada entidad, que intervenga en los procesos a analizar, es decir no colocaremos en el DER de un proceso x, una entidad que no hace referencia a ese contexto, sino solo las que intervengan.

Para poder identificar entidades, a partir de entrevistas, procedimientos, etc., debemos de seguir las siguientes reglas:

- Las entidades siempre son sustantivos, pero no todos los sustantivos son entidades.
- Ignorar los sustantivos los cuales identifican datos específicos.
- Para definir el nombre de una entidad se debe de:
  - Usar un nombre en singular
  - Hacer diccionario de datos, para documentar cada entidad.
  - Eliminar homónimos de entidades

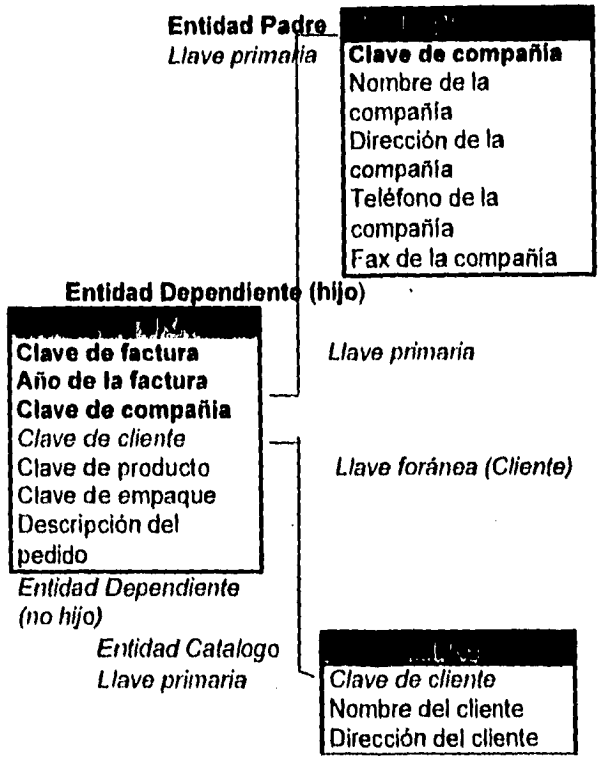
Clasificar cada tipo de entidad

Un sistema tiene elementos, los cuales se relacionan a través de dependencias, por lo cual existen diferentes tipos de entidades, las cuales deben ser plenamente identificadas.

Entidad Padre. Es una entidad independiente, de la cual dependen una o más entidades, a través de su llave primaria.

Entidad Catalogo. Es una entidad independiente, de la cual dependen una o más entidades, pero en menor grado, ya que la dependencia se da a través de un atributo o varios, pero no de la llave primaria.

Entidad dependiente o hijo. Es aquella que depende de una o varias entidades, a través de la llave primaria o de un atributo de la entidad Padre o Catalogo.



Definir entidades a partir de entidades similares.

En ocasiones se puede identificar varias entidades con características en común, con atributos, relaciones, llaves primarias similares. En estos casos se unifican en una sola entidad, y se generalizan los atributos y relaciones, para que sea una entidad representativa a todos lo casos.

Gastos de publicidad Gastos de transportación Gastos de exportación	Agrupamos las entidades en:	**Gastos de Venta**

Caracteristicas de las Llaves primarias de las Entidades

Una llave primaria de las entidades deben de ser :

- Únicas para cada instancia
- No aceptar valores desconocidos
- Debe ser conocida todo el tiempo
- No duplicable

Otras características deseables:

- Corta y simple
- Sea fácil de identificar

Las llaves primarias, siempre son atributos de una entidad, por lo cual pueden existir varios atributos para ser llaves primarias, estas se conocen como *llaves candidatas*. Las llaves primarias pueden definirse en forma compuesta, es decir a través de la concatenación de varias llaves candidatas

Tipos de relaciones

La relación de una entidad a otra define dependencia, pero existen dos grados de dependencia:

- Dependencia directa
- Dependencia indirecta.

La primera es aquella que se denota en la llave primaria, es decir, la dependencia es total. Mientras que la segunda se denota como llave foránea en los atributos de la entidad, ya que solo sirve como catalogo de la entidad dependiente.

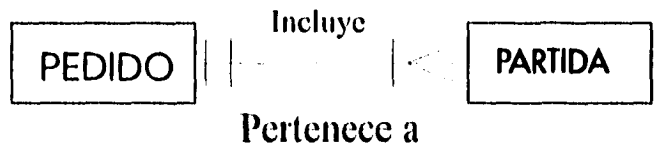
Reglas para definir relaciones

- Las relaciones son verbos, pero no todos los verbos son relaciones
- Ignorar los verbos que describan datos más que procesos
- Busca a través del texto relaciones implícitas
- Usar voz activa cuando sea posible
- Documentar sinónimos en un diccionario de datos y elimina homónimos
- Puede existir relaciones recursivas es decir que una entidad, se relacione con ella misma.
- Eliminar relaciones indirectas, ya que esto es redundancia de información

Cardinalidad

La cardinalidad es el numero de ocurrencias que hay en una relación, entre una y otra entidad, los tipos de cardinalidades que hay son:

||---	Necesariamente existe solo uno	Uno(a) y solo un(a)
|0---	Puede existir máximo uno	Un(a) o Cero
>|---	Necesariamente existe uno o más	Uno(a) o muchos(as)
>0---	Pueden existir uno o más (no muy usada)	Cero o muchos(as)



El diagrama del ejemplo anterior incluye la siguiente información: Las Entidades PEDIDO y PARTIDA y la relación entre ellos que se lee como:

PEDIDO *Incluye* Una o muchas PARTIDAS
PARTIDA *Pertenece a* Uno y solo un PEDIDO

Cabe aclarar que para interpretar la relación se parte de una entidad leyendo su nombre, después la leyenda de la relación (en Dirección de las manecillas del reloj), inmediatamente la cardinalidad y luego el nombre de la otra entidad.

La cardinalidad son identificadas a través de las reglas de negocio o políticas.

Las leyenda de las relaciones deben ser definidas en cuanto a la dependencia entre las entidades, los tipo verbos que debemos de definir son:

- Cuando una entidad depende de una sola entidad se debe de identificar con el verbo **pertenece a** y la cardinalidad será de 1-1 (independiente-dependiente)
- Cuando una entidad depende de más de una entidad, la relación será **asocia a** y la cardinalidad será máxima de 1-1 (independiente-dependiente)
- Cuando una entidad depende de otra entidad, pero en forma de catalogo la relación será **corresponde a**

Mientras que la cardinalidad de regreso será un verbo que identifique otra relación.

Las relaciones entre entidades mucho a uno, se definen como llaves foráneas que aparecen en varios registros por lo que hay redundancia de datos.

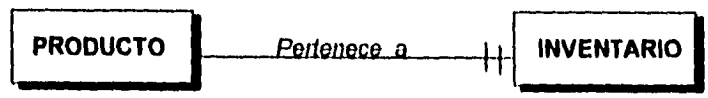
Válido

bdg_clave	prd_clave	emp_clave	inv_precio	inv_peso neto	inv_peso bruto
00001	95	301	523	500	450
00001	96	301	523	500	450
00001	97	301	523	500	450
00001	98	301	523	500	450

No válido

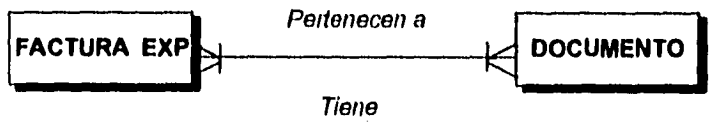
bdg_clave	prd_clave	emp_clave	inv_precio	inv_peso neto	inv_peso bruto
00001	95 96 97 98	301	523	500	450
00001	100	303	537	500	450
00001	102	303	248	500	460

Las relaciones uno a uno, se definen como llaves foráneas que aparecen en un solo registro



bdg_clave	prd_clave	emp_clave	inv_precio	inv_peso neto	inv_peso bruto
00001	**95**	301	523	500	450
00001	**96**	301	523	500	450
00001	**97**	301	523	500	450
00001	**98**	301	523	500	450

Las relaciones muchos a muchos, se convierte en una tabla asociativa.



fexp_clave	fexp_año	cia_clave	prd_clave	emp_clave	fexp_desc
00001	**95**	**301**	523	500	450
00002	**95**	**301**	523	500	450
00003	**95**	**301**	523	500	450

ENTIDAD ASOCIATIVA

fexp_clave	fexp_año	cia_clave	doc_clave	num_originales	num_copias
00001	**95**	**301**	***23***	4	2
00001	**95**	**301**	***24***	2	2
00003	**95**	**301**	***23***	1	2

doc_clave	*doc_desc*	*doc_origen*
23	certificado de exportación	SECOFI
24	certificado de calidad	SECOFI
25	visa consular Argentina	Embajada Argentina

- Una llave foránea aparece dos veces, en la entidad independiente y en la asociativa
- La llave primaria de la entidad asociativa, se compone de la llave primaria de las dos entidades
- Las llave primaria de la entidad asociativa, no debe de aceptar valores nulos y no default

Identificar Atributos De La Entidad

Los atributos por lo general tienen una serie de caracteristicas tales como:

- Nombre
- Tipo físico de dato (carácter, entero, flotante, binario, etc.)
- Numero de caracteres o digitos de ese atributo
- Valores predefinidos
- Reglas de validación
- Un valor por default

Los atributos también tienen cardinalidades, un atributo puede tener un numero mínimo o máximo de valores. Entre el numero de cardinalidades que puede tener son: cero, uno, muchos.

Para definir las cardinalidades existen una serie de términos los cuales son:

- Si un atributo es singular su cardinalidad es uno
- Si un atributo es plural su cardinalidad es de muchos
- Si un atributo es opcional, su cardinalidad es cero
- Si un atributo es requerido su cardinalidad es uno

Tipos de atributos

Derivados o calculados. Son aquellos que adquieren su valor a partir de una computación
Primitivos. Son aquellos que son definidos directamente.

Reglas Para Definir Atributos

- El nombre de un atributo deben ser un nombre mnemónico, con el prefijo del nombre de la entidad, de tres o cuatro letras, seguido de un guión bajo y por ultimo el nombre del atributo.

  Atributos de la entidad **Compañía**; *cia_clave, cia_nombre, cia_direccion,* etc

- Los atributos que sean llaves foráneas de una entidad, conservaran su nombre de origen.

  La entidad **Factura,** tiene una relación con la entidad **Producto,** a través del atributo *prd_cve_producto* el cual conserva su mismo nombre de origen en la entidad **Factura.**

- Los tipos de datos de un atributo se definen previamente
- Eliminar homónimos y documentar sinónimos
- Los atributos con cardinalidad mínima de 1 su valor no será nulo
- Las llaves primarias y foráneas no tendrán valor nulo ni por default
- Los atributos no requerido podrán tener un valor nulo

- Los atributos con cardinalidad M a M y 1 a M se convierten en entidades hijas de la entidad.

cte_cve (1-1)	cte_nombre(1-1)	cte_direccion(1-1)	cte_tel (M-M)	cte_rfc(1-1)
1001	CELANESE, S.A.	Av Revolución	3-34-78-90 3-90-67-89 4-89-12-34	CEL099876

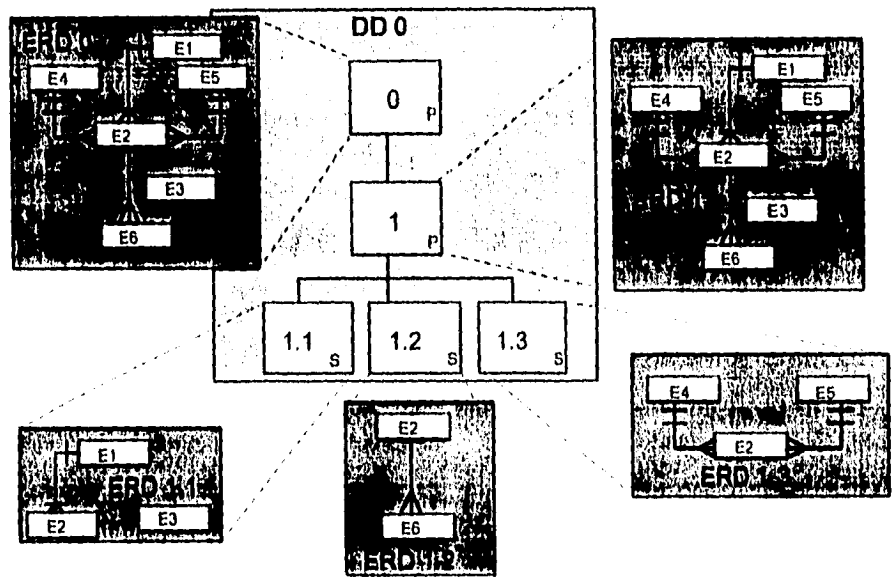
cte_cve (1-1)	cte_nombre(1-1)	cte_direccion(1-1)	cte_rfc(1-1)
1001	CELANESE, S.A.	Av Revolución	CEL099876

cte_cve	te_telefono
1001	3-34-78-90
1001	3-90-67-89
1001	4-89-12-34

La cardinalidad de los atributos de las entidades debe ser 1 a 1 o 0 a 1, después de terminar por completo el modelo de datos.

Como se dijo en un principio el DER se desarrolla a partir del DD, ya que para cada modulo del diagrama, se desprenderá un DER, utilizando el método botton-up, es decir desarrollando para cada nodo terminal su DER, hasta llegar a formar el DER de todo el sistema.

El siguiente dibujo muestra los DER de cada uno de los procesos.



Es importante remarcar como los tres procesos terminales (1.1,1.2 y 1.3) tiene diferentes entidades y relaciones. Pero el proceso (1) se compone de todas las entidades y relaciones que se presentaron en los procesos 1.1, 1.2, y 1.3.

El enfoque de construcción del DER, es bajo el diseño de un modelo de datos normalizado, es decir, cada una de las reglas definidas anteriormente, garantizan la normalización del DER, hasta la tercera forma normal.

Normalización

Es la técnica por medio de la cual, se simplifica la estructura lógica de los datos, eliminando la redundancia de datos en tablas y columnas, determina claves únicas necesarias para el acceso a los elementos de datos. La normalización pone a las bases de datos en una consistencia lógica con un mínimo de redundancia.

Los niveles de normalizaciòn para las tablas son :

Primera forma normal.

Regla: Cada celda de una tabla contiene un solo valor, y no puede contener varios valores. Por definición todas las tablas están en la primera forma normal.

Segunda forma normal.

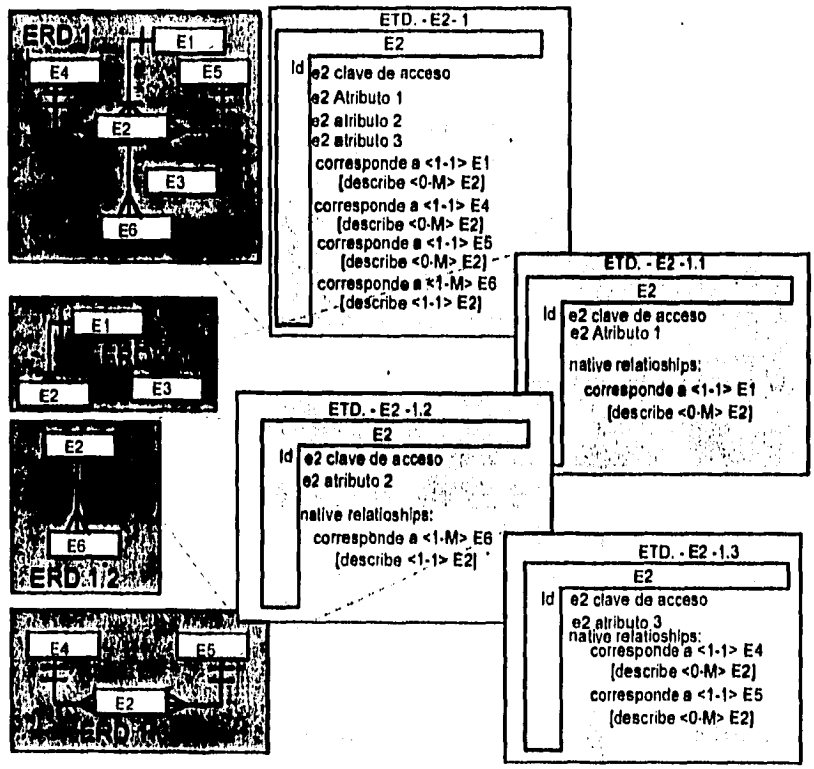
Regla: La tabla que esta en la primera forma normal y cada columna no llave depende de la llave primaria entera.

Tercera forma normal

Regla: La tabla que esta en la segunda forma normal y un campo no llave no debe depender de otro campo no llave.

Diagrama De Descripción De Entidades (DDE)

Este diagrama define las características o atributos de una entidad (solo las relevantes a su contexto) . En un DDE se listarán primero los atributos (peso, precio etc.), e inmediatamente las relaciones de la entidad con otras entidades. (pertenece a < 1- 1> COMPAÑÍA)
En todo DDE se encontrará un atributo con la marca de **'Id'** que indica que este atributo será usado como identificador de los elementos de la Entidad, este identificador es un requisito para cada entidad.
Por ultimo en el siguiente dibujo se presentan de los DER del dibujo anterior. Desprendiéndose el DDE de la Entidad **E2.**



Dominios De Datos

Una vez que determinamos los atributos del DDE, debemos de asignar a cada uno de ellos el dominio de datos, el cual identifica el dominio de valores que soporta. La forma de hacer es la siguiente:

1. Generar los tipos de datos.

La generación de los tipos de datos, es a través del análisis del contenido de la información de nuestro sistema, ya que existen diferentes tipos de valores, ya sean fechas, nombres, números enteros, cantidad de dinero, números decimales, etc. Por lo que debemos definir tipos de datos genéricos y específicos para los tipos de datos.

- *Identificar los tipos de datos genéricos.* En un sistema de información, existen nombres, descripciones, fechas ,etc., por lo que se definirá un tipo de dato que indique la naturaleza del atributo, anteponiendo el prefijo *typ* , para identificar que es un tipo de dato, después un guión bajo y la naturaleza de la información, por ejemplo para los nombres, la palabra *nombre* ,el tipo de dato quedaria como *typ_nombre* , es decir un tipo de dato que contiene un nombre. El cual se le asignará a todos los atributos que se refieran a nombres.

- *Identificar los tipos de datos específicos.* Pueden existir atributos, los cuales el tipo de dato sea especifico o único, esto por la importancia o formato de información que requiera. Por ejemplo el atributo de clave del producto, el cual debe tener un dominio de información especifico, por lo que el nombre del tipo de dato será *typ_prd_cve* , en donde *typ* es para identificar el tipo de dato, *prd* es el mnemónico de la tabla producto, y *cve* el atributo clave del producto.

2. Definir para cada tipo de dato su dominio de información.

Cada tipo de datos, deberá tener asociado un dominio de valores, el cual puede ser:

- **Definir para cada tipo de dato.** El tipo de valores que soporte, si el tipo *typ_nombre*, es un char de 20, es decir el tipo de dato *typ_nombre*, soporta solo 20 caracteres alfanuméricos. El tipo de información que definamos para los tipos de datos, deberá ser soportado por el manejador de base de datos que se vaya a utilizar. Esta regla es necesaria para completar el análisis.

- **Definir un rango de valores.** Cuando un tipo de dato se le asigna un valor de entero y numero decimal, a estos, se les puede definir un rango de valores. Por ejemplo el tipo de datos fact_fec, que tiene el tipo de dato typ_fecha que hace referencia a fechas, tiene un rango de 1/1/1990 al 31/12/1996, y todos los valores de ese atributo deberán ser entre esos valores.

- **Definir un dominio de información previo.** Esta sección es optativa, ya que un valor puede tener asignado una lista de valores, que solo debe tomar el atributo al cual le asignemos el tipo de dato. Por ejemplo el atributo que tenga el tipo de dato, *typ_pais_nombre*, podrá tener el valor de México, USA, Venezuela y Japón.

- **Definir valores por default.** Un atributo puede tener un dato, el cual cuando en una operación de inserción o actualización, el valor de ese atributo sea un valor previamente definido. Por ejemplo el atributo que tenga el tipo de dato, *typs_pais_nombre*, podrá tener el valor por default de México.

3. Asociar a cada atributo, un tipo de dato.

Una vez que definimos los tipos de datos, el siguiente paso es asignar a cada atributo, el tipo de dato correspondiente.

fact_numero	typ_numero	entero
fact_año	typ_año	entero entre 90 y 97
cte_cve	typ_cte_cve	entero a partir de 100
prd_cve	typ_prd_cve	carácter de 3
fact_desproducto	typ_dsc	texto
fact_importe	typ_cantdin	dinero
fact_iva	typ_cantdin	dinero
fact_imptotal	typ_cantdin	dinero

ESTA TESIS NO DEBE
SALIR DE LA BIBLIOTECA

MODELO DINAMICO

El modelo dinámico nos plasma dos elementos básicos: eventos y estados. Un objeto puede tener diferentes status, los cuales cambian debido a que un evento, afecta al objeto.

Estados De Un Objeto.

Un objeto puede tener o estar en varios estados, por ejemplo una factura puede estar:

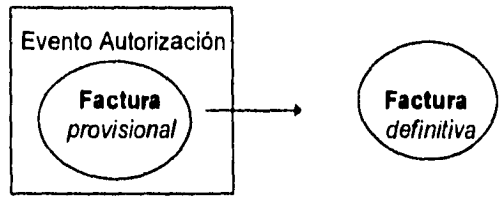
Factura Provisional
Factura Definitiva
Factura Surtida
Factura Cancelada

Los objetos tienen un ciclo vital en sus estados, algunos de estos estados pueden seguir diferentes caminos, es decir una factura provisional, puede pasar al estado de Definitiva o Cancelada, pero nunca pasa del estado *Definitiva* a *Cancelada*.
Al implantarlo en un lenguaje de programación OO, el estado se registra en los datos almacenados con relación al objeto. Se determina mediante las clases y valores de los campos asociados con el objeto.
En los lenguajes OO, las solicitudes se envían y provocan la activación de los métodos o procedimientos. Estos métodos cambian el estado del objeto. El estado se registra en los datos del objeto.

Eventos

El mundo funciona bajo la activación de eventos. Para el diseño orientado a objetos el mundo se describe en términos de los objetos y sus estados, así como de los eventos que modifican esos estados.

El objeto factura que tiene un estado de provisional, cuando a este es afectado por el evento autorización, el objeto factura cambia su estado a definitiva.

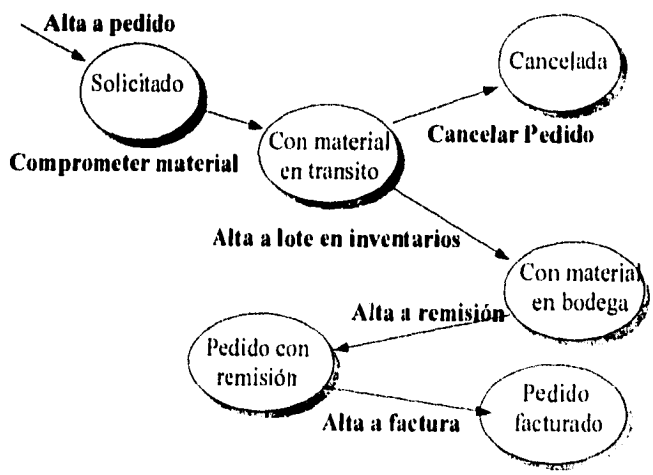


Sin los eventos, el mundo no cambiaría, y en un mundo sin eventos podríamos construir y generalizar bases de datos sin preocuparnos por actualizarlas. Sin embargo, en las aplicaciones debemos de conocer los eventos posibles que producen los eventos en la actualización, borrado e inserción de registros.

Un evento puede provocar la reacción en cadena de uno o varios eventos, por lo cual podemos tener una red de eventos, que son provocados unos por otros (causa-efecto). Por ejemplo en un procesador de palabras, el objeto documento con el estado *sin salvar*, se le aplica el *evento salir*, el cual provoca el evento de *preguntar si se desea guardar ese documento*, y si el objeto hubiera tenido el estado *salvado*, el evento salir, hubiera provocado la salida del procesador de palabras y no preguntar si se desea salvar el documento. Los eventos modifican el estado de los objetos o entidades por medio de la aplicación de un proceso, pero antes de aplicarse el evento se realiza una evaluación de cierta condición, que es el estado del objeto.
Con lo anterior podemos definir, que un sistema cuenta con una serie de eventos, que suceden en el tiempo dependiendo del estado del objeto o entidad al que haga referencia y a la vez estos son transformados por los eventos. Y que existe una cadena de eventos desatados unos por otros para la consecución de un proceso o acción. Por esto es necesario contar con las herramientas necesarias, para lograr modelar los estados de las entidades u objetos y sus eventos de afectación en el tiempo.

Diagrama De Transición De Estados (DTE)

Los objetos de un sistema, tienen un ciclo vital, y la sucesión de eventos pueden modificar su estado. Para poder identificar cuales son esos estados y los eventos que incurren en los mismos, tenemos una herramienta conocida como diagrama de transición de estados, el cual hace referencia a ciertos objeto y sus posibles estados, se representan por medio de círculos, que son los estados de un objeto, mientras que las flechas, representan los eventos o procesos que transforman el objeto.



El evento es la realización de un proceso aplicado a un objeto, ya que el evento de componer un auto, implica aplicar el proceso de compostura del carburador, al objeto auto, que cambia el estado de descompuesto a compuesto.

Los objetos de interés para nuestro sistema, son aquellos elementos que producen información de importancia a nuestro contexto.

Los criterios para determinar los estados de cada entidad u objeto, se definen bajo las reglas del negocio o políticas de la empresa, ya que si un pedido esta con material en transito, es el único estado en el cual puede ser cancelado, pero bien puede ser cancelado cuando el pedido esta con material en bodega. Ese tipo de criterios son definidos por la organización.

Siempre un diagrama de este tipo tendrá un estado inicial y un final, el cual marca un ciclo de vida de esa entidad u objeto. Por lo tanto con este tipo de diagramas podemos modelar el comportamiento de los elementos de un sistema, en forma individual, como es para una factura, remisión, etc. Pero en la realidad cada uno de los elementos del sistema interactuan entre si, por lo cual debemos de modelar esas relaciones entre elementos, ya que el estado de un objeto puede provocar el cambio de estado de otro objeto, como consecuencia de un evento.

Diagrama De Esquema De Eventos (DEE)

En el diagrama anterior logramos definir un medio para modelar los estados y eventos de un objeto en forma individual, pero estos objetos dentro de un sistema no son únicos, ya que interactua junto con otros, por lo cual debemos de modelar esa interacción entre objetos.
El diagrama de esquemas de eventos nos ayudan a visualizar la relación entre objetos y la sucesión de eventos en cada función del sistema a automatizar.

Con este diagrama podemos modelar las cadenas de eventos (causa-efecto), los cambios de estados de los objetos, los procesos que interactuan como producto del evento y los datos o información que interviene en ese proceso, también como consecuencia de esos eventos.

La nomeclatura que interviene en los diagramas de orientación de eventos es:

Representa llamados de un objeto externo.

Representa un objeto

Representa una condición

Representa el proceso

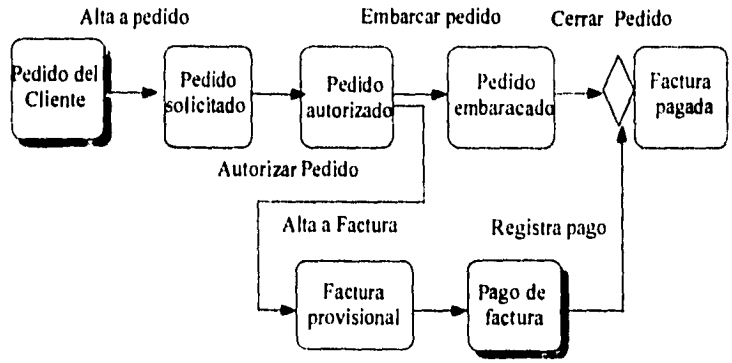
Representa un evento externo, provocado por un reloj

Los eventos provocan cambios de estados a los que un sistema debe conocer y reaccionar ante ellos de algún modo. Muchas operaciones se producen desde el exterior, como consecuencia de un evento de nivel superior o del mismo nivel o en su caso como un estímulo por parte del usuario, como el presionar un botón o un menú.

"Cuando ocurre un evento, lo usual es que el cambio de estado active una serie de operaciones dentro de un objeto, las cuales operen bajo un contexto de datos. Por esto las reglas de activación definen una relación de causa-efecto, ya que siempre que ocurra un evento de cierto tipo, la regla de activación invoca a una operación bien definida. Un tipo de evento puede tener varias reglas de activación, cada una de las cuales tiene su propia operación dentro de un objeto, las cuales se activan en paralelo"[3]. Las operaciones paralelas pueden producir diferentes cambios de estados en forma simultánea, es por esto que debemos de procurar una lógica de eventos, para que no caigamos en circulo viciosos, en donde un evento cambie el estado de un objeto y otro lo regrese a su estado inicial.

Un evento puede ser invocado desde varios lugares, una operación puede ser invocada por una o varias reglas de activación, pero estas reglas de activación pueden verificar una condición de control, en la cual se
evalúe una condición o un estado del objeto, en donde si la condición es verdadera se causa un evento A, y si no lo es, no causará el evento A, sino el evento B.
Las condiciones de control también pueden actuar como puntos de sincronización, para el procesamiento en paralelo.

Esquema de Eventos Pedido-Factura

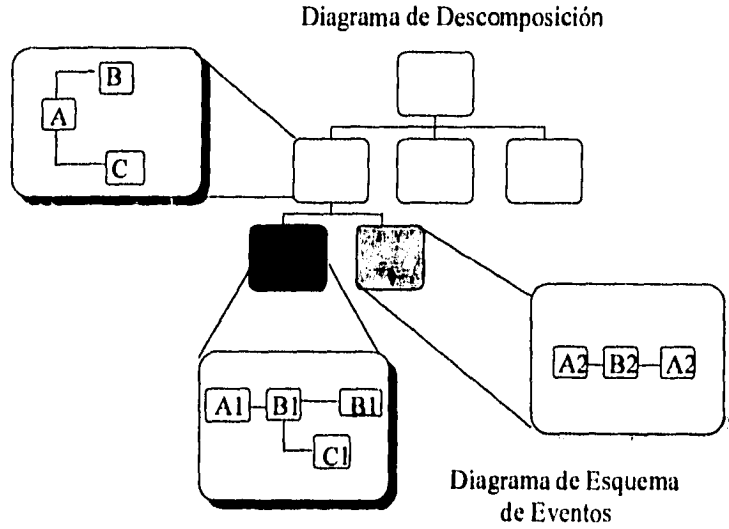


Cada operación lleva a cabo su tarea sin importar lo que ocurra en otra parte. Una operación es invocada por varios mecanismos de activación, ejecutando un método y se espera que este modifique el estado de un objeto, este evento tendrá un cierto efecto, dependiendo de la causa que lo origino, por lo cual se reconoce una causa efecto.

[3] Análisis y Diseño Orientado a Objetos,James Martin, p105

En el diseño del comportamiento de los objetos, trazamos diagramas para mostrar la interacción dinámica de los eventos. Estos diagramas se conocen como esquemas. Un esquema de objetos expresa el tipo de objetos y sus asociaciones en un sistema dado. Un esquema de eventos expresa un guión de procesamiento que cambia los estados de objetos, puesto que los eventos se expresan al cambiar el estado de determinado tipos de objetos.

El proceso para construir el diagrama de esquema de eventos, es a partir del diagrama de descomposición, ya desarrollaremos para cada función del sistema, un DEE en donde intervendrán los objetos de ese contexto y las transformaciones que sufre.



A partir del DD de análisis, desde los procesos terminales, definimos los objetos que intervienen en ese contexto, realizamos el DEE, así para cada uno de estos. El siguiente paso es hacer la composición ascendente de los objetos y estados a los siguientes niveles del DD, hasta llegar al nivel superior.

MODELO FUNCIONAL

En el modelo funcional debemos identificar los flujos de datos, que corren en un sistema y son transformados por procesos. Por lo cual contamos con dos elementos para definir este modelo; el diagrama de flujo de datos el cual en forma gráfica desde el nivel superior (diagrama de contexto) del sistema, podemos ver como los flujos de entrada al sistema se introducen a este y son transformados por los elementos internos del sistema, hasta el nivel de detalle deseado, y posteriormente suben hasta el nivel superior para generar una salida de ese sistema. Mientras que por medio del texto estructurado, se describen esos flujos y transformaciones en forma narrativa y textual.

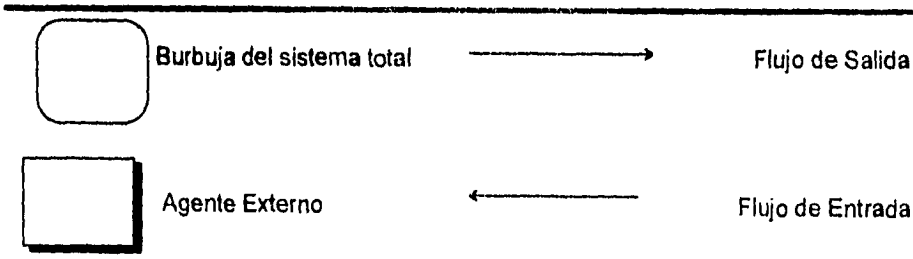
Diagrama De Contexto (DC)

En el análisis, la labor más difícil en la especificación de un sistema es determinar qué es parte del sistema y qué no. Ya que en cualquier sistema que se desarrolle, no importa lo ambicioso ni lo grandioso, será parte de un sistema mayor.

Por lo que el primer paso para definir el modelo, es el desarrollar un análisis para identificar la frontera entre el sistema y el resto del universo, así como definir qué ésta en el interior del mismo, con esto podremos saber que información entra al sistema, del exterior, y qué información se produce como salida hacia el mismo.

Las entradas y salidas no se producen al azar: ningún sistema de información toma todos los datos disponibles en el universo, ni expulsa cosas al azar en el medio ambiente. Los sistemas que son desarrollados son racionales y tienen un propósito; específicamente, producen una salida como respuesta a un estímulo del ambiente. Por lo que hay que identificar que procesos externos influyen como estímulo al sistema y que procesos internos se dispararan para captar ese estímulo y por ende que respuesta será la salida del sistema.

El diagrama de contexto, es un modelo gráfico el cual representa la totalidad de un sistema, a través de una burbuja. En la cual se reflejan las entradas y salidas de información al sistema, por lo que debemos de tener bien identificada, las fronteras del sistema. Ya que la información fluye entre el sistema y agentes externos.



Cada uno de los elementos anteriores, deben tener un nombre corto para poder identificarlo.

Elementos de un diagrama de contexto

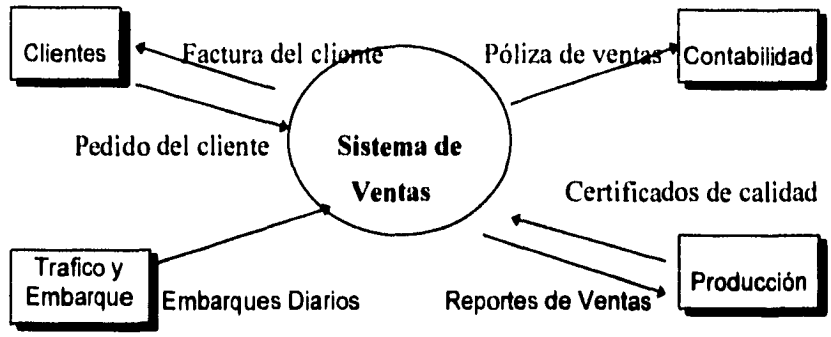


Diagrama de Contexto

Diagrama De Flujo De Datos (DFD)

Es un diagrama que describe el flujo de información a través de los procesos de un sistema. Un DFD nos ayuda a modelar los procesos y la información en un sistema, los cuales son susceptibles de descomposición hasta el nivel que se quiera, utilizando al igual que el DD, el método Top-Down.

Entre las características de un DFD están:

- La notación es clara y sencilla
- Se basan en metodología como Gane/Sarson y Yourdon/Demarco
- Identifica a los elementos que producen información
- Define el flujo a seguir de información

Los componentes de un DFD son:

- Procesos
- Flujos
- Almacén de datos
- Flujo de entrada o salida
- Conjunciones de flujo de datos

Proceso

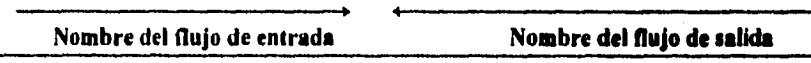
Es sinónimo de burbuja, función o transformación. Es una actividad que transforma entradas en salidas.
Este es representado por un circulo en la metodología DeMarco-Yourdon y como un rectángulo redondeado en Gane-Sarson.

Dentro de la figura lleva el nombre del proceso.

Proceso

Flujo

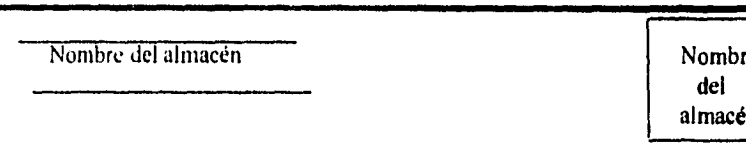
Es representado por una flecha, son las entradas y salidas de información que fluyen entre los procesos o eventos, todos los flujos deben de tener un nombre, representativo de la información que fluye y solo tendrá un solo sentido, entrada o salida.



Flujo

Almacén o almacén de datos.

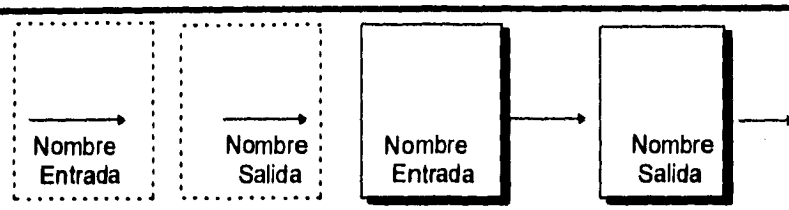
Es utilizado para modelar cualquier tipo de colección de datos en reposo, del cual, entran o sale información.
Este es representado por dos líneas paralelas en DeMarco-Yourdon y como un cuadro sin el lado derecho en
Gane-Sarson.



Almacén

Flujo de entrada o de salida

Este elemento representa la entrada o salida de información al diagrama, ya sea de un nivel superior o de un modulo del mismo nivel.
Es representado por un rectángulo con línea punteada en Gane-Sarson y como un cuadro con sobra en Yourdon-DeMarco.



Flujo Entrada/Salida

Conjunción de flujos de datos

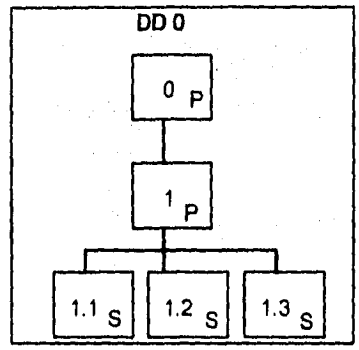
Representa un punto en el cuál los flujos de datos se unen en uno solo, o un punto en el cual un flujo de datos, se dispersa en varios. Y es representado con un punto:

Conjunción

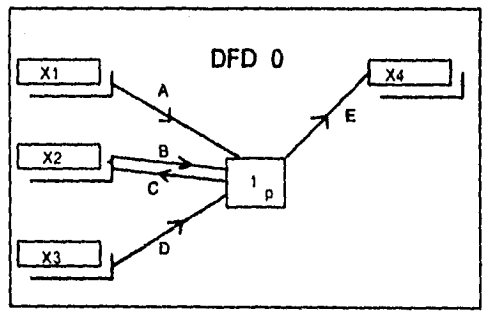
La construcción de un DFD parte del diagrama de descomposición, ya que cada proceso que tenga una serie de procesos secuenciales colgados, será susceptible de un DFD, sólo los nodos terminales o secuenciales no tendrán asociados un DFD.

Un DFD ayuda a representar un sistema a cualquier nivel de abstracción , ya que un DFD es descompuesto por niveles por lo que guarda una estrecha relación con los diagramas de descomposición, ya que el primer nivel de un DFD es el nivel cero o de contexto, que es lo que conocemos como Diagrama de Contexto, el segundo nivel es el uno, en donde se plasman todos los procesos que interactuan en el sistema, y así sucesivamente hasta el grado de descomposición que se quiera. A esto se le conoce como explosión de sistemas. Así mismo las entradas y salidas se descomponen, las entradas en forma descendente y las salidas en sentido ascendente.

Ejemplo:
Tenemos un Sistema compuesto por tres actividades. Su DD se muestra a continuación

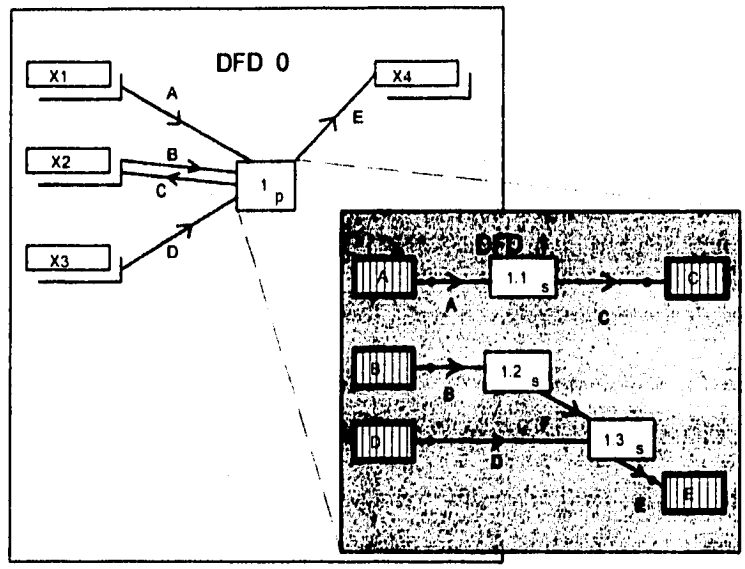


El DFD del proceso **0** se vería como:



El DFD del proceso **0** hace la función de Diagrama de Contexto, puesto que nos permitirá ver con que *agentes externos* (x1,x2,..x4) tiene intercambios nuestro sistema y cuales son los flujos de datos con estos. (A,B,C,E,D).

La siguiente figura contiene el DFD de los Proceso **0** y **1** mostrando con los flujos de datos (del **0**) se convierten en *conjunciones de contexto* en el siguiente (**1**)



Nótese que el DFD 1 se incluyeron también tres procesos **1.1, 1.2** y **1.3** que son precisamente los proceso en que se descompone el proceso **1.**

Una vez que desarrollamos el diagrama de descomposición y el de flujo de datos, hemos modelados los procesos que intervienen en el contexto del sistema que se esta analizando, el siguiente paso es definir los datos que intervienen en esos procesos que hemos encontrado, ya que siempre un proceso tendrá datos que transformar y producir una salida.

Texto Estructurado

El texto estructurado es una descripción narrativa del sistema modelado, en forma estructurada y secuencial, la cual debe de reflejar procesos principales, flujos, entidades y atributos. Esta se realiza en forma paralela al modelo gráfico del sistema. El Texto Estructurado se describen los mismos procesos o funciones que el DD, describe los flujos del DFD y contiene las entidades del DER.

La elaboración de la lista de eventos, se debe de hacer por medio de una hoja de calculo o procesador de textos de la siguiente manera:

Lista de eventos.

Generar una lista numerada de los módulos o funciones del DD, con un font definido con estilo de **negritas**.

1. **Embarques de Planta a Bodega (Planta)**
2. **Llegada de Embarques en Bodega (Bodega)**
3. **Pedidos Nacional. de bodega (Ofna.Vtas)**
4. **Remisiones de Bodega (Bodega)**
5. **Solicitud de Embarques de Planta Para Bodega (Ofna.Vtas)**
6. **Facturación de Remisiones de Bodega en Oficinas de Venta (ADN)**
7. **Facturación de Pedidos para Bodega en Bodega (Propuesto al Elab. Remisiones en Bodega).**
8. **Lotes en Bodega para Registro de material dañado en Bodega**
9. **Elaboración de Pedido de Material fuera de especificación de la Bodega**
10. **Elaboración de Remisiones para Pedidos de Material fuera de Especificación en Bodega.**
11. **Facturación de una Remisión de Material Fuera de Especificación.**
12. **Elaboración de Pedidos de Nal. de Planta a Cliente (ofna Vtas.)**
13. **Embarques de Planta a Cliente (Planta)**
14. **Facturación de Pedidos de Planta a Cliente**
15. **Notas de Cargo**
16. **Notas de Crédito**
17. **Clasificaciones de Productos.**

Listar actividades de cada evento.

Para cada evento o modulo debemos de listar las actividades que lo comprenden, así como en el DD, numerándolas, a partir del numero del evento. La descripción de esta actividades será a nivel narrativo y con font de estilo normal.

1.- Embarques de Planta a Bodega (Planta)

1.1 Altas Inf.Diario Emb. Planta a Bodega (IDE)

1.1.1 Indica el ***Informe Diario de Embarque*** a Complementar:

- *Compañía*
- *Planta.*
- *Fecha del Informe*

1.1.2 Despliega ***Embarque en tránsito*** registrados.

1.1.3 Despliega ***Inventario Lote Bodega*** (Embarques Recibidos en Bodega).

1.1.4 Despliega (Embarques y Recibidos Cerrados) ***Histórico del Embarque.***

1.1.5 Registra el resto de los datos del Nvo embarque:

- *Clave del Producto*
- *Clave del Empaque*
- *Clave de la Bodega destino*
- *Placa de Camión o Núm. de Carro*
- *Peso Neto Embarcado*
- *Peso Bruto Embarcado*
- *Cantidad de Empaques Embarcados.*
- *Clave del Lote*
- *Fecha de Fabricación*
- *Tipo de Equipo*
- *Fecha Estimada de Arribo a Bodega*

1.1.6 Carga en ***Embarque en tránsito*** el Nvo embarque:

- *Compañía*
- *Planta.*
- *Fecha del Informe*
- *Clave del Producto*
- *Clave del Empaque*
- *Clave de la Bodega destino*
- *Placa de Camión o Núm. de Carro*
- *Peso Neto Embarcado*
- *Peso Bruto Embarcado*
- *Cantidad de Empaques Embarcados.*
- *Clave del Lote*
- *Fecha de Fabricación*
- *Tipo de Equipo*
- *Fecha Estimada de Arribo a Bodega*

1.2 Cambios Inf.Diario Emb. Planta a Bodega
1.2.1 Indica el ***Informe Diario de Embarque*** a modificar:
- *Compañía*
- *Planta.*
- *Fecha del Informe*

1.2.2 Despliega los ***Embarque en tránsito*** registrados. [Si tu Quieres]
1.2.2.1 Despliega *Cantidades Embarcadas.*
1.2.2.2 Despliega *Cantidades Comprometidas* vía Pedidos de ***Embarque Bodega Comprometido.***
1.2.3 Despliega cada uno de los embarques Recibidos (Abiertos y Cerrados) [Si tu Quieres]
1.2.3.1 Despliega de ***Histórico del Embarque***, Las Cantidades Embarcadas, Recibidas,
Recibidas Fuera de especificación y Fuera de especificación en Almacén.
1.2.3.2 Despliega de ***Inventario Lote Bodega***, la Existencia
1.2.3.3 Despliega de ***Lote Bodega Comprometido***, la Existencia comprometidas y a que pedido abastecerán
1.2.3.4 Despliega de ***Histórico Lote Bodega Entregado***, el material entregado y a que pedido abastecieron

1.2.4 Selecciona el Embarque a Modificar. (Sólo de los que están en Tránsito)
- *Clave del Producto*
- *Clave del Empaque*
- *Clave de la Bodega destino*
- *Placa de Camión o Núm. de Carro*

1.2.5 Modifica embarque en memoria.
- *Peso Neto Embarcado*
- *Peso Bruto Embarcado*
- *Cantidad de Empaques Embarcados.*
- *Clave del Lote*
- *Fecha de Fabricación*
- *Tipo de Equipo*
- *Fecha Estimada de Arribo a Bodega*

1.2.6 Diagnostica viabilidad cambio al Embarque
1.2.7 Cambia IDE en ***Embarque en tránsito***

1.3. **Bajas a Embarques**
1.4. **Consultas a Embarques**

En la lista anterior se puede observar que el los puntos 1.1, 1.2, 1.3 y 1.4. se describen las funciones *de altas, cambios, bajas y consultas de la función principal*, que son los Embarques de Planta a Bodega.

Definir elementos de cada evento.

En la descripción de cada evento, es necesario identificar ciertos elementos, tales como:

- Flujos de información (DFD)
- Entidades o elementos que generan información (DER)
- Atributos o caracteristicas de una entidad (DDE)
- Agentes internos que realizan una función
- Agentes externos que realizan una función

Flujos de información. Es la información que fluye a través del sistema, y sufre transformaciones, en cada eventos, se debe de indicar con un font de estilo *itálico* para identificarlo como flujo. Los flujos deben ser los mismos que hay en el DFD, para cada proceso o evento.

3 Elaboración de carta de instrucción CEQ (Carta de Envio de Quimicos)

3.1 Altas a Carta de instrucción CEQ
3.1.1 Selecciona *Compañía,Bodega,Producto,Empaque*
3.1.2 Registra infamación de la CEQ

Entidades. Es cualquier elemento que nos proporcione información, de interés para el sistema. Estos debemos identificarlos a través de la font con estilo ***itálico y negritas.*** Y deben se las mismas del DER para cada proceso del DD.

2.1.3.1 Registra en ***Inventario lote Bodega*** el embarque:

Atributos. Son las características de una entidad, también se define como la información que genera la entidad, estas se identifican a través de un font con estilo *itálico*, en una lista, abajo de un evento que haga referencia a la entidad . Un flujo de información y un atributo es información, pero su diferencia radica en la forma que se genera, ya que un flujo se define en un renglón como parte del evento, mientras que un atributo se define en una lista, como parte de una entidad. Los atributos deben ser los mismos listados en el DDE de cada entidad.

2.1.3.1 Registra en ***Inventario lote Bodega*** el embarque:

- *Compañía* *
- *Planta* &
- *Fecha del Informe (la del embarque desde la planta)*
- *Clave del Producto*
- *Clave del Empaque*
- *Clave de la Bodega destino*
- *Placa de Camión o Núm. de Carro*
- *Peso Neto Existencia* (el Recibido)
- *Peso Bruto Existencia* (el Recibido)
- *Cantidad de Empaque Existencia* (el Recibido)

* Pertenecen a la entidad Compañía
& Pertenece a la entidad Planta

Los atributos subrayados, corresponden a los identificadores únicos de esa entidad, y los atributos que al final tienen un * , & u otro signo, indican que son atributos de otra entidad, la cual se debió haber descrito antes.

Agentes internos. Son aquellas personas, puestos o unidades de una organización, que intervienen en el desarrollo de un proceso interno en el sistema. Estos se indican con font de estilo **normal y negrita** y entre paréntesis, al final de la actividad o conjunto de estas, que se describan.

3. Elaboración de Pedidos Nal. de bodega (Ofna.Vtas)

3.1	Indica quién elabora el Pedido: *Gerente de Venta* y *Año del Pedido* ·
3.2	Indica a que *Compañía* se le colocará el pedido
3.3	Selecciona el ***CLIENTE***

Agentes externos. Son aquellas personas, puestos o unidades de una organización, que intervienen en el desarrollo de un proceso externo al sistema. Estos se indican con font de estilo **normal y negrita** entre paréntesis, al final de la actividad o actividades que este realiza. Para identificar que es externo, las actividades que desarrolla irán subrayadas en su numeración y con esto definimos que todo lo que se encuentre en esa línea es externo: flujos de información, entidades y atributos

4.1.	Registra la infamación de Embarque en la CEQ **(Tráfico y Embarques)**
4.1.1	Registra *Pesos bruto, Peso Neto* y *Cantidad de Sacos* embarcados (reales)
4.1.2	Registra *Descripción del Transporte* y *Estimated Time Arrival*
4.1.3	Actualiza *Agencia Aduanal* , *Frontera de Salida* y *Frontera Destino*

La descripción de la lista de eventos, debe ser con un lenguaje fácil y sencillo de comprender, de tal forma que el usuario o cualquier persona pueda entender el funcionamiento de los eventos. Y asl esta lista sea el medio por el cual los usuarios valide el modelo del sistema, que se realizo a través de los diagramas.

La descripción del contexto de un sistema en los tres modelos anteriores, nos proporcionan los elementos suficientes para poder comprender las necesidades de datos y procesos de los clientes o usuarios. Por lo que ahora podemos diseñar esos modelos de análisis a elementos de software y hardware, tanto para el cliente como para el servidor. La descripción del sistema analizado en el texto estructurado, nos da un elemento de validación del análisis por parte del usuario, ya que una vez terminado, se presenta a este para que sea revisado y validado, y a partir del cual se puedan generar las modificaciones pertinentes y reflejarlas en los modelos construidos.

RESUMEN

El proceso para realizar un análisis para un sistema cliente/servidor, orientado a un back-end de tipo relacional y a un front-end orientado a objetos, se define a través del desarrollo de tres modelos, que conceptualizan la realidad del contexto que nos interesa. Estos modelos son; modelo entidad/objeto, modelo dinámico y modelo funcional, los cuales muestran los elementos, datos y procesos que interactuan en un sistema. Los modelos son representados a través de diagramas; modelo entidad/objeto: diagrama entidad/relación y diagrama de descripción de entidades, modelo dinámico: diagrama de transición de estados, diagrama de esquema de eventos y modelos funcional: diagrama de flujo de datos y texto estructurado.

El enfoque que tiene este análisis, nos da pauta al diseño de un servidor relacional a través del modelo entidad/objetos y del cliente por medio del modelo dinámico, y los procesos tanto del cliente y del servidor con el modelo funcional.

La forma en la cual se conceptualiza el sistema, es en la descomposición de los procesos principales del sistema, en las funciones de *alta, baja, cambios y consultas* de cada función del sistema.

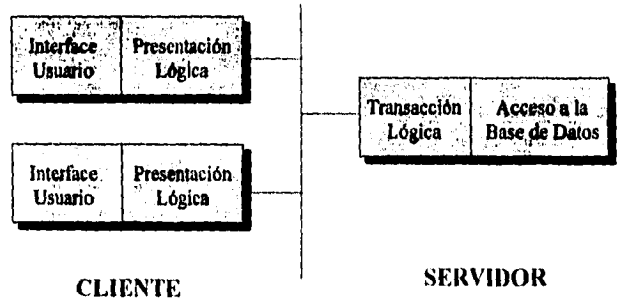
CAPITULO 5
DISEÑO DE SISTEMAS

Cuando un arquitecto y un ingeniero civil se dan a la tarea de construir un edificio, estos antes de comenzar la construcción del inmueble, desarrollan una serie de planos y dibujos, para poder saber que es lo que van a levantar. Un ingeniero desarrolla planos de la estructura de las columnas y cimientos del edificio, mientras que un arquitecto diseña un plano arquitectónico de las fachadas y vistas internas y externas del edificio, las cuales revestirán a la estructura del edificio. El diseño arquitectónico tanto externo como interno ocultará a los ojos humanos la estructura de las columnas y planchas de concreto, ya que lo único que percibirá será la funcionalidad de cada cuarto del edificio para satisfacer sus necesidades de espacio, pero bajo una ambientación de colores, texturas, luces, mecanismos de control (apagadores, puertas, entre otros), etc.

El diseño de un sistema cliente/servidor, se asemeja mucho a la construcción de un edificio, ya que como el ingeniero y el arquitecto, se deben desarrollar la estructura y las fachadas del sistema.

La arquitectura cliente/servidor que se propone en este trabajo, es como se muestra en el diagrama

Arquitectura Cliente/Servidor



Arquitecta cliente/servidor a seguir

El servidor o back-end del sistema se encargará del almacenamiento, seguridad e integridad de *la base de datos*, además de llevar a cabo aquellos procesos de alta complejidad, lo cual se define como *lógica transaccional* . Mientras que el cliente llevará a cabo la *interface con el usuario* a través de los GUIs, por medio de los cuales se realiza una *presentación lógica* de los datos, que soportan las necesidades de información del usuario final.

DISEÑO

Durante la fase de análisis nos enfocamos en el *qué* debe hacerse, por lo cual creamos tres modelos, para que a partir de esto, en diseño debemos definir *cómo* se va hacer, para poder implementar un sistema, el cual refleje la realidad de un contexto.
Para un sistema cliente/servidor, el diseño se subdivide, en dos partes; el diseño del servidor y del cliente: el primero, trata de la creación de un esquema relacional, para la base de datos del sistema, la cual almacene datos, sea consistente, completa y segura, así como de llevar a cabo transacciones de alta complejidad del sistema. Mientras que para el cliente, se trata de diseñar una interface gráfica, la cual se encargue del control de los procesos y transacciones del sistema, así como la presentación de los datos, este diseño debe orientarse hacia la programación orientada a objetos y eventos, como filosofía de desarrollo.

Objetivo

"Definir como determinados datos y procesos en un área de negocios son implementados en términos de eventos definidos y como estos deben interactuar con el usuario". [1]

Características

- Diseño del cliente y servidor como elementos autónomos.
- Modelar eventos que involucren procesos y datos
- Crea sistemas flexibles y abiertos al cambio
- Automatizar el diseño y documentación
- Crea prototipos

Productos.

- Esquema relacional de la base de datos
- Prototipos

Insumos

- Modelo de entidades u objetos
- Modelo dinámico
- Modelo funcional

Los pasos a seguir en la fase de diseño, son los siguientes:

- **Diseño del Servidor de datos**
  - Esquema relacional de la base de datos
- **Diseño del Prototipo**
  - Diseño del Diagrama de Flujo de Eventos para el Prototipo
  - Diseño del Diagrama de Flujo de Objetos para el Prototipo
  - Construcción del Prototipo GUI
  - Programación de Librerías de Clases
  - Programación de Funciones
  - Definición de Estándares

[1] Seminario, Análsis y Diseño de Sistemas, Factun Infomática, Macera, p7

DISEÑO DEL SERVIDOR DE DATOS

El proceso del diseño del servidor de datos implica el desarrollo de un Esquema relacional de los datos, el cual, no es más que una transformación de enfoque, del Diagrama Entidad Relación, junto con la definición de otras características, para crear un base de datos consistente y segura.

Diseño Del Esquema Relacional De Los Datos

Una base de datos relacional, tiene una estructura definida, la cual almacena datos que guardan cierta relación entre si, por lo cual deben de existir mecanismos para garantizar que los datos están seguros y guardan cierta integridad lógica entre ellos, por lo cual debemos de implementar para este esquema:

- Esquema relacional (Estructura del sistema)
- Mecanismos de integridad de datos del esquema
- Mecanismos de integridad referencial del esquema

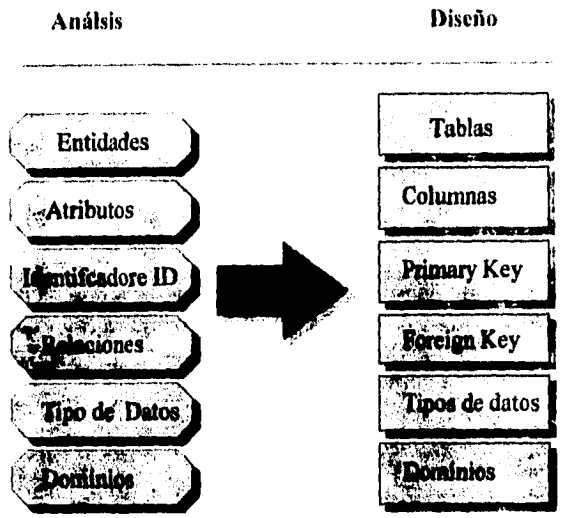
Esquema Relacional

Para poder entrar de lleno al diseño del esquema, en análisis modelamos un DER, este debe cumplir con ciertos requisitos.

- Cada entidad debe tener su identificador *ID*
- No deben de existir relaciones M-M en el análisis
- No deben existir atributos, con cardinalidad 1-M y M-M
- Cada atributo de las entidades, debe tener un tipo de dato asociado.
- Cada tipo de dato debe tener un dominio de valores asociados.

El esquema de la base de datos implica la creación de los elementos que soportaran el almacenamiento de los datos y las relaciones entre estos:

- Tablas
- Columnas
- Llaves Primarias (PK Primary Key)
- Llaves Foráneas (FK Foreing Key)
- Índices
- Tipos de datos



Análisis-Diseño

Entidades a tablas

El proceso de transformación de entidades en análisis, ahora son tablas las cuales se encuentran normalizadas desde la fase de análisis, hasta la tercera forma normal.

Atributos a columnas

Los atributos de las entidades, son convertidos a columnas de tablas en diseño, las cuales deberán heredar de análisis ciertas características, como son la cardinalidad y el identificador único de la entidad:

Atributo identificador de la entidad ID	Campo Único
Atributo con cardinalidad 1-1	Campo que no acepta valores nulos
Atributo con cardinalidad 0-1	Campo que acepta valores nulos

La propiedad de la cardinalidad en diseño se identifica con el status de *NULL* o *NOT NULL*, ya que si es NULL el campo aceptará ausencia de valor, mientras que NOT NULL nunca podrá tener un valor desconocido.

La propiedad de identificador de análisis, en diseño servirá como la columna o columnas que identifican en forma única un registro de esa entidad, así mismo a esta columna se le asocia un ***índice***, el cual es un medio físico para hacer más rápido la busque la de un registro dentro de una tabla.

Identificador ID a Llave Primaria (Primary Key)

En el análisis definimos que cada entidad, tiene un identificador único *(ID)* para cada instancia. Este ahora en diseño se transforma en llave primaria (*Primary Key*) .

```
CREATE TABLE KIT
    (fab_num_fabricante   int NOT NULL,
     pza_num_suministro   int NOT NULL,
     kit_num_suministro   int NOT NULL,
     kit_descripcion      CHAR(40) NULL
)
go
CREATE UNIQUE INDEX XPKKIT
 ON KIT
(
fab_num_fabricante  ,
pza_num_suministro  ,
kit_num_suministro
)
go
```

Relaciones a llaves foráneas (Foreing Key)

Las relaciones entre las entidades definidas en análisis, en diseño ahora se conocen como llaves foráneas, las cuales definen referencias entre las tablas, que cuando son definidas en el servidor, garantizan la integridad referencial, la cual trata de la consistencia de los datos entre tablas. Ya que si queremos insertar o actualizar un registro en una tabla con un atributo que es FK de una tabla, esta llave foránea debe de existir como PK en la tabla propia.

```
CREATE TABLE PIEZA
    (fab_num_fabricante   int NOT NULL,
     pza_num_suministro   int NOT NULL,
     pza_descripción_pieza CHAR(40) NULL,
     pza_desc_estd        CHAR(40) NULL
)
go
CREATE TABLE IMAGEN
    (fab_num_fabricante   int NOT NULL,
     pza_num_suministro   int NOT NULL,
     img_imagen           image NULL
)
go

exec sp_foreignkey IMAGEN, PIEZA,
   fab_num_fabricante,
   pza_num_suministro
```

Tipos de datos y dominios de información

Los tipo de datos definido en análisis se consideran de una forma lógica, mientras que en el diseño ahora pasan a ser tipos de datos físicos, ya que su dominio de información debe de estar definido, bajo los tipos de datos que soporte el manejador de base de datos, que se utilizará.

- Tipos de datos del sistema
- Tipos de datos definidos por el usuario

Tipos de datos del sistema

Son los datos que soporta una base de datos relacional, por lo general son:

Enteros			
tinyint	1	0 a 255	88
smallint	2	-+32767	23445
int	4	-+2,147,483,647	479,990
Números exactos			
numeric(p,s)	2 a 17	$-+10^{38}$	55.903
decimal(p,s)	2 a17	$-+10^{38}$	-99.34
Reales			
float	4 a 8	depende del equipo	3.243
double precisión	8	depende del equipo	
real	4	depende del equipo	
Date/Time			
smalldatetime	4	1/1/1900 a 6/6/2079	
datetime	8	1/1/1753 a 12/31/9999	
Carácter			
char(n)	n	Hasta 255 caracteres	
varchar(n)	variable	Hasta 255 caracteres	
nchar(n)		Hasta 255 caracteres	
nvarchar(n)		Hasta 255 caracteres	
text(n)		2,147,438,647	
Money			
smallmoney	4	-+ 214,748.3647	
money	8	-+922,337,203,685,477.58	
Binary	n	255 bytes	
varbinary	variable	255 bytes	
image		2,147,483,647	
Bit string			
Bit	1	0 o 1	

Tipo de datos para Sybase[2]

La tabla anterior de tipos de datos, son operados por el manejador relacional Sybase.

[2] Curso Transact SQL Sybase, Tomo 1, p37

Tipos de datos definidos por el usuario

En los manejador de base de datos, podemos crear tipos de datos a partir de los datos del sistema. Combinando características de los tipos de datos o valores por default.

Se crea un tipo de dato llamado tid, el cual es un varchar de 6 caracteres y no soporta valores nulos

exec sp_addtype tid, "varchar (6)", "not null"

La definición de un tipo de datos nos proporciona grandes ventajas, ya que si nuestros atributos deben tener más de una característica, tal como el tipo de datos *tid*, solo basta con crear ese tipo, y asignarlo a cada atributo desde análisis.

Mecanismos De Integridad De Datos Del Esquema

La integridad de datos es el cumplimiento de un dominio de información por parte de una columna de una tabla. Estos dominios de información, se definen a partir de las reglas del negocio, ya que una empresa que se dedica a vender productos químicos, una política de esa empresa es que *no se surtirán pedidos menores a una tonelada*, por lo que en la columna *cantidad de material*, de la tabla *pedido*, no aceptará valores menores a una tonelada.
Las reglas de integridad de datos, se definen desde análisis, cuando damos valor a los tipos de datos.

La implementación de la integridad de datos la podemos definir en un RDBMS por medio de:

- Integridad Declarativa
- Integridad por medio de objetos

Integridad declarativa

La implementación de la integridad de los datos, en forma declarativa, es desde la declaración del *create table*, mediante los siguientes mecanismos:

- Defaults
- Restricciones de Chequeo (Check constraint)
- Restricciones Única (Unique constraint)
- Restricción de la llave Primaria (Constraint Primary Key)

Defaults

Es la declaración a nivel columna, de valores que tomará un campo de información por default en el caso de una inserción al registro de ese campo.

```
create table país
( país_cve    int not null,
  país_nom    varchar(40) default "USA")
```

Cuando se haga una inserción de un registro en esa tabla, si no se especifica el valor de *país_nom*, será instanciado con *USA*.
El valor por default, debe corresponder al tipo definido para ese campo.

Restricción de Chequeo (Check constraints)

Es una declaración a nivel tabla de un dominio de información, para una o varias columnas. Los constraints pueden especificar: lista de valores, rango de valores, mascaras de edición y condiciones para validar datos. Los constraints actúan durante las acciones de actualización e inserción.

Los constraints, son definidos a nivel columna o a nivel tabla.

A nivel columna.

Es cuando declaramos un constraint que afecta a una sola columna.

```
create table cliente
(cte_cve    int  not null,
  cte_nom   char(40)  null,
  cte_tel     char(10) null
    constraint con_cte_tel
     check ( cte_tel like "99[0-9[[0-9[999"),
  cte_dir     char(10) null)
```

A nivel tabla

Es cuando declaramos un constraint que afecta a más de una columna.

```
create table factura
( fact_num int not null,
  fact_total  num(10,3) null,
  fact_iva  num(10,3) null,
  fact_totaliva  num(10,3) null,
constraint con_total_check
check (fact_totaliva=fact_total+fact_iva))
```

Restricción única (Unique Constaint)

Es la declaración de un constraints, el cual especifica que para una o varias columnas existirá un valor único en esa tabla y no se repetirá en otra instancia de la tabla. Esta declaración crea automáticamente un índice. Este constraint puede ser definido a nivel columna o a nivel tabla.

create table producto (prd_cve not null **unique,** prd_nom char(40))	crate table factura (fact_num not null, fact_año not null, fact_total null) **unique** (fact_num, fact_año)

Restricción de la Llave Primaria (Primary Key Constraint)

Es la declaración de un constraint, en donde hacemos referencia a una llave primaria, la cual no tendrá valores repetidos en la tabla, no permitirá valore nulos.

create table producto (prd_cve **primary key,** prd_nom char(40))	crate table factura (fact_num not null, fact_año not null, fact_total null) **primary key** (fact_num, fact_año)

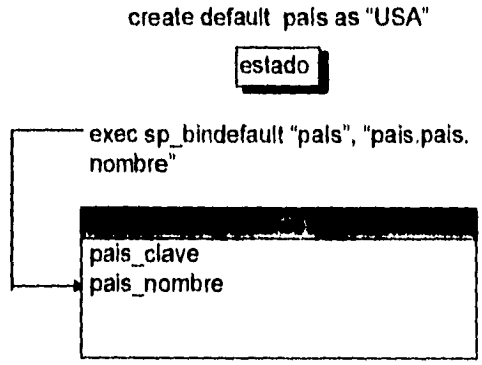
Implementación de la integridad de dato por medio de objetos

Otra forma de poder llevar a cabo la implementación de la integridad de lo datos, es por medio de la creación de objetos como parte de la base de datos, lo cual nos permite asociar esos objetos a varias tablas. Mientras en la integridad declarativa, los criterios solo se definen para una sola tabla. Los objetos por los cuales podemos definir la integridad son:

- Defaults
- Reglas

Defaults

Es un valor determinado previamente, para ser asignado a una columna de una tabla

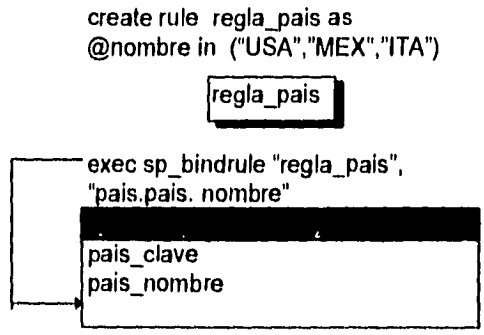


Reglas

Las reglas son objetos de la base de datos las cuales definen una restricción a una columna, para implantar la integridad de un dominio de información, tales como:

- Lista de valores
- Rango de valores
- Mascaras de edición
- Implementación de condiciones

Las reglas son aplicables cuando actualizamos o insertamos un dato.



Asi como las reglas y los defaults, pueden ser asociadas a una columna de una tabla, también a un tipo de dato definido por el usuario, se le puede colgar una regla o default.

Integridad Referencial

La integridad referencial, trata de la consistencia de los datos entre tablas, ya que en un modelo relacional, la explotación de la información se hace, a partir de las relaciones entre tablas.

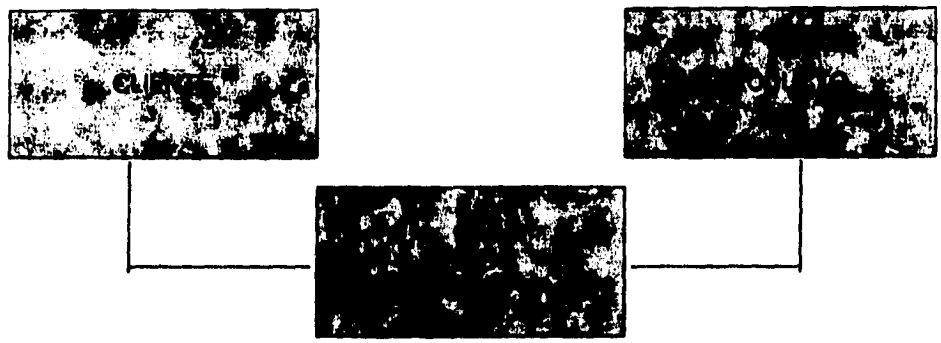
Los criterios estándares para la integridad referencial, son definidas debido a tres eventos: inserción, actualización y borrado.

- Inserción.
  - Cuando insertamos una PK, esta debe ser única.
  - Cuando insertamos una FK, esta debe existir como PK. De lo contrario no se realiza la inserción.
  - Cuando insertamos una FK que no existe como PK, permita la inserción a la tabla padre y después a la hija (Inserción en Cascada)

- Borrado.
  - Cuando borramos una PK, y esta tiene una FK, no se realiza el borrado. (Restricción borrado padre-hijo)
  - Cuando borramos una PK, y esta tiene una FK, se realiza el borrado de la PK y la FK (Borrado en cascada padre-hijo)
  - Borrar la llave primaria y nulificar los valores de la llave foránea
  - Mover los renglones que contienen la llave foránea a una tabla de valores históricos

- Actualizamos.
  - Cambiar la correspondiente llave foránea, acorde con el valor (Actualización en cascada padre-hijo)
  - No permitir la actualización, hasta que se actualice la llave foránea (Restricción)

En cada evento los criterios son mutuamente excluyentes, ya que la selección del criterio dependerá de las reglas del negocio para el sistema. La forma para definir estos criterios es para cada evento un criterio por cada tabla que tiene una relación, es decir si una tabla tiene relación con tres tablas, cada una de estas relaciones deberá tener un criterio de inserción, de actualización y de borrado.



A partir del diagrama entidad relación de análisis, podemos ver que la entidad PEDIDO, tienen dos relaciones uno con CLIENTE y otra con PRODUCTO, debemos definir para la entidad PEDIDO seis criterios de integridad referencial, tres para la relación CLIENTE-PEDIDO y otros tres para PRODUCTO-PEDIDO, para inserción, actualización y borrado. Estos criterios serán para la entidad PEDIDO, ya que las entidades CLIENTE y PRODUCTO son padres para la entidad PEDIDO.

Debemos de documentar los criterios de integridad referencial, a través de la definición en una hoja de calculo o procesador de palabras de cada uno de los criterios en las relaciones de nuestra base de datos.

Los objetos por medio de los que podemos implementar la integridad referencial, en una base de datos, son:

- Restricciones (Constraints)
- Disparadores (Triggers)

Restricciones (Constraints)

Como se explico anteriormente, los constraints nos ayudan a definir integridad de datos, pero también referencial. Esto por medio de la asociación de constraints a las relaciones que existen entre las tablas de una base de datos.
Estos constraints, nos permiten garantizar la integridad de las relaciones entre tablas, para los criterios de Inserción y Actualización.

Los constraints se pueden implementar a nivel columna, cuando solo afecte una sola columna de la tabla y a nivel tabla, cuando sean más de dos columnas.

A nivel columna.

Es cuando declaramos un constraint que afecta a una sola columna.

```
create table factura
( fact_num int not null,
  prd_cve  int null
   constraint producto
   references producto (prd_cve),
  fact_total num(10,3) null,
  fact_iva  num(10,3) null,
  fact_totaliva num(10,3) null)
```

La tabla producto debe de existir, así como el campo prd_cve, y debe ser un campo único, y no se podrá insertar o actualizar, si el valor no existe en la tabla de *producto*. Primero se deben de insertar o actualizar en producto, para poderlo hacer en factura.

A nivel tabla

Es cuando declaramos un constraint que afecta a más de una columna.

```
create table factura
( fact_num int not null,
  fact_total num(10,3) null,
  fact_iva  num(10,3) null,
  fact_totaliva num(10,3) null,
  ped_tervta  int null,
  ped_cond   int nul,

 constraint venta
  foreign key (ped_tervta, ped_cod)
    reference pedido (ped_tervta, ped_cond))
```

La tabla Pedido debe existir, así como los campos ped_tervta y ped_cond. Los valores que se inserten o actualicen deben ser iguales a los que existen en la tabla *pedido*. Para lo cual se deben de insertar o actualizar primero los valores en *pedido*, y después en *factura*.

Los constraints, son aplicables a los eventos de inserción y actualización, así como para borrado, ya que si una tabla padre tienen varias instancias hijas, este padre no podrá ser borrado, hasta que las instancias hijas se borren.

Disparadores (Triggers)

Los disparadores son procedimientos almacenados, que son ejecutados cuando sucede un determinado evento, como actualización, inserción o borrado de un registro o más, en una tabla.
Los triggers son medios más flexibles para garantizar la integridad referencial. Ya que podemos cumplir todas los criterios de integridad referencial para inserción, actualización y borrado.

Los triggers son definidos a través de la programación del criterio de referencia, con código para el RDBMS que se este utilizando.

Al igual que en la integridad de datos, podemos implementarla en diversas formas, también en la integridad referencial, existen varios caminos para implementar cada criterio de integridad referencial que sea hayan definidos:

- Integridad referencial por medio de restricciones (constraints).
- Integridad referencial por medio de disparadores (Triggers).

Por lo que recomendamos el uso de restricciones para implementar los eventos de:

- Inserción.
  - Cuando insertamos una FK, esta debe existir como PK. De lo contrario no se realiza la inserción.
- Actualizamos.
  - No permitir la actualización, hasta que se actualice la llave foránea (Restricción).
- Borrado.
  - Cuando borramos una PK, y esta tiene una FK, no se realiza el borrado. (Restricción borrado padre-hijo)

los cuales son los más comunes, mientras que para implementar los demás eventos, deberemos utilizar la programación de Triggers, tales como : borrados, actualizaciones e inserciones en cascada o algún otro que forme parte del proceso a realizar.

DISEÑO DEL PROTOTIPO

Una vez que diseñamos las columnas y planchas del edificio, el siguiente paso es diseñar el revestimiento de la estructura. La creación de un prototipo implica el diseño de un medio para la explotación de los datos del servidor, según los requerimientos definidos por nuestros clientes o usuarios. El cual sea un modelo a seguir en la fase de construcción para las demás aplicaciones del sistema.
La nueva tecnología de los lenguajes orientados a objetos o eventos, requiere de un diseño diferente a los lenguajes procedurales.

El Cliente como interface ante el usuario, debe de controlar la funcionalidad del sistema, tanto para los datos como para los procesos. La interface debe ser construida en base a objetos, mientras que su funcionamiento se debe basar en eventos, por lo que es necesario conocer el Paradigma de la Orientación a Objetos.

Paradigma Orientado a Objetos

Es una filosofía que conceptualiza al universo en objetos, formados por datos y procedimientos los cuales realizan una cierta función dentro de su universo. Este paradigma es aplicado al desarrollo de sistemas, tanto desde análisis, diseño y programación.

La programación orientada a objetos, se compone de ciertos elementos los cuales son; Mecanismos Básicos (Basic Mechanims), Conceptos (Key Concepts) y Términos Técnicos (Related Technical Terms). Los cuales se describen a continuación.

Componentes

Mecanismos Básicos (Basic Mechanims)

Son los elementos que forman a la programación orientada a objetos. Haciendo una semejanza con la programación estructurada son las variables, constantes y procedimientos en un programa convencional.

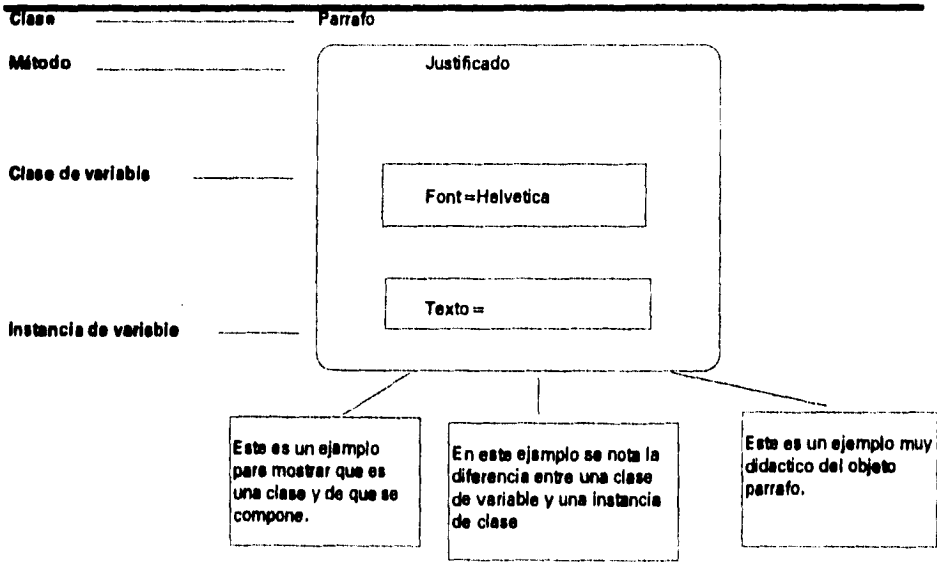
Objetos (Objects)

Un objeto en un elemento autónomo y unitario, el cual actúa y genera información de interés para nuestro contexto. Existen dos tipos de objetos: los pasivos y los activos, los primeros solo actúan cuando son llamados (ejemplo: un botón de Aceptar), los segundos están todo el tiempo activo (ejemplo: un monitor de eventos).

Mensajes (Messages) & Métodos (Method)

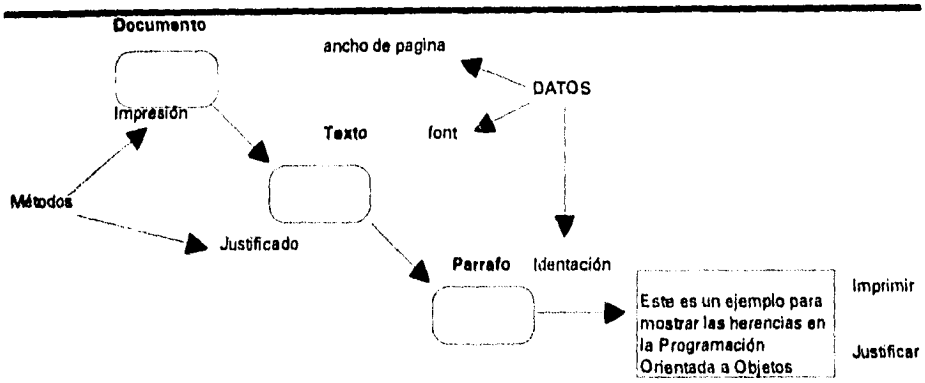
Un objeto pasivo es activado por medio de un mensaje y no por llamados, es decir cuando sucede un evento, este evento activa un objeto el cual ejecutará cierta acción, el medio que activo ese objeto se llama mensaje.
El procedimiento o acción que ejecuta un objeto al ser activado se le conoce como Método.

Clases y Subclases (Classes and Subclasses)
Una clase o subclasse es la implementación de un objeto en código fuente , es decir es la programación de un objeto con datos y métodos.
En programación orientada a objetos los datos tienen dos formas class variables (clase de variables) y instance variables (instancias de variables), la primera nos define el tipo de variable de un objeto y las segundas tienen los valores de la clase de variables asociados a cada objeto creado de la clase.
Las Subclases son derivaciones de una clase, es decir una clase es padre de una subclasse.



Ejemplo de un objeto programado

Herencia (Inheritance)
En un mecanismo el cual comparte automáticamente los métodos y datos de una clase a una subclase.



Conceptos (Key Concepts)

Son las características de la programación orientada a objetos.

Encapsulamiento (Encapsulation)

Es la unión de métodos y clase de variables en un objeto, en donde solo esos métodos manipulan sus datos.

Abstracción (Abstraction)

En términos de programación se debe de pensar que los objetos son aplicaciones abstractas, es decir que son objetos independientes uno del otro.

Poliformismo (Polymorphism)

Un objeto es activado mediante la sucesión de un evento, por medio de un mensajes, ese objeto puede ser activado como resultado de diferentes acciones y/o eventos.

Persistencia (Persistence)

Cuando un objeto es ejecutado este es creado en memoria y este será destruido de la memoria cuando termine de ejecutarse y así sucesivamente, por lo tanto la persistencia es la cantidad de tiempo que un objeto permanece en memoria. Ya que utiliza apuntadores a memoria, en los cuales crea y destruye objetos.

Términos Técnicos (Related Technical Terms)

Técnicas que soporta la programación orientada a objetos

Ligado Dinámico (Dynamic Binding)

Esta es una característica de los lenguajes orientados a objetos, la cual funciona durante la compilación de los programas, en donde se realiza un ligado entre los datos y los métodos. Creando los objetos cuando se necesitan y una vez utilizados, se destruyen.

Programación Visual (Visual programming)
La programación visual es una manera de crear objetos, a través de un menú, tales como iconos, pantallas, ventanas, scroll bar, barras, ventanas de diálogo, etc.

Blob.
Significa "Binary Large Object" el cual es un método que utiliza la programación orientada a objetos para compactar los datos los objetos de gran tamaño en pequeños objetos.

Ventajas

- El Código de programación es 100% reusable para el desarrollo de nuevas aplicaciones.
- Código en objetos es más fácil mantenible que el código estructurado.
- Reduce el tamaño y la complejidad del código en comparación con el código estructurado
- El aumento de la productividad en la programación, ya que lo compilación del código es mas rápida,
  asi como la reducción de memoria al crear objetos de dinámicos.
- Mejoramiento en la calidad de las interfaces gráficas
- Mayor modulación del código

Desventajas

- Alto grado de dificultad para comprender la filosofía

Diseño Orientado A Objetos

El diseño orientado a objetos, tiene por objetivo la construcción de las interfaces, las cuales controlen los procesos y datos bajo el paradigma de la orientación a objetos. Lo cual se realiza a partir de la codificación de un prototipo, en el cual se define la interface de control y presentación. Este prototipo se diseña a partir de un Diagrama de Flujo de Objetos, el cual nos muestre el flujo de los eventos que suceden en la interface, además muestra la afectación a los objetos por parte del evento que sucede.

Una vez que se tiene ese prototipo, se procede a definir clases de objetos y estándares, para que las demás formas o pantallas del sistema, sean desarrolladas en forma automática, basándose en esos objetos y estándares, y además en el diseño de los respectivos DFO para cada una de estas.

Los pasos a seguir para el diseño del prototipo del sistema son:

- Diseño del Diagrama de Flujo de Eventos para el Prototipo
- Diseño del Diagrama de Flujo de Objetos para el Prototipo
- Construcción del Prototipo GUI
- Programación de Librerías de Clases
- Programación de Funciones
- Definición de Estándares

Las funciones a programar en el cliente, serán las que se muestran desde análisis, en el Diagrama de Descomposición, por lo que para cada una de estas se construirá una aplicación.

Diagrama de flujo de eventos para el Prototipo

Como sabemos un diagrama de flujo de eventos, nos presenta la sucesión de eventos o estados que sufre un objeto o entidad. En este caso debemos definir una forma estándar de funcionamiento para el prototipo, por lo cual por medio del diagrama de eventos podemos definir los estados por los cuales pasará el prototipo, en la realización de ciertos procesos.

En una aplicación de tipo administrativa, las funciones principales que lleva a cabo son: Alta, Bajas, Consulta y Modificaciones, por lo cual a cada una de estas acciones, las podemos definir como estados, los cuales el prototipo deberá de cumplir. La definición del flujo de eventos, es una decisión muy importante ya que a partir de esta deberemos de hacer todas las aplicaciones. (Ver Capitulo 5 en la sección del Diagrama de Flujo de Eventos)

Diagrama de flujo de objetos para e Prototipo

Los lenguajes de cuarta generación de tipo GUI, funcionan basados en el paradigma de la orientación a objetos, la cual nos dice que el mundo se compone de objetos, estos en un lenguaje de programación, se transforman en clases, las cuales contienen datos y métodos. La forma de funcionar de estas clases es a través de mensajes o eventos que lanzan procesos o métodos, los cuales modifican el estado de los objetos. Por lo cual es necesario contar con una herramienta de diseño, con la cual podamos representar el funcionamiento de un programa de este tipo. En el cual se muestre el flujo de los mensajes o eventos y la afectación de estos a las clases u objetos, así como las peticiones del cliente al servidor, y las respuestas, a partir de un evento. La herramienta la cual nos ayudará a resolver este problema, es el Diagrama de Flujo de Objetos, en el cual se muestra como a partir de las interfaces gráficas y sus eventos, se disparan procesos, ya sea hacia en el cliente o en el servidor y las respuestas del servidor al cliente. Con esto podremos garantizar el control de los programas y procesos, desde una interface gráfica.

Los Diagramas de Flujo de Estados son adecuados para la descripción de procesos en términos de eventos, de reglas de activación, de condiciones y operaciones a datos. Ahora es necesario modelar esos eventos para poder construir el cliente, el cual controla por medio de la interface gráfica los programas y los procesos.

"Los diagramas de flujo de objetos proporcionan una arquitectura de los componentes autocontenidos y las interfaces en un contexto"[3]. Al dividir los procesos complejos en partes manejables, se tiene un alto grado de comprensión para la creación del cliente.

En los Diagramas de flujo de Objetos podemos identificar los siguientes elementos:

Entradas/salidas al contexto.

Producto u objeto de interface gráfica.

Objeto conceptual del sistema

Mensaje o evento del cliente.

Llamado/respuesta al/del servidor.

[3] Análsis y Diseño Orientado a Objetos, James Martin, p110

Representa el origen/destino de una entrada/salida al contexto del diagrama del DFO, los cuales pueden ser el presionar un botón con el mouse por parte del usuario o el accionar un menú con el teclado, para el caso de un sistema.

Producto u objeto de interface gráfica.

En una interface gráfica podemos identificar dos tipos de objetos: los objetos de control y los objetos de despliegue, los primeros son los medios para el control de las acciones o funciones del sistema, como son botones, menús, etc., mientras que los segundos son para la edición y despliegue de los datos del servidor, como matrices, campos de edición, cajas combo, etc. Por lo que en el DFO uno de sus objetivos principales es el mostrar los objetos gráficos o interfaces que en programación se mostrará en la pantalla del cliente.

Estos objetos gráficos son la representación visual de un objeto conceptual del sistema, ya que un botón que da de alta un pedido, representa un mecanismo de activación del *método alta* del *objeto pedido* o un campo de edición es un mecanismo de despliegue de un atributo del *objeto pedido* en el *método consulta*.
Por lo cual cada objeto gráfico es una interface gráfica para actuar sobre un objeto conceptual del sistema, el cual en forma física se encuentra implementado en una tabla relacional del servidor.

Form Window	Forma o pantalla
Table Windows	Hoja de calculo
Dialog Box	Caja de dialogo
MDI Window	Conjunto de formas
Data field	Campo de edición
Multiline field	Campo de edición múltiple
Push button	Botón
Radio button	Radios Botón
Check box	Caja de chequeo
Option button	Botones de opciones
List box	Lista en caja
Combo box	Caja de combo
Picture Imagen	Imágenes
Horizontal scroll bar	Barra de scroll
Vertical scroll bar	Barra de scroll
Column	Columnas
Gráficas	Gráficas

Objetos Gráficos[4]

[4] Objetos que maneja SQLWindows. Gupta Inc.

Objeto conceptual del sistema

Representa al objeto o entidad que interviene en el contexto del sistema. En este diagrama se muestran para ver como a través de los objetos gráficos, son controlados en cuanto a su funcionamiento, estado y procesos. Para que el usuario obtenga los resultados requeridos por el sistema.

Mensaje o evento del cliente.

Representa los mensajes o eventos que interactuan en la interface gráfica del cliente, y los cuales hacen que se disparen procesos en el sistema. Ya que si en pantalla tenemos un botón para ejecutar una consulta, el *evento presionar botón de consulta*, producirá que se ejecute el proceso para la consulta hacia el servidor.

Llamado/respuesta al/del servidor.

Como sabemos en los sistemas cliente/servidor, el cliente hace peticiones al servidor y este le contesta, con este símbolo podemos ver cuando y que objeto gráfico del cliente hace una petición o recibe una respuesta del servidor. Así mismo este símbolo nos muestra el flujo de procesos dentro del servidor, ya que una petición del cliente al servidor, puede ser una transacción, la cual se lleva dentro del servidor y realiza ciertos cambios en las tablas del sistema, y por último da la respuesta al cliente.

Construcción Del Diagrama De Flujo De Objetos

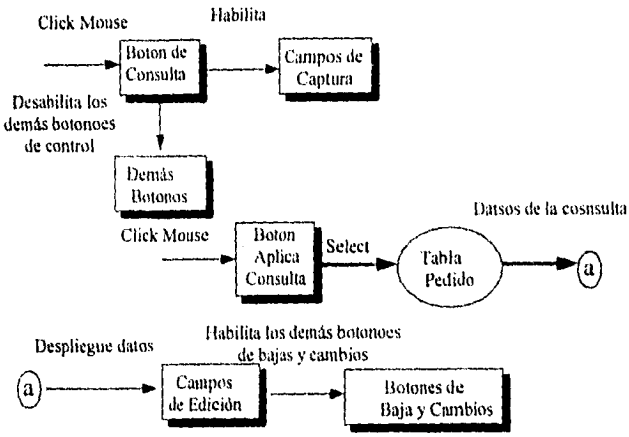
La construcción del DFO se realiza a partir del Diagrama de Descomposición, tomando un modulo de este para el desarrollo del prototipo, esta mima mecánica de construcción del prototipo, será para las aplicaciones del sistema.
Una vez que definimos que para cada función del sistema se debe realizar un DFO, debemos identificar del modelo dinámico en análisis, el correspondiente Diagrama de Esquema de Eventos y Diagrama de Flujo de Datos de cada función del sistema.

Cada diagrama de esquema de eventos describe la historia de un objeto, mientras que una transacción es un cambio de estado correspondiendo la operaciones del objeto. Se puede asociar una operación con cada evento recibido por un objeto. En el diagrama de esquema de estados la acción ejecutada por una transacción depende del evento y estado del objeto. Si el mismo evento puede ser recibido por más de un objeto, el código implementado el algoritmo debe depender del estado.

Un evento mandado por un objeto puede representar una operación en otro objeto. Una acción o actividad iniciada por una transición en el diagrama de estados puede expandirse a un diagrama de flujo de datos más completo en el modelo funcional. La red de procesos dentro del diagrama de flujo de datos representa el cuerpo de la operación. El flujo de datos en el diagrama correspondiente a valores intermedios de la operación. Los procesos en el diagrama de flujo de datos constituyen suboperaciones que pueden ser operaciones en el objeto original o en otros objetos.

Una vez que identificamos cada DEE y los DFD para los procesos que se llevan a cabo, el siguiente paso es la construcción del DFO. Esta se realiza en términos de las funciones de alta, baja, cambios y consultas para un contexto, ya que como se definió en análisis estas son las tareas primordiales de la automatización de procesos.

Proceso de Consulta de Pedidos



En el ejemplo anterior definimos un DFO para el proceso de consulta, en este proceso contamos con dos objetos gráficos principales, los botones de control para consultas y los campos de edición y despliegue de los datos. El diagrama comienza con un mensaje o evento, de presionar el click del mouse sobre el *botón de*

consulta, esta acción produce dos mensajes; uno dirigido hacia los campos de edición del la forma, el cual habilita a estos para ser editados, mientras que el segundo mensaje inhabilita a los demás botones de la forma, como pueden ser altas, bajas y cambios. Esta funcionalidad se da debido a que si se presiono el botón de consulta, la forma GUI esta en un estado de consulta, en el cual podremos hacer un Query dinámico a partir de editar en los campos de la forma los criterio de búsqueda por lo cual se habilitan para la edición, mientras que en este estado de consulta de la forma, no podemos hacer altas, bajas ni cambios, por lo cual se inhabilita los botones. Una vez que se edito el criterio de consulta, se debe proceder a la activación del botón de *Aplicar la consulta*, es decir se ejecuta el proceso de enviar una clausura select al servidor de la base de datos, es por lo cual se muestra una línea doble entre el botón de aplicar y la tabla de pedido en el servidor, una vez que se ejecuto la consulta, el servidor envía la respuesta al cliente, el cual la recibe y despliega los datos de la consulta en los campos de edición de la forma GUI. Cuando se realizo esta acción, ahora se envía un mensaje de habilitación a los botones de bajas y cambios, ya que a partir de un registro consultado del servidor, podremos hacer borrados y modificaciones.

Para definir las demás funciones del sistema como son altas, bajas y cambios, se realizan en forma semejante, por lo que se recomienda primero hacer el DFO de las funciones de altas, consultas, después bajas y cambios.

El diagrama de flujo de objetos, como lo definimos, nos ayuda a construir y diseñar una aplicación. En primera instancia, este diagrama lo utilizaremos en diseño para la construcción del prototipo, posteriormente en la fase de construcción se utilizará para construir las demás aplicaciones del sistema.

Construcción Del Prototipo Inicial

La labor de programar varias formas para el cliente, debe ser una tarea muy estandarizada, es decir si en un sistema van existir 10 formas, estas deben de tener los mismo colores, tipos de letras, la misma distribución, así como los reportes que requieran los usuarios.

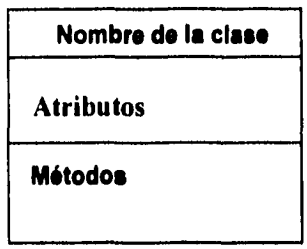
Por lo cual debemos de diseñar un prototipo para cada tipo de aplicación y reporte, un prototipo se asemeja a una maqueta en la construcción de una casa o edificio, ya que esta muestra como será la fachada y vista del inmueble. Un prototipo es semejante, ya que nos enseña como se verá el cliente, pero en realidad no realiza ninguna función. Este prototipo se diseña junto con los usuario, para formas y reportes, a su gusto y según sus necesidades. Y a partir de esos prototipos nos basaremos en la construcción del sistema, de todo el sistema.

Diseño y Programación De Clases

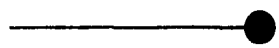
Una vez que construimos el Prototipo, este contienen objetos gráficos como botones, menús, matrices, etc., los cuales serán utilizados para la construcción de las demás interfaces del sistema. Por lo que en los lenguajes orientados a objetos, estos pueden ser programados en clases de objetos reutilizables, con lo cual podemos programar en forma más rápida un sistema, tan solo con llamar a esa clase de objetos y colocarlo en la interface que se necesite.

Diagrama de clases

El diagrama de clases nos sirve para diseñar las clases de objetos dentro del lenguaje de cuarta generación.
La nomenclatura a utilizar en este diagrama es:



Clase



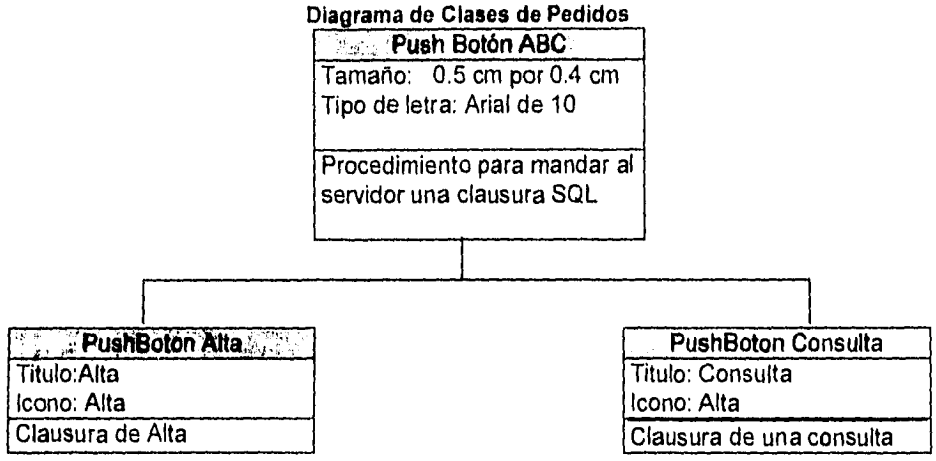
Relación entre un objeto y otro

Herencia de una clase a una subclase

La naturaleza del tipo de metodología a usar, basada en la interface orientada a objetos, hace necesario solo diseñar clases de los objetos gráficos del sistema para el control de la interface y no para los elementos o entidades que intervienen en el contexto.

El diseño de un diagrama de clases, consta de la definición del nombre de la clase, los atributos o datos de la clase y de los métodos o procesos que realiza ese objeto. El criterio para seleccionar que objetos del cliente serán pasados a clases, es dependiendo del numero de veces que se utilizará en el resto del sistema.

Diagrama de Clases de Pedidos



En el diagrama anterior podemos visualizar la Clase Push Botón ABC, la cual es un botón que tiene atributos definidos para su tamaño y el tipo de letra que tiene su rotulo, así como el método para enviar cualquier clausura select hacia el servidor. Esta clase hereda sus características y métodos a otras dos: PushBoton Alta y PushBoton Consulta, los cuales tiene características individuales cada una de estas clases.

Con este diagrama podemos partir para programar las clases del sistema y crear librerías de código, ya que en una aplicación pueden existir más de un botón de alta para cualquier información y lo único que se hace, es mandar pedir la clase, colocarla y adecuarla a ese proceso y si no fuera así tendríamos que programar el mismo código varias veces.

Programación de Funciones

Como observamos en el punto anterior, el proceso de programar clases de objetos estándares para las aplicaciones, es una labor que reduce el trabajo de programación. Ahora por otra parte también debemos de tener una librería de funciones estándares y generales, las cuales sean utilizadas en el resto del sistema.
Tales como funciones para armar fechas, para armar instrucciones Insert, Delete, Update y Select para el servidor, etc.

Definición De Estándares

El proceso de definir clases de objetos y funciones para ser utilizados en las demás interfaces, hace que la codificación del sistema se haga más estándar. Pero en la fase de construcción se tendrán que definir variables y objetos de las clases programadas, por lo es necesario definir estándares de programación en cuanto a:

- Nombres de clases
- Nombre de objetos
- Nombres de variables
- Nombres de programas y versiones

La generación del prototipo, estándares, procedimientos, librerías de objetos y funciones, crea una infraestructura para comenzar con la construcción de las aplicaciones.

RESUMEN

La etapa de diseño para un sistema cliente/servidor, es diferente a los diseños tradicionales, ya que debemos diseñar dos elementos independientes, el cliente y el servidor, los cuales son elementos independientes, pero trabajan en forma conjunta. El cliente como interface de control para procesos y el servidor como medio de almacenamiento de datos y ejecutor de transacciones complejas, el cliente como medio de control debe ser diseñado para ser programado en un lenguaje orientado a objetos, por lo cual la fase de diseño debe orientarse hacia esta filosofía.

Por lo cual en esta etapa debemos obtener dos elementos principales: la generación física de la base de datos y un prototipo GUI.

El primer punto se refiere a la transformación del Diagrama Entidad-Relación de análisis a una base de datos generada físicamente en un RDBMS, la cual cuente con mecanismos para garantizar la integridad de datos y la referencial, a través de ciertos elementos de programación.

El prototipo GUI, será un programa modelo, el cual define una forma de funcionamiento de las aplicaciones, esto basado en eventos. La construcción de este prototipo se basa en objetos, por lo que debemos de seguir una metodología diferente para su diseño. Esto por medio de un Diagrama de Flujo de Objetos, el cual representa los objetos y eventos que intervienen en un aplicación de este tipo.
Una vez que se diseño el prototipo, el siguiente paso es crear librerías de clases de objetos, funciones y estándares, los cuales junto con el prototipo serán herramientas para la construcción de las aplicaciones GUI.

Una que se tienen creados estos dos elementos: la base de datos y el prototipo, estamos en condiciones para poder comenzar con la programación del sistema. En base a la explotación de datos del servidor, a través de los clientes GUI.

De la fase de análisis como proveedor de insumos para el diseño, obtuvimos tres modelos; entidades/objetos, dinámico y funcional.

El proceso de diseño del servidor es un paso muy sencillo y transparente a partir de nuestro Diagrama entidad-relación, ya que una vez que lo tenemos balanceado en análisis nuestro modelo de datos, el siguiente paso es transformar a un esquema de base de datos relacional, así como definir los criterios y reglas del negocio en cuanto a integridad de datos y referencial, para poder programar una base de datos relacional segura y consistente, que desde cualquier punto o cliente que sea explotada funcione eficiente y correctamente.

CAPITULO 6
CONSTRUCCION DE SISTEMAS

La construcción de las aplicaciones de un sistema, es una tarea en la cual plasmaremos los requerimientos de información y procesos definidos en análisis, en líneas de código, lo cual dará como producto un sistema automatizado.

El proceso de construcción de un sistema cliente/servidor debe ser un proceso que lleve corto tiempo y sea muy fácil, esta condición se cumple siempre y cuando los objetivos de la fase de diseño se hayan cumplido. En el diseño como fase terminada obtuvimos la generación física de la base de datos, un prototipo de las aplicaciones del sistema, librerías de objetos y funciones, estándares, procedimientos y un diagrama de flujo de eventos, estos elementos y junto con el modelo dinámico y funcional de análisis, debemos de programar las aplicaciones para el sistema, las cuales principalmente radica en programas GUI, que requieren de ciertos servicios del servidor.

CONSTRUCCION

En el capitulo anterior, mencionamos el ejemplo de la construcción de un edificio, en donde a partir del diseño de los planos, se edifica el inmueble. Para la programación de un sistema cliente/servidor, el caso es el mismo, una vez que contamos con el diseño del cliente y del servidor, el siguiente paso es la construcción de cada elemento del sistema.

Objetivo

Implementar las aplicaciones GUI que satisfagan los requerimientos del cliente final, utilizando
el "Know-How" de diseño y análisis.

Características

- Permite la generación de código automática
- Reduce el tiempo y complejidad de la programación
- Impone reglas desde los niveles de análisis y diseño, para cumplir con los requerimientos del sistema.
- Reduce los costos de mantenimiento y operación

Insumos

- Base de datos generada
- Prototipo GUI, estándares de programación, librerías de objetos, funciones y procedimientos.

Productos

- Aplicaciones GUI del Cliente

El proceso de construcción de las aplicaciones, radica en poder programar por partes, tomando cada función del sistema revisando el modelo funcional los procesos y requerimientos para ese modulo, para poder programar en el servidor los elementos necesarios, tales como vistas (view), disparadores especiales (triggers) y procedimientos almacenados (store procedures). Para posteriormente poder montar sobre ese conjunto de tablas la aplicación GUI, la cual se construirá en base a el prototipo, estándares, objetos, librerías y procedimientos generados en diseño

Por lo cual los siguientes pasos a seguir en la construcción es:

- Construcción del Diagrama de Flujo de Objetos para la Aplicación.
- Generar vistas, disparadores y procedimiento almacenados, para cada proceso.
- Construir la aplicación GUI .

Diagrama de Flujo de Objetos para la Aplicación

Una vez que contamos con toda la infraestructura de diseño, y de inicio la construcción de las aplicaciones, debemos de visualizar del Diagrama de descomposición, cada uno de los módulos que este contienen, ya que estos pasaran a ser las aplicaciones o módulos del sistema, por lo que a partir de DD, revisaremos para cada modulo, el modelo funcional y dinámico para poder saber que es lo que construiremos, así como su modelo entidad/objeto del modulo.
Con el estudio de los elementos anteriores, procederemos a construir el Diagrama de Flujo de Objetos de la aplicación, siguiendo las reglas definidas en Diseño. Con esto podremos saber cuales serán los objetos que intervendrán en la aplicación y a que tablas de la base de datos harán referencia.
El Diagrama de Flujo de Objetos, nos da una guía de las funciones que debe de realizar tanto el cliente como el servidor, así como que elementos debemos de codificar el servidor para el caso de vistas, disparadores especiales y procedimientos almacenados y la naturaleza de estos.

Generar Vistas, Disparadores y Procedimiento Almacenados.

Como mencionamos anteriormente el proceso de construcción de una aplicación, es tomar el conjunto de tablas, que intervienen en el proceso a programar. A este conjunto de entidades le programaremos una serie de elementos, como Vistas, disparadores y procedimientos almacenados, esto dependerá de la naturaleza del proceso.

Vistas

El Diagrama de Descomposición de la fase de análisis, representa funciones del sistema por medio de módulos, cada función tiene asociados procesos y datos. En este caso nos interesa conocer por medio del Diagrama Entidad Relación, los datos que intervienen en el contexto de esa función.

El diseño de una vista de datos será lo que el cliente vea de la base de datos, en el contexto que se encuentre, es decir en un modulo de facturación, el cliente solo podrá ver las entidades que satisfacen los datos necesarios para esa función y no más.

Las vistas es el producto de una unión de las tablas que intervienen en la ejecución de una función, lo cual presenta un contexto definido de datos y no el contexto universal, por lo cual las vistas nos proporcionan:

Seguridad. Las vistas limitan la información que el cliente presenta, y de esta manera estamos restringiendo la explotación de la información para los procesos y para las personas que realicen esos procesos.

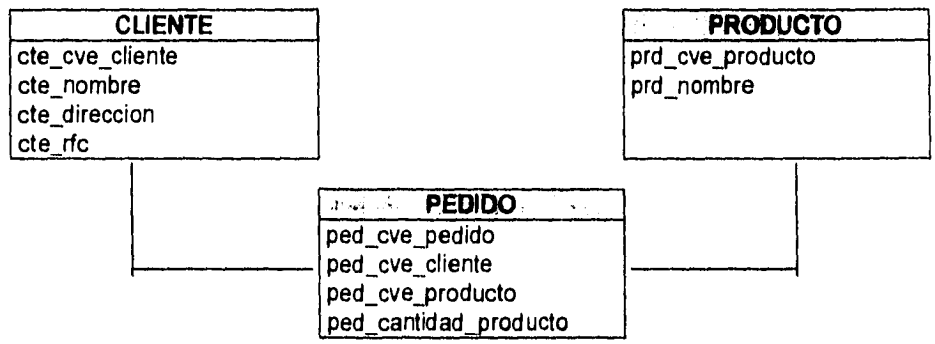
Uso de datos. Las vistas nos dan solo la información necesaria para una función y no más ni menos.

Rendimiento. El uso de vistas aumenta el tiempo de respuesta del sistema, ya que las vistas serán la unión de varias tablas, este proceso se encuentra precompilado desde que se creo la vista, y cuando el cliente requiera esa información solo tendrá que hacer un sencillo *select* a la vista, y de lo contrario se tendría que hacer un *joins* desde el cliente, el cual antes debe ser compilado y verificado, lo cual hace lento el rendimiento del sistema

No en todos los procesos es necesario crear vistas. Esto depende de las consultas que se requieran en el proceso, ya que si esta requiere datos de más de tres tablas, es preferible hacer una vista, mientras que si la consulta es sobre una o dos tablas, basta con un select desde el cliente

La programación en una vista, partiendo de la concepción en diseño, es simplemente el producto de una unión de tablas de la base de datos.

Proceso Alta a pedido




```
create view alta_pedido as
  select cte.cte_cve_cliente, cte.cte_nombre, cte.cte_direccion    cte_rfc,
prd.prd_cve_producto,
          prd.prd_nombre, ped.ped_cve_pedido, ped.ped_cve_cliente,
ped.ped_cve_producto
         ped.ped_cantidad_producto
from
          cliente cte, producto prd, pedido ped
where
          ped.ped_cve_cliente = cte.cte_cve_cliente and
ped.ped_cve_producto=prd.prd_cve_producto
```

Las entidades CLIENTE, PRODUCTO Y PEDIDO, intervienen en la función de consulta a pedidos y los atributos que se muestran en la pantalla, son los definidos para ese contexto, ya que esas tablas contienen más atributos, pero no son necesarios o no se deben ver por seguridad.

Las vistas en un servidor de datos, no son tablas que almacenan información, sino que son tablas virtuales que tienen ciertas ligas hacia las tablas de la cuales se crean. Las vistas solo son aplicables para los procesos de consulta y no para altas, bajas y cambios.
Con las vistas para cada función, ya tenemos lo que el cliente debe solicitar al servidor en cada contexto, ahora debemos definir que procesos debe llevar a cabo el servidor a través de transacciones, para insertar, borrar y actualizar datos, como resultado de ciertos eventos en el cliente.

Disparadores Especiales

Como sabemos un disparador, es un procedimiento almacenado que se ejecuta cuando suceden ciertas condiciones en la base de datos. En el capitulo de diseño, definimos una serie de criterios para garantizar la integridad referencial, los cuales deben ser llevados a cabo por medio de estos disparadores. Pero existe otra aplicación de estos disparadores, los cuales no garantizan la integridad de datos, sino que son ejecutados para realizar cierto proceso en forma automática, cuando se ejecuta cierta condición.

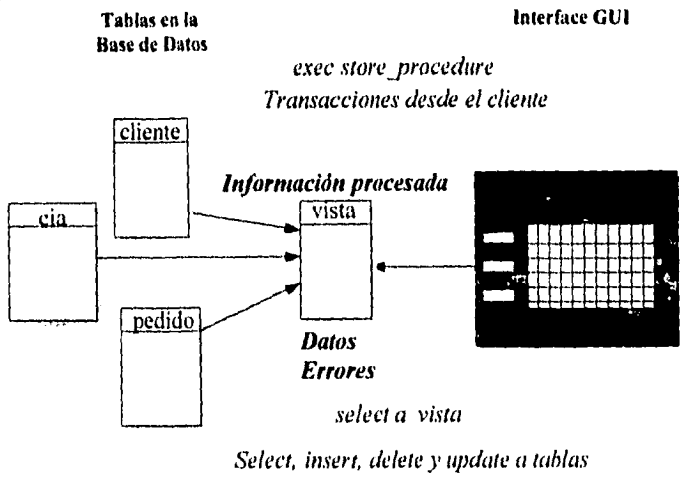
Los disparadores especiales se llevan a cabo cuando se desea procesar ciertos datos, cuando una condición se cumpla, como es el caso del traspaso de saldos de material cuando se cumple el periodo de corte, o el traspaso de ciertos datos de una tabla a una tabla histórica cuando se cumple una fecha o cuando una tabla de un inventario se agota, se genera una solicitud de material, entre otras.

En el caso de la construcción del sistema, dependiendo de cada proceso se debe de programar estos disparadores cuando una condición se cumpla, esto en base a los requerimientos del usuario.

Procedimientos Almacenados

Las transacciones de alto volumen de información se deben de llevar en el servidor. Debido a que son procesos largos y pesados, que un procesador de tipo RISC los llevará a cabo. Estas transacciones se envían desde el cliente con una clausura SQL o con el llamado a un procedimientos almacenados, dependiendo del proceso.

La codificación de las transacciones para cada proceso del sistema, se definen a partir del DFO de cada función del sistema y se programan a partir del modelo funcional de análisis. La forma de programar estos procesos será en forma procedural en el lenguaje de programación propio para el RDMS en el cual se definió la base de datos.



Vistas y procedimientos almacenados

En el diagrama anterior, podemos apreciar como de un lado tenemos al servidor y del otro al cliente, entre estos dos hay una serie de mensajes de comunicación, los cuales son peticiones del cliente al servidor y las respuestas del servidor al cliente. El cliente controla y define que mensajes enviar al servidor dependiendo del proceso o evento que el usuario requiera, los requerimientos que el cliente puede hacer al servidor son:

- Select a vistas definidas
- Select a tablas
- Insert, Delete y Update a tablas
- Ejecución de store procedures
- Envió de transacciones desde el cliente

En el proceso de construcción de la aplicación debemos definir en que eventos o procesos se envían los mensajes al cliente.

Mientras que las respuestas que el cliente puede tener del servidor son:

- Errores
- Datos
- Información procesada

Construir la aplicación GUI .

La tarea de programar las aplicaciones GUI, debe ser una labor muy fácil y rápida, ya que una vez que contamos con el Diagrama de Flujo de Objetos y con el funcionamiento del prototipo, así como con librerías de objetos, funciones, estándares y procedimientos. La construcción se transforma en un ensamblaje departe prediseñadas las cuales solo hay que colocarlas y coordinarlas entre si, siguiendo los estándares definidos para declarar variables y objetos y con los procedimientos para programar las funciones básicas de Altas, bajas, cambios y consultas de un proceso o función.

Una vez que tenemos terminada la fase de construcción obtendremos:

- Una base de datos con niveles de seguridad en la información
- Una base de datos que garantiza la consistencia, integridad y validación de datos, desde cualquier tipo de cliente que sea explotado
- Un servidor que realiza grandes y complejos procesos de información.
- Un cliente que realiza la función de interface gráfica.
- Un cliente que lleva el control de los programas del sistema
- Un cliente que válida la información de entrada, para no enviar datos erróneos al servidor
- Un cliente que satisface las necesidades de información del usuario, a los tres niveles de la pirámide organizacional.

RESUMEN

Este capitulo es el más corto que hemos presentado, ya que como mencionamos la construcción de los sistemas cliente/servidor debe de ser rápida y sencilla, es por lo cual que el contenido del capitulo es poco.

Pero en el cual se menciona que una vez que contamos con la infraestructura de diseño, como es: la base de datos, el prototipo, librerías de objetos, funciones, estándares y procedimientos, podemos realizar una construcción eficiente del sistema, en la cual los pasos a seguir son:

- Construcción del Diagrama de Flujo de Objetos para la Aplicación.
- Generar vistas, disparadores y procedimiento almacenados.
- Construir la aplicación GUI .

El primer punto se refiere al diseño de un plano a seguir para la construcción de la aplicación, para saber que objetos y proceso intervendrán en esta. El segundo se refiere a la programación de vistas, disparadores y procedimientos almacenados en el servidor, necesarios para llevar a cabo una aplicación o proceso y por último el tercer punto nos habla de la construcción de la aplicación GUI del cliente, ensamblando objetos y funciones definidas y siguiendo el prototipo, estándares y procedimientos para la programación de la interface de control.

Al finalizar la fase de construcción del sistema, contaremos con una aplicación cliente la cual cumpla con los requerimientos del usuario, en donde la aplicación se comunique hacia el servidor requiriendo datos y procesamiento de información.

CAPITULO 7

SISTEMA DE VENTAS QUIMICO

A través de cada uno de los capitulos de la tesis, se presentaron una serie de fases para el desarrollo de un sistema cliente/servidor, en donde muchos de los temas presentados, se basaban en conceptos teoricos y en algunos ejemplos. Por lo cual el presente capitulo tienen el objetivo de presentar como se implanto el Sistema de Ventas Quimicos, en base al contenido de la tesis. Para hacer esta presentación solo tomaremos un proceso de todo el sistema.

El modulo que se presenta es el proceso de ***"Certificados de Análisis"***, del cual se analizarán como se codifico el servidor y el cliente, y como se realiza la comunicación entre estos dos.

SISTEMA VENTAS QUIMICOS

El Sistema de Ventas de la División Químicos desarrollado bajo la filosofía cliente/servidor, cuenta con las siguientes caracteristicas:

Servidor

- Una base de datos que es administrada por el RDBMS Sybase de Sybase Inc.
- La base de datos cuenta con 104 tablas relaciónales y 1450 atributos, así como 756 relaciones entre las tablas.
- La base de datos cuenta para su integridad de datos mecanismos cono:
  - Dominios predefinidos de datos
  - Constraints declarados en las tablas
  - Defaults
- La base de datos para garantizar la integridad referencial cuenta con :
  - 756 llaves foráneas
  - 60 triggers
- El hardware en el cual se aloja el servidor, es una estación de trabajo Apollo de Hewlett-Packet, con las siguients caracteristicas:
  - Procesador RISC
  - 32MB en RAM
  - 2GB en Disco Duro
- Se codificaron 20 vistas para consultas del cliente a la base de datos.
- Se codificaron 20 store procedures que son ejecutados por el cliente.

Cliente

- La interface GUI fue codificada SQLWindows de Gupta Inc. como lenguaje orientado a objetos.

- El hardware de los clientes cuentan con las siguientes características.
  - Computadoras HP Vectra
  - Procesador 486
  - 8MB en RAM
  - 100 MB en Disco Duro
  - Tarjeta de red

- La interface cuenta con 23 formas o pantallas de trabajo, además de 20 interfaces para dar mantenimiento a las tablas que son catálogos de datos.

- Se programaron 30 reportes para la explotación de la información

- Cada forma o proceso del cliente cuenta con las funciones de alta, bajas, cambios y consultas del contexto en el cual nos encontramos

- Cada forma o proceso del cliente cuenta con los servicios de:
  - Impresión
  - Correo electrónico
  - Consulta a catálogos de datos
  - Consulta a otros contextos
  - Interface con Microsoft Excel para graficación y edición de datos

- La arquitectura de la interface del cliente quedo bajo dos módulos principales:

  - Ventas Nacionales
  - Ventas Exportación

Ventas Nacional

Se encarga de gestionar el proceso de ventas para el mercado nacional. Las funciones que se llevan a cabo son:

1. **Embarques de planta.** Este modulo se encarga de controlar los embarques de productos que salen de las Plantas de la División Químicos (Quirey, Quimar y GPC), en FFCC o camiones, hacia las bodegas de ventas en el territorio nacional o directamente a las instalaciones de los clientes.
2. **Llegada de embarques en bodega.** Este modulo se encarga de la gestión de los embarques que llegan a las bodegas, de las plantas, para ser convertidos en Inventario de la bodega.
3. **Pedidos nacional.** Este modulo realiza el control de los pedidos de productos para ser surtidos en bodega o entregados directamente a los clientes.
4. **Remisiones de Bodega .** La función de este modulo trata de la administración de las remisiones que son generadas para la entrega de producto a los cliente de bodega.
5. **Solicitud de embarques de planta para bodega .** Tiene la función de generar solicitudes de embarques para la bodega, cuando los inventarios de las mismas llegan al punto de reorden.
6. **Lotes en Bodega de material dañado en bodega.** Realiza la gestión del material dañado en bodega.
7. **Facturación de Pedidos.** Se encarga de la gestión del proceso de facturación, de los pedidos de bodega o de pedidos de cliente.
8. **Notas de Cargo y Crédito.** Este modulo se encarga de la administración de las notas de cargo y crédito para las facturas realizadas
9. **Póliza de Ventas.** Este modulo se encarga de la gestión de las pólizas de ventas, para la contraloría de la División Químicos.
10. **Certificados de Análsis.** Este proceso se encarga de administrar los certificados de composición de los embarques que salen de la planta, de los inventarios y del material facturado. Este tipo de certificado es muy importante, ya que cada cliente requiere de conocer las especificaciones fisico-quimicas del material
11. **Reportes.** Tienen la función de generar los reportes definidos por el usuario, de cualquier tipo de información que se requiera

Ventas De Exportación

Se encarga de gestionar las ventas de exportación de la División Químicos, las cuales se componen de los siguientes procesos:

1. **Pedido de ventas exportación.** Se encarga de la gestión de los pedidos del extranjero, para bodegas propias, bodegas a consignación y a clientes directamente.

2. **Cartas de Embarques Químicos (CEQ).** Se encarga de la gestión de las Cartas de Embarque Químicos, las cuales son documentos que avalan los embarques que salen de México al Extranjero.

3. **Llegada de material a bodegas en el extranjero.** Se encarga de los embarques que llegan de México a las bodegas propias y a consignación.

4. **Inventarios de bodegas propias y a consignación.** Este modulo gestiona los inventarios en el extranjero, de las bodegas propias o a consignación.

5. **Orden de Ventas Químicos (OVQ).** Realiza la gestión de las Ordenes de Ventas Químicos, las cuales soportan una venta del extranjero, ya sea desde bodegas o de ventas directas al cliente.

6. **Orden de Muestra Químicos(OMQ).** Gestiona las Ordenes de Muestra Químicos, las cuales son muestras de material a clientes en el extranjero.

7. **Facturación.** Este modulo gestiona la facturación de las ventas en el extranjero.

8. **Instrucción de Cobranzas.** Este modulo realiza la función de administrar, las instrucciones de cobranzas, las cuales son documentos que nos muestra cuanto hay que cobrarle al cliente, por motivo de una venta.

9. **Notas de cargo y crédito.** Realiza la función de gestionar las notas de cargo o crédito para ventas al extranjero.

10. **Gastos de Mercado.** Gestiona los gastos de venta para las ventas al extranjero, desde que sale el embarque hasta que el producto es entregado.

11. **Póliza de Ventas.** Este modulo genera las pólizas de ventas del extranjero para contraloría.

12. **Comisiones.** Este modulo se encarga del control de las comisiones de las ventas a consignación

13. **Reportes.** Tienen la función de generar los reportes definidos por el usuario, de cualquier tipo de información que se requiera

Interface Grafica para el Usuario

La interface gráfica de control de los procesos y datos del sistema de ventas, como se menciono se compone de dos módulos principales: Ventas Nacional y Ventas Exportación.

A partir de cada uno de los módulos principales se desprenden las respectivas funciones de cada uno de estos. Cada función esta diseñada bajo un prototipo base, el cual tiene la funcionalidad de:

- Altas, bajas, cambios y consultas del contexto principal de la función
- Servicios de Impresión, correo electrónico y consultas a contextos secundarios

Cada una de las funciones anteriores, disparan ciertos eventos y procesos.

Este es el menú principal de Ventas Nacional, cada uno de los botones que se presentan, cuando son presionados, el control del programa pasa a una interface, la cual gestiona la función seleccionada.

CERTIFICADOS DE ANALSIS

Los embarques de material, los lotes de material y el material facturado, por lo general cada uno de estos llevan un certificado de análisis de la composición quimica y fisica de los mismos. Estos también se consideran certificados de calidad, ya que los clientes requieren de conocer estas especificaciones para poder comprar el material para ser transformado a diferentes procesos.
El proceso de certificado de análsis, trata de gestionar las altas, bajas, cambios, copias de los certificados de análsis, así como la consulta de los certificados en general y los certificados asociados a embarques, inventarios y material facturado.

Back-End

Las tablas que intervienen en este proceso son:

- **Compañía.** Es una compañía de la División Quimicos, la cual se hace responsable de la emisión y facturación del pedido. Nombre de la tabla *COMPANIA*.
- **Almacén.** Es el lugar físico, encargado de surtir un pedido de bodega. Nombre de la tabla *ALMACEN*.
- **Empaque.** Es el empaque físico del material o producto, ya sea a granel, en sacos, tambos y cilindros. Nombre de la tabla *EMPAQUE*.
- **Inventario Producto Almacén.** Es la entidad que identifica inventarios de un determinado producto en un almacén
- **Planta.** Es el lugar físico en el cual se produce el material a vender. Nombre de la tabla *PLANTA*.
- **Informe Diario de Embarque.** Es el documento emitido en planta, en el cual se registran los embarque que salen diariamente de la planta y sus destinos. Nombre de la tabla *INF_DIARIO_EMBARQ*.
- **Producto.** Es el catalogo de los productos que se venden en la División Quimicos. Nombre de la tabla *PRODUCTO*.
- **Inventario Lote Almacén.** Es el registro de los lotes que existen en los almacenes. Nombre de la tabla *INV_LOTE_ALM*.
- **Embarque Almacén.** Son los embarques del Informe Diario de Embarques que dirigen a las bodegas. Nombre de la tabla *EMB_ALM*.
- **Embarque Cliente.** Son los embarques del Informe Diario de embarques que se dirigen a los clientes. Nombre de la tabla *EMBARQUE_CLIENTE*.
- **Certificado de Análisis.** Es un documento asociado a un embarque o lote, en el cual se describen las caracteristicas fisico-quimico de estos. Nombre de la tabla *CER_ANALISIS*.
- **Especificaciones.** Es la descripción del Certificado de Análisis. Nombre de la tabla *ESPECIFICACION*.
- **Lugar de Facturación.** Es el catalogo de los lugares en los cuales se factura el pedido. En Oficinas de la división o en bodegas. Nombre de la tabla *LUGAR_FACTURACION*.
- **Factura Nacional.** Es el documento que se genera cuando se factura un pedido. Nombre de la tabla *FACTURA_NAL*.

El esquema de la base de datos como se menciono se generó para el manejador de base de datos Sybase en su versión System 10, en la cual la se implemento la integridad declarativa a nivel tabla, definiendo :

- Valores por default
- Restricciones de condición
- Dominios o tipos de datos predefinidos

Mientras que para la integridad referencial fue definida a través de:

- Restricciones foraneas a nivel tabla
- Disparadores

Apendice A Esquema en SQL para Sybase System 10

Front-End

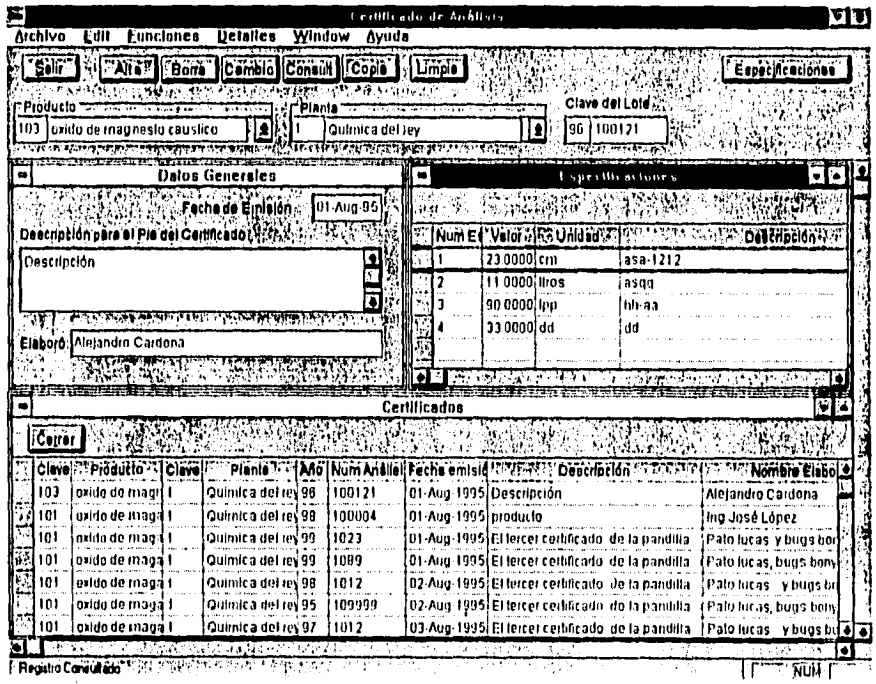
La construcción de esta aplicación para el proceso de Certificados de Análisis, como se sabe se hizo a partir de un prototipo previamente definido, así como basado en el ensamblaje por medio de librerías de objetos y funciones, así mismo se siguieron una serie de estandares y procedimientos.

Ver Apendice B Librerias de objetos y funciones, estandares y procedimientos.

Interface de despliegue

Esta interface GUI, es la interface gráfica para la función de Certificados de Análisis, esta forma se divide en seis secciones:

- Menú
- Barra de herramientas
- Cuerpo de datos de la forma
- Tabla de hijas
- Tablas de consulta en forma de lista
- Barra de estatus



Menú

El menú como en todo programa, nos proporciona una interface de control para realizar ciertas operaciones. Este puede ser operado por medio del teclado y vía mouse. El menú es de tipo Pop Menú, en el cual cuando se presiona la opción deseada, de abre una lista vertical de subopciones. En este caso el menú esta compuesto por:

Archivo

Esta sección del menú, nos maneja el control archivos para la forma activa.

- **Guardar Como** Con esta opción podemos exportar los datos de la forma activa, hacia un archivo de texto, para ser explotado por un procesador de palabras u hoja de calculo.
- **Imprimir.** Imprime el registro de la entidad del contexto, en este caso imprime del Certificado de Análisis.
- **Salir**. Nos hace salir de la forma o interface actual y nos regresa el control al nivel superior inmediato

Edición

Esta opción del menú nos habilita las funciones de **cortar, pegar, copiar** y **borrar** un campo de edición, como se realiza en un procesador de texto o en una hoja de calculo

Funciones

Esta opción del menú, permite realizar las funciones principales de los requerimientos del usuario.

- **Altas.** Esta opción nos permite dar de alta un Certifcado y sus Especificaciones
- **Borrado.** Esta opción nos permite eliminar un registro de la entidad Certificado, el cual es el que se despliega.
- **Cambios.** Realiza un cambio en alguno de sus atributos y no en su llave primaria, del registro desplegado de la entidad de Certificado
- **Consultas.** Realiza la consulta de uno o varios registros de Certificado de Análsis y sus Especificaciones. Esta consulta permite hacer *Querys by examples*. Así como los embarques, inventarios y facturas asociados a los Certificados.
- **Copia.** Esta función realiza una copia de un registro de la tabla de Certificado de Análisis, el cual es desplegado en la interface y al cual se le modifica su llave primaria
- **Aplicar Altas, Aplicar Borra, Aplicar Cambio y Aplicar Copia.** Nos permite aplicar alguna alta, baja, cambio o copia en la base de datos.
- **Limpiar.** Limpia todos los campos de edición de la forma y deja a esta en su estado inicial.

Detalles

Nos permite crear una interface propia para registrar, borrar, consultar y modificar los detalles adicionales de un Certificado de análsis, como son las Especificaciones.

- **Especificaciones.** Un certificado de análsis, puede tener más de una especificación, por lo cual esta opción abre una tabla en la cual se gestionan las altas, bajas, cambios y consultas de las especificaciones.

Servicios

Esta opción nos permite tener servicios adicionales para la forma.

- **Correo Electrónico.** Esta opción nos permite enviar un correo electrónico, del pedido que se despliega actualmente.

Ventana

La interface GUI de Pedido y todas las demás, son de tipo MDI (Multiple Document Interface), es decir es una forma la cual nos permite tener abiertas al mismo tiempo varios documentos o formas en la misma. Un ejemplo análogo es cuando en un procesador de Texto u Hoja de calculo, podemos tener varios archivos abiertos a la vez. Esta opción del menú nos permite organizar las ventanas abiertas en la forma MDI:

- **Organizar verticalmente.** Permite visualizar las ventanas activas en forma vertical.
- **Organizar en mosaico.** Permite visualizar las ventanas activas, en forma de mosaico.
- **Lista de ventanas activas.** Es una sección que permite visualizar las ventas activas, para poder pasarse de una a otra.

Barra de herramientas

Esta sección contiene botones de control para realizar ciertas funciones, las mismas que se realizan en el menú de *funciones y detalle*. Los botones (pushButtons) son : Alta, Borrar, Cambios, Consultas, Copia, Limpiar y Aplicar alta, borra, copia y cambia, y para detalles Especificaciones.

Así mismo también contiene los objetos de despliegue gráfico de los atributos de la llave primaria de la entidad CER_ANALSIS (certificado de Análisis), como son : df_pta_cve_pta, cmb_pta_raz_soc_pta, df_pdt_cve_prd, cmb_pdt_nom_espaniol_prd, df_cea_anio_anslis y df_cea_num_analsis, el el boton pbEspecificaciones, para abrir la tabla la cual contiene los atributos de la tabla CERTIFICADO

Cuerpo de datos de la forma.

Esta sección contiene todos los objetos de despliegue de los atributos no PK de la tabla Cerificado de Analisis (CER_ANALISIS). Como son:

df_cea_fec_elaboracion
ml_cea_descripcion
df_cea_nom_elaboro_certf

Tablas Hijas

Es una tabla, la cuale despliegan las Especificaciones (Tabla ESPECIFICACIONES) de los Certificados de Análsis. En donde a cada Certificado de corresponde una o mas registros de Especificaciones.

Tablas de consultas

Son tablas en las cuales se despliegan los registros de los Certificados en general, de los certificados asociados a los Embarques para cliente, para almacen, los lotes del inventario y los embarques facturados. Es una tabla por cada consulta.

Barra de estatus

Es el pie de la forma, en la cual se despliegan mensajes del estado o situación de la forma.

Interface de control de las funciones del cliente

La interface de despliegue anterior (forma), cuenta con los objetos GUI necesarios para ejecutar los procesos requeridos por el cliente final, así como para mostrar la información que el usuario necesita. Pero esa forma GUI, lleva a tras de todo esto, una serie de procesos internos, que son realizados por el cliente y por el servidor, para llevar a cabo la función requerida.
Por lo que enla sigueinte sección se explicará como es la ejecución de los procesos que el cliente controla.

Estado inicial

La interface o forma de Certificados de Análsis, cuando nace tienen un estado inicial de funcionamiento, del cual podemos partir para realizar ciertas funciones. Del estado inicial de la forma, podemos hacer una alta o una consulta de registros. Por lo que cuando la forma nace, los botones de Alta y Consulta estan habilitados para ser presionados, mientras que los demás botones como Borra, Cambio, Copia y Especificaciones, se presentan desabilitados. Así mismo las opciones del menú se encuentran de la misma forma que sus respectivos botones.

Estado Inicial De La Forma

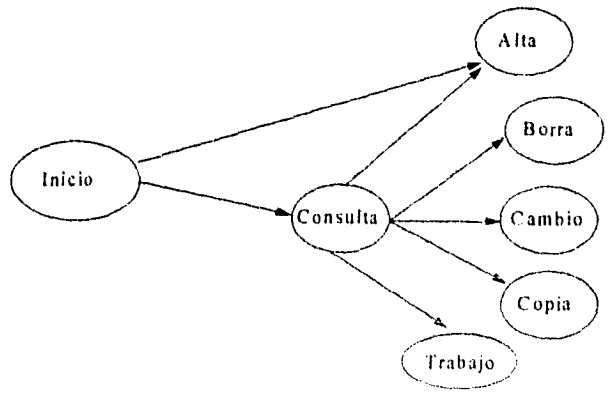
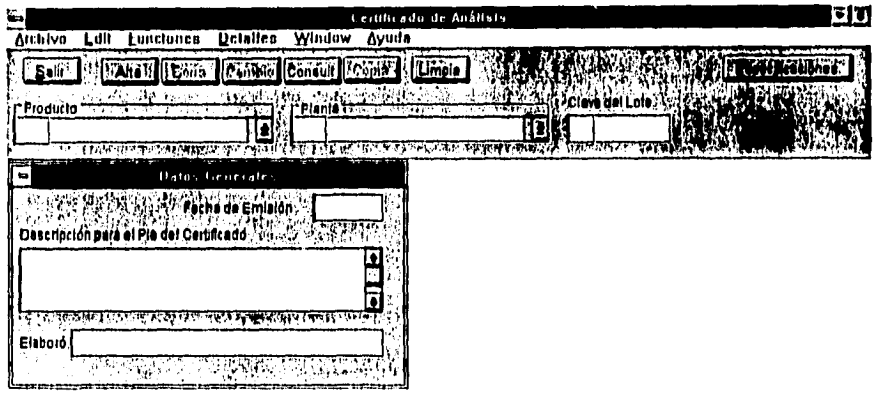


Diagrama de Flujo de Datos.

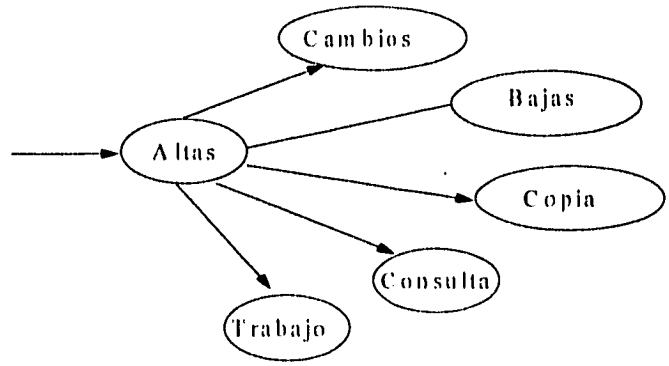


Botón de alta

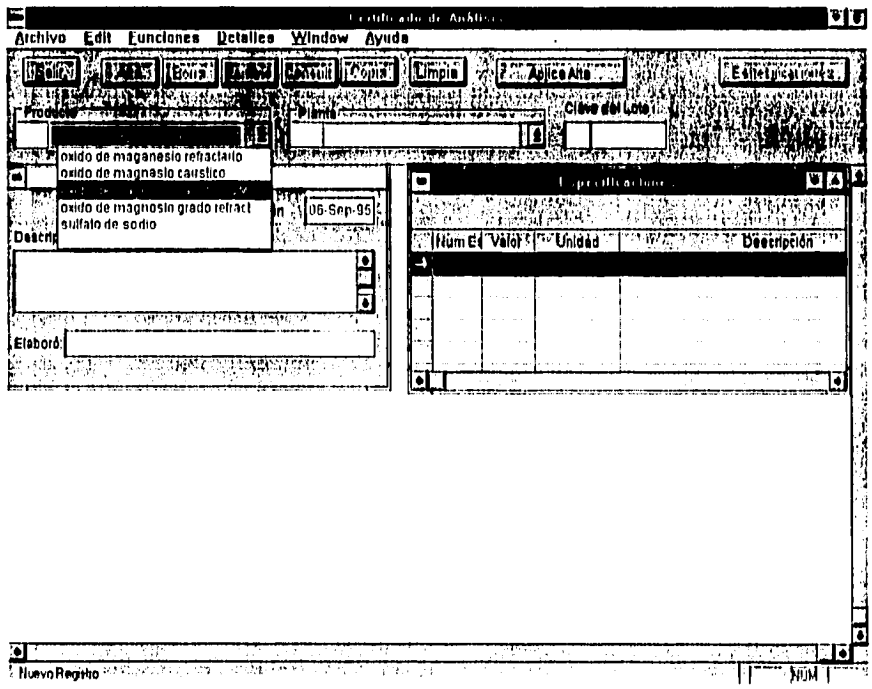
A partir del estado inicial, podemos generar una alta de un registro. Por lo que una vez que se presiona el botón de Alta, la forma entra en un estado para capturar los datos en los campos de la forma, por lo cual los botón de Consultas se desabilita y por ende los demás botones como Borra, Cambio y Copia también. Así mismo automaticamente se abre la tabla de ESPECIFICACIONES, para capturar las especificaciones, y aparece en la forma un boton con la leyenda de Aplica Alta.

El proceso de alta de un registro, es la captura de ciertos campos para hacer un *insert* sobre la base de datos, una vez que se capturan los datos del Certificado de análisis y las Especificaciones, el usuario debe de presionar el botón de Aplica alta, lo cual hace una validación para verificar que el registro a dar de alta tenga la llave primaria completa y que los atributos no nulos en la base de datos, contengan algún valor, si esta validación es negativa se presenta un mensaje el cual indica que campo es el que falta ser llenado, mientras que si es positivo, el cliente ejecuta desde la PC una transacción hacia el servidor. La cual incluye clausuras ***Insert*** para la tabla CER_ANALSIS y uno o varios para la tabla ESPECIFICACIONES.

Estado De Alta De La Forma



Despues de un alta podemos efectuar los procesos que se muestran



Botón de consultas

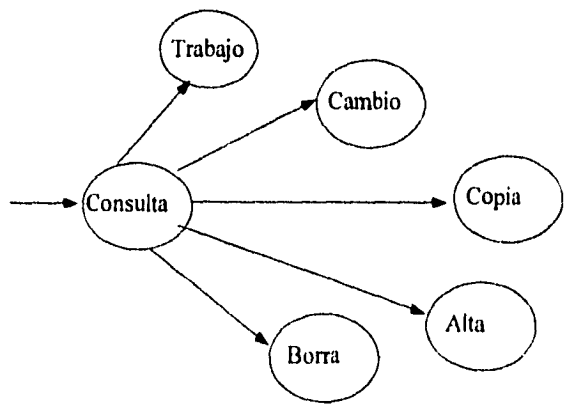
Para la consulta de los certificados, debemos presionar el boton de Consulta, el cual abre una caja de dialogo, para seleccionar el tipo de consulta, ya sea para certificados en general, asociados a un embrque, a un inventario o a una factura. Esta consulta se puede realizar a partir de cualquier criterio de búsqueda, es decir podemos hacer una búsqueda de Certificados, ya sea por producto, por planta, por año del certificado, por numero de certificado, por fecha, por descripción y/o por nombre de elaboración, escribiendo en la forma de Certificados el texto a buscar. Este tipo de consultas se les conoce como Querys dinámicos. El resultado de la consulta se deposita en una de las tablas de consulta, dependiendo del tipo de consulta, seleccionada en la caj de dialogo.

Las consultas en el servidor se realizan a través de un SELECT a una vista, dependiendo del tipo de consulta.
Ver apendice Vistas A

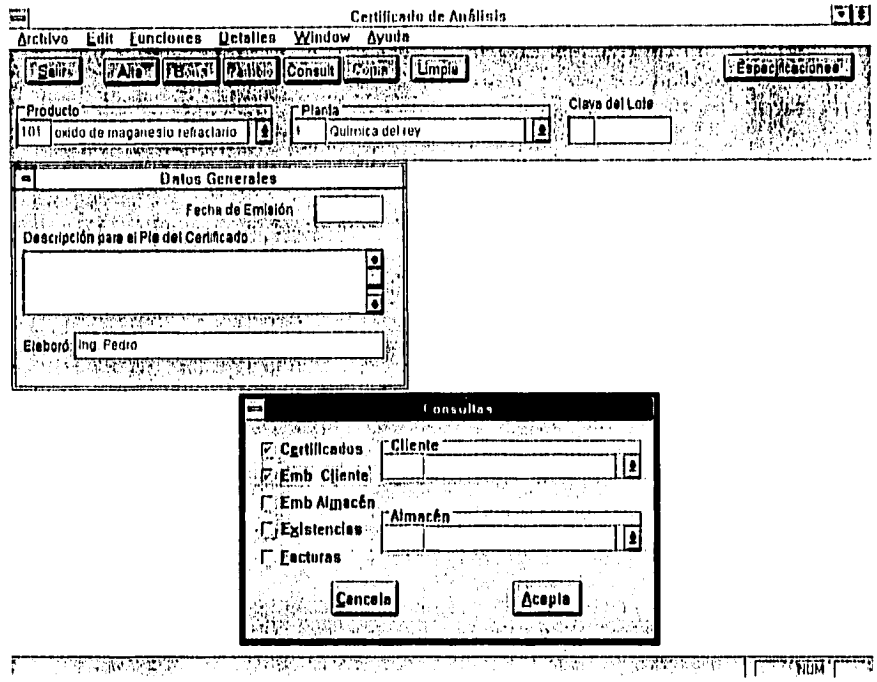
Una vez que se desplegaron todos los registros de la consulta, en la tabla, podemos seleccionar uno en especial, para trabajar con este, en la interface de certificados, esto mediante un DragDrop de un renglón seleccionado de la tabal, a la forma de datos generales o la barra de herramientas la aplicación.

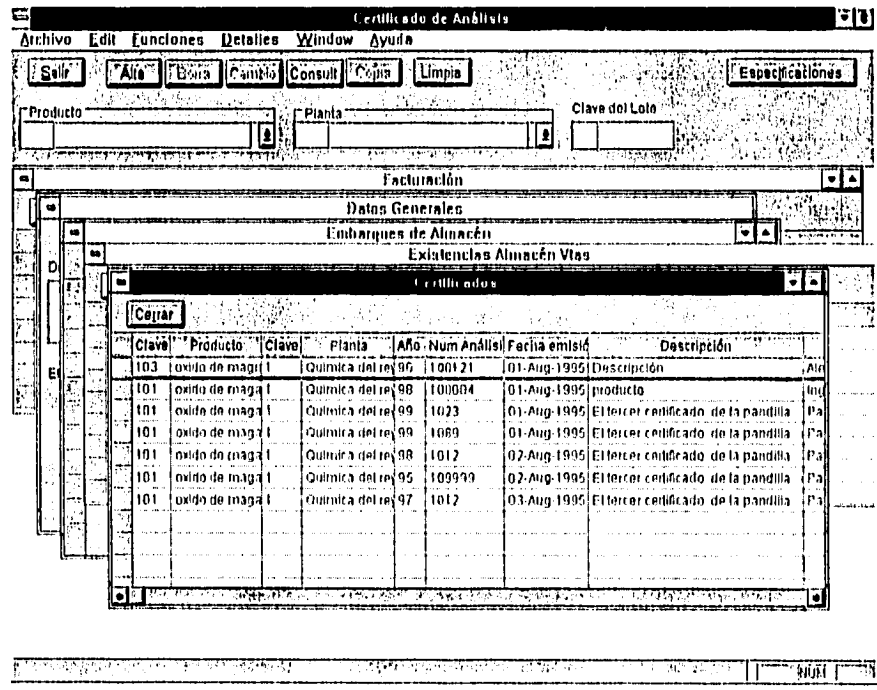
Cuando se realiza el Drag and Drop, a las zonas mencionadas, automaticamente se abre la tabla de especificaciones, en la cual se despliegan las respectivas al registro consultado.

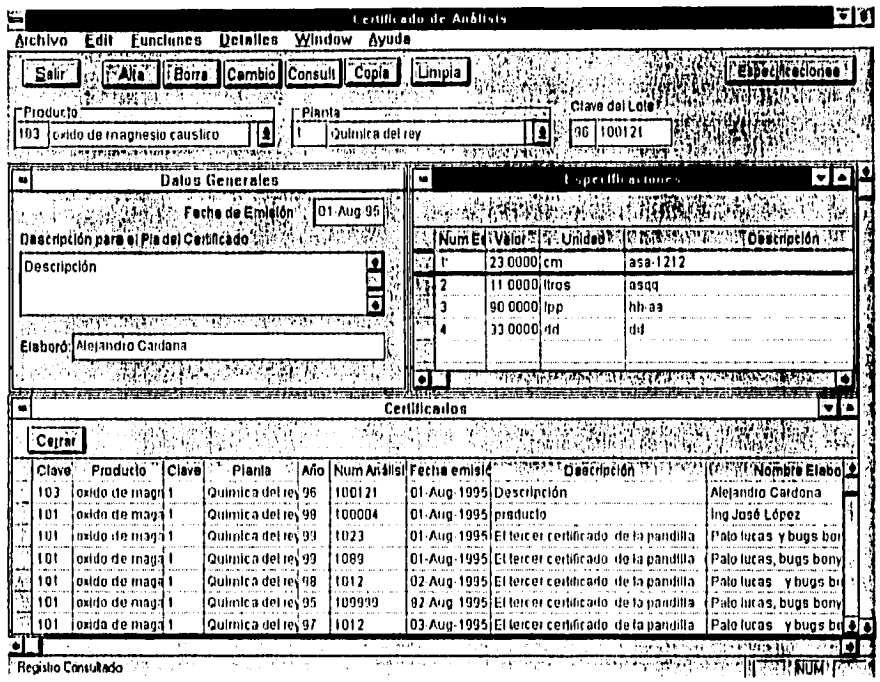
Estado De Consulta De La Forma



Una vez que consultamos un registro a este le podemos aplicar


Certificado de Análisis
Archivo Edit Funciones Detalles Window Ayuda
Salir
Alta
Borra
Cambio
Consult
Copia
Limpia
Especificaciones
Producto
101 oxido de magnesio refractario
Planta
1 Quimica del rey
Clave del Lote
Datos Generales
Fecha de Emisión
Descripción para el Pie del Certificado
Elaboró Ing Pedro
Consultas
Certificados
Emb Cliente
Emb Almacén
Existencias
Facturas
Cliente
Almacén
Cancela
Acepta
NUM

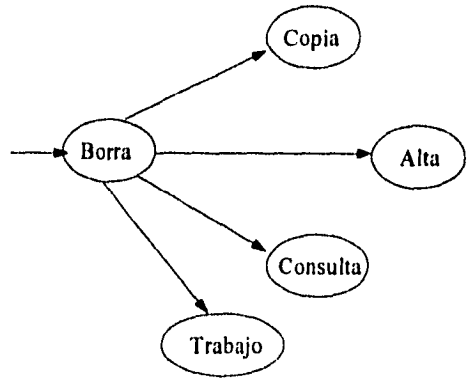

Certificado de Análisis
Archivo Edit Funciones Detalles Window Ayuda
Salir
Alta
Borra
Cambio
Consult
Copia
Limpia
Especificaciones
Producto
Planta
Clave del Lote
Facturación
Datos Generales
Embarques de Almacén
Existencias Almacén Vtas
Certificados
Cerrar
Clave Producto Clave Planta Año Num Análisi Fecha emisió Descripción
103 oxido de magn 1 Quimica del re 96 100121 01-Aug-1995 Descripción
101 oxido de magn 1 Quimica del re 98 100004 01-Aug-1995 producto
101 oxido de magn 1 Quimica del re 99 1023 01-Aug-1995 El tercer certificado de la pandilla
101 oxido de magn 1 Quimica del re 99 1069 01-Aug-1995 El tercer certificado de la pandilla
101 oxido de magn 1 Quimica del re 98 1012 02-Aug-1995 El tercer certificado de la pandilla
101 oxido de magn 1 Quimica del re 95 109999 02-Aug-1995 El tercer certificado de la pandilla
101 oxido de magn 1 Quimica del re 97 1012 03-Aug-1995 El tercer certificado de la pandilla
NUM


Certificado de Análisis
Archivo Edit Funciones Detalles Window Ayuda
Salir Alta Borra Cambio Consult Copia Limpia Especificaciones
Producto
103 oxido de magnesio caustico
Planta
1 Quimica del rey
Clave del Lote
96 100121
Datos Generales
Fecha de Emisión 01-Aug-95
Descripción para el Pie del Certificado
Descripción
Elaboró: Alejandro Cardona
Num Ed Valor Unidad Descripción
1 23.0000 cm asa-1212
2 11.0000 ltros asqq
3 90.0000 lpp hh-aa
4 33.0000 dd dd
Certificados
Cerrar
Clave Producto Clave Planta Año Num Análisi Fecha emisi Descripción Nombre Elabo
103 oxido de magn 1 Quimica del re 96 100121 01-Aug-1995 Descripción Alejandro Cardona
101 oxido de mag 1 Quimica del re 98 100004 01-Aug-1995 producto Ing José López
101 oxido de mag 1 Quimica del re 99 1023 01-Aug-1995 El tercer certificado de la pandilla Palo lucas y bugs bor
101 oxido de mag 1 Quimica del re 99 1089 01-Aug-1995 El tercer certificado de la pandilla Palo lucas, bugs bony
101 oxido de mag 1 Quimica del re 98 1012 02-Aug-1995 El tercer certificado de la pandilla Palo lucas y bugs b
101 oxido de mag 1 Quimica del re 95 109999 02-Aug-1995 El tercer certificado de la pandilla Palo lucas, bugs bony
101 oxido de mag 1 Quimica del re 97 1012 03-Aug-1995 El tercer certificado de la pandilla Palo lucas y bugs b
Registro Consultado
NUM

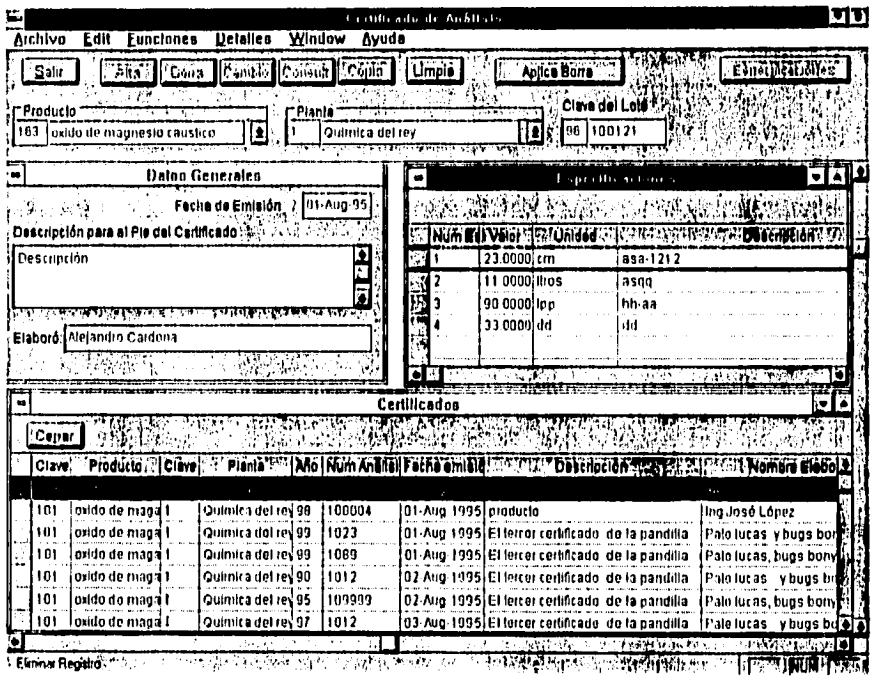
Botón de borra

El borrado o delete de un registro de la base de datos, se realiza a partir de la existencia de un registro, por lo cual esta acción se puede realizar después de una consulta. Una vez que la consulta de un registro fue válida, ahora podemos presionar el botón de borra, para eliminar ese registro de la tabla Pedido. Cuándo ejecutamos el botón de borra, aparece un botón con la leyenda de Aplica Borra, este tiene la función de aplicar físicamente el borrado en la base de datos, en el caso de que no se dese borrar, se presiona el botón de Limpia, el cual cancela la operación y vuelve a la forma a su estado original. Pero si se presiona AplicaBorra, el cliente, toma los campos identificadores de ese registro y arma un *DELETE*, el cual es enviado a la base de datos sobre el registro definido.

Estado De Borrado De La Forma



Funciones que se pueden realizar despues de un borrado



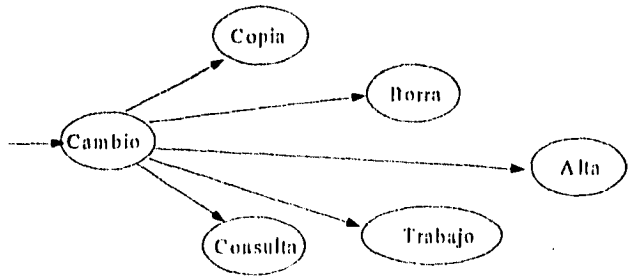
Botón de cambios

El cambio o modificación de un registro, al igual que el borrado, se ejecuta a partir de una consulta de un registro válido. Una vez que presionamos el botón de Cambio, la forma entra en un estado, en el cual algún campo de la forma, puede ser editado a exepción de los campos de la llave primaria. Cuando presionamos el botón de cambio, aparece el botón de AplicaCambio, el cual al presionarlo, el programa Cliente, verifica que realmente el registro haya sido modificado en algún de sus campos, si esto no se realiza el cliente despliega un mensaje, el cual indica que como no modifico ningún campo, esta acción no se ejecuta hasta que se cambie algún campo o presionemos el botón de Limpia, el cual cancela la acción en curso.
La transacción que se envía del cliente al servidor, puede ser formada por :

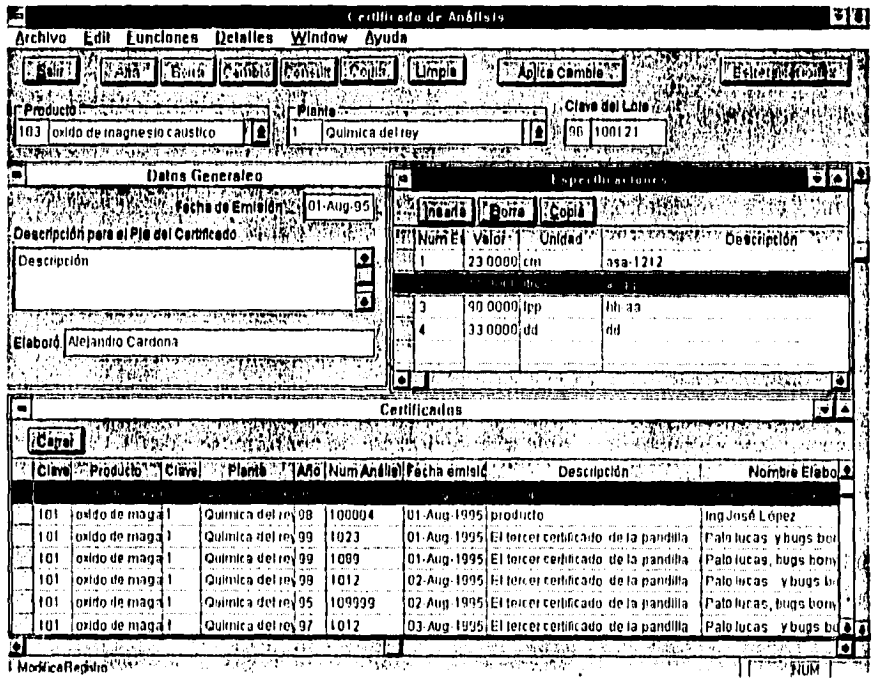
- Update algún campo de la tabla CER_ANALISIS
- Delete, Insert a un registro de la tabla ESPECIFICACION
- Update algún campo de la tabla ESPECIFICACION

Ya que cuando modificamos un Certificado de Análsis podemos actualizar un dato propio del mismo, o insertar, borrar o actualizar alguna de sus especificaciones.

Estado De Cambio De La Forma



Acciones a seguir despues de una actualización



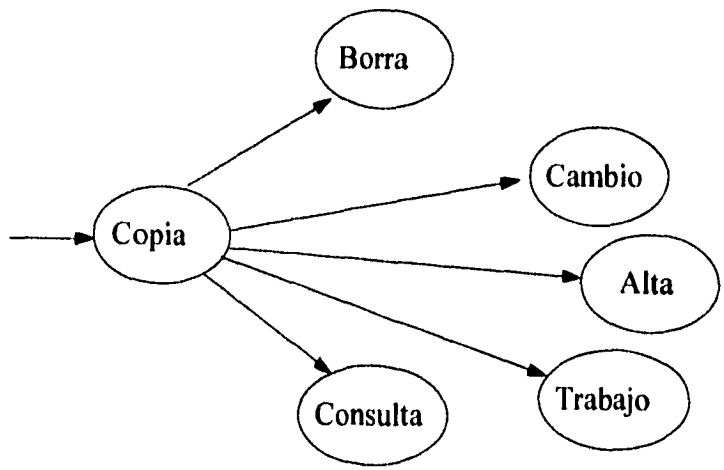
Botón de copia

La función de Copia, tiene el objetivo de hacer una copia de un registro de Certificado de Análsis, a partir de una consulta valida, pero siempre y cuando se hayan modificado los atributos identificadores del registro, ya que no pueden existir dos registros con la misma llave primaria. La función de copia es muy practica, ya que en ocasiones se desea dar de alta un registro, parecido a uno ya existente, por lo cual solo debemos modificar algunos atributos.

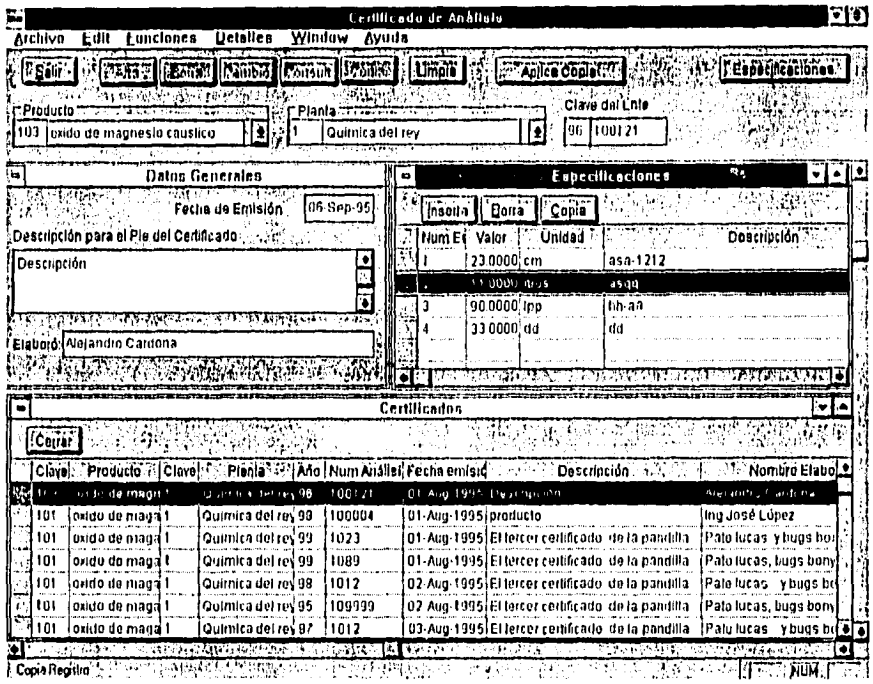
Cuando se presiona el botón de Copia, la forma espera recibir la modificación de alguno de sus campos de la PK necesaramente y opcionalmente cualquier otro, se presenta el botón de AplicaCopia, el cual cuando se presiona, la interface verifica que por lo menos se haya modificado algún atributo de la llave primaria, para poder ejecutar la copia. Posteriormente se ejecuta una transacción de insert a la tabla CER_ANALSIS y un conjunto de inserts a la tabla hija ESPECIFICACION.

Esta acción también puede ser cancelada si presionamos el botón de Limpia.

Estado De Copia De La Forma



Acciones a seguir despues de una Copia



Botón de Especificaciones

El botón cuando es presionado, abre una tabla la cual contiene los atributos para gestionar altas, bajas, cambios, copias y consultas de las especificaciones de un Certificado de Análsis.

Todas las demás aplicaciones para los procesos de nacional como de exportación, funcionan de igual manera, basadas en los mismos conceptos, tanto en el servidor como en el cliente. Los cual nos garantiza disminución del mantenimiento y estandarización en el proceso de desarrollo de aplicaciones.

Como logramos observar el cliente es el medio de explotación y de control de los datos y procesos del sistema, ya que la interface GUI, nos proporciona :

- Despliegue de los datos en interface gráfica
- Control de procesos, por medio de botones y menús
- Alta, baja, cambios y consultas de los datos en el servidor.

Mientras que la base de datos, solo se encarga de mantener los datos en consistencia y seguridad, para que el cualquier cliente, sea capaz de explotarla, pero principalmente su cliente el cual se apega a las necesidades del usuario final.

El cliente proporciona una serie de estatutos SQL a través de la red hacia el servidor, para poder explotar las características del servidor, tanto en datos como en procesos.

Por lo cual podemos observar que tanto el cliente y el servidor, son elementos autónomos e independientes en el funcionamiento, lo cual nos proporciona grandes ventajas de descentralización de procesos, lo cual hace más eficiente los sistemas.

RESUMEN

En este capitulo se presento como el Sistema de Ventas Químicos quedo implementado, tanto en el Cliente como el Servidor. El cliente se presenta la interface de despliegue y de control de los procesos, para poder llevar a cabo los requerimientos de los usuarios. Esta interface gráfica para cada función del sistema, funciona bajo un prototipo estándar, el cual nos presenta los procesos de altas, bajas, cambios, consultas y copias de la función en cuestión. Estas funciones disparan clausuras SQL y store procedures al servidor, por medio de la red, mientras que el servidor da una respuesta a esas peticiones.
El servidor compuesto por una estructura relacional, la cual esta soportada bajo la integridad de datos y referencial, para garantizar la seguridad y consistencia de la información.

El sistema de ventas químico constituido por dos aplicaciones principales, mercado nacional y exportación, se encargan del proceso administrativo de ventas, para ventas a nivel nacional e internacional.

Ventas Nacional

1. Embarques de planta
2. Llegada de embarques en bodega
3. Pedidos nacional.
4. Remisiones de Bodega .
5. Solicitud de embarques de planta para bodega .
6. Lotes en Bodega de material dañado en bodega.
7. Facturación de Pedidos.
8. Notas de Cargo y Crédito.
9. Póliza de Ventas.
10. Reportes.

Ventas De Exportación

1. Pedido de ventas exportación.
2. Cartas de Embarques Químicos (CEQ).
3. Llegada de material a bodegas en el extranjero.
4. Inventarios de bodegas propias y a consignación

5. Orden de Ventas Químicos (OVQ).

6. Orden de Muestra Químicos(OMQ).

7. Facturación.

8. Instrucción de Cobranzas.

9. Notas de cargo y crédito.

10.Gastos de Mercado..

11.Póliza de Ventas.

12.Comisiones.

13. Reportes.

De todos estos procesos, se presento a fondo el funcionamiento de Pedido de ventas Nacional, del cual se hablo de las tablas relaciónales que intervienen en este proceso, sus mecanismos de integridad de datos y de integridad referencial, mientras que por el cliente, se mostraron las interfaces GUI y los procesos de control y de despliegue entre el cliente y el servidor.

De esta manera como se dijo al principio del resumen, las demás aplicaciones del sistema, funcionan en forma parecida para el desarrollo de los requerimientos del cliente final.

CONCLUSIONES

El desarrollo de mejores sistemas de información en las organizaciones, ha provocado que estas sean más eficientes y productivas, reduciendo costos y mejorando procesos, por lo que son de gran premisa en las empresas, el desarrollo de los mismos.

A lo largo de la historia del desarrollo de los sistemas , la tecnología en computo ha evolucionado desde las maquinas electromecánicas hasta los equipos de hoy en día. Esta evolución no solo se ha dado en los equipos, sino también en el desarrollo de las comunicaciones y el software.

Todo esto ha provocado el cambio en los paradigmas en el desarrollo y en la arquitectura de los sistemas, ya que anteriormente las empresas desarrollaban sistemas de tipo centralizados, los cuales se basaban en equipos mainframe, en lenguajes de tipo procedural y base de datos jerárquicas o de red, en la actualidad la tecnología nos permite la implementación de nuevos sistemas, basados en nuevos equipos de computo, nuevo software y comunicaciones, como es el caso de la Arquitectura Cliente/Servidor.

Los sistemas Cliente/Servidor, se basan en una filosofía de trabajo, la cual se adecua a la tecnología de hardware y software actual, y a las necesidades de información de las empresas. Ya que esta filosofía se basa en el trabajo distribuido a través de una red de comunicaciones, entre dos elementos, el Cliente y el Servidor. El primero se encarga del control de las interfaces y procesos, mientras que el segundo realiza el almacenamiento de datos, así como las transacciones de alto volumen de información.

El nuevo paradigma de desarrollo de información, hace que la forma de desarrollo y arquitectura de los sistemas cambie, de como se hacia anteriormente, ya que ahora:

- La filosofía de trabajo, cambia de un ambiente centralizado a uno distribuido, en cuanto a proceso, tareas, seguridad, mantenimiento y trabajo.
- La metodología de desarrollo, ahora el análisis y diseño de los sistemas, se centra en dos elementos, en el cliente y en el servidor, como elementos autónomos pero coordinados.
- El hardware y software, los equipos de computo son más pequeños, pero más poderosos en capacidad, procesamiento y almacenamiento, mientras que el software de desarrollo es más amigable y más eficiente en el procesamiento de datos.
- Se crea una infraestructura de objetos, estándares, datos, aplicaciones, etc. para el desarrollo y programación de nuevos sistemas.
- El mantenimiento de los sistemas se minimiza en un gran porcentaje, debido a las estructuras de programación, ya que anteriormente en el ciclo de vida de los sistemas, el mantenimiento se lleva el mayor porcentaje.
- Los sistemas ahorra son totalmente abiertos, por lo que se pueden integrar con otros sistemas de su mismo tipo.
- La interface con el usuario es más amigable
- La arquitectura de los componentes se distribuye físicamente, lo cual descentraliza actividades de mantenimiento y operación.
- La seguridad de los sistemas se vuelve más fácil de implementar y mas eficiente.

Estos cambios que provee la nueva tecnología, son orientados hacia la mejora de las organización y sus procesos, tanto en las áreas usuarias como en la gerencia de desarrollo. Por lo que la aportación de esta tesis en el ámbito empresarial, es la introducción y aplicación de esta tecnología, en una empresa real, como lo es Servicios Industriales Peñoles, S.A., en su División Químicos. Ya que esta empresa contaba con un sistema de ventas en una plataforma centralizada. El cual tenía dos problemas principalmente; el sistema no satisfacía todas las necesidades y requerimientos de las áreas usuarias y el mantenimiento del mismo era costosos y llevaba mucho tiempo. Por lo que era necesario crear un nuevo sistema, el cual solucionará esos problemas y fuera más eficiente. Es por esto que la presente tesis se muestra la arquitectura cliente/servidor como nueva plataforma de desarrollo del Sistema de Ventas Químicos, así mismo se presenta una estrategia para la implementación del sistema, tanto a nivel de arquitectura de procesos y a nivel de metodología para su desarrollo.

La arquitectura del nuevo sistema se definió como cliente/servidor, la cual se basa como sabemos en dos elementos, el cliente y el servidor, cada uno de estos tienen un funcionamiento definido, por lo cual el servidor basa su arquitectura en el modelo relacional de datos y en un modelo procedural para la ejecución de los procesos, mientras que por parte del cliente, este debe de contar con una interface gráfica de despliegue y de control de los proceso, las cuales deben de estar orientadas a objetos. Con esto obtenemos una arquitectura cliente/servidor, basada en un servidor de datos de tipo relacional y en un cliente orientado a objetos.

Una vez definida la arquitectura de procesos y datos, del sistema, se presenta el método de análisis y diseño, para poder implementar esta arquitectura.

Para el análisis del sistema, se crearon tres modelos, los cuales representan los requerimientos de datos y procesos del usuario, estos modelos son: modelo entidad/objeto, modelo de estados y modelo funcional, estos tres modelos se interrelacionan en uno, el cual es el modelo de análisis. En el modelo entidad/objetos, se definen las entidades o objetos que intervienen en el sistema y que son de importancia para el mismo, este modelo se define en un diagrama entidad relación y en la descripción de los atributos de cada entidad, el modelo de estados se basa en los estados de las entidades u objetos del sistema, resultado de la aplicación de ciertos eventos, este se representa en un diagrama de estados de cada entidad, por último el modelo de procesos, nos muestra los procesos y datos que intervienen en un sistema, bajo un diagrama de flujo de datos. Todos estos diagramas se integran desde un diagrama de descomposición, el cual integra el modelo de análisis.

Una vez que se tienen los modelos de análisis, el siguiente paso es hacer el diseño, para cliente y para el servidor a partir de los modelos de análisis.

Para el diseño del servidor, basado en el modelo entidad/objeto, se construye la estructura relacional de la base de datos, en cuanto a tablas, campos, tipos de datos, relaciones, llaves primarias e índices, con esto podemos tener una base de datos definida para nuestro sistema, la cual por medio de tipos de datos, defaults y reglas, los datos tendrán integridad, mientras que a través de triggers y constraints garantizaremos la integridad referencial, estos elementos son definidos por medio de las reglas de negocio, para que reflejen la realidad de la información. Una vez que se tienen estos elementos, contamos con una back-end, como un sistema independiente, explotable, seguro e integro.

El diseño del cliente, se basa en la creación de un prototipo, el cual será la base y guía para la programación de las demás aplicaciones del sistema, ya que vía este prototipo se crearan clases, estándares y procedimientos para codificación de las demás formas del sistema junto con el modelo de eventos y de procesos de análisis, podremos contar con una interface de despliegue de datos y control de los procesos del sistema.

La programación del cliente a partir del prototipo de los modelos de análisis, se realiza en dos fases: la construcción de la interface de presentación por medio de objetos y la programación del funcionamiento y procesos que deba de llevar esa interface, en l a cual se envían mensajes, los cuales ejecutan procesos en el cliente o en el servidor, en el caso que sea al servidor , el cliente envían store procedures o clausuaras SQL, al servidor y este procesa el mensaje y envía una respuesta. Las peticiones del cliente al servidor, pueden ser de datos o de procesos.

El proceso de cambio del sistema centralizado a uno nuevo, en Servicios Industriales Peñoles, S.A., se dio debido dos problemas principalmente; el sistema centralizado no contemplaba todos los requerimientos de las áreas usuarias y el mantenimiento del sistema era constante y de costo elevado, esto dio como resultado la necesidad de hacer un nuevo sistema, el cual solucionara los problemas anteriores y además que en cuestión de desarrollo y de producción, fuera eficiente y productivo.

Por lo que concluimos de esta investigación que el desarrollo de un Sistema Cliente/Servidor, es un elemento de cambio de mejora, tanto para la Gerencia de Informática y para las Areas Usuarias. En el análisis y diseño y en la producción del Sistema Cliente/Servidor.

Beneficios para la Gerencia de Sistemas en el Desarrollo de Sistemas

- Hace más eficiente y productivo el desarrollo de los sistemas, ya que se crea una infraestructura de estándares, clases de objetos, aplicaciones, librerías, procedimientos para hacer más eficientes nuevos desarrollos.
- Los sistemas cliente/servidor, hace que cada desarrollo sea un sistema abierto, hacia la misma empresa y a fuera de ella, integrando cada vez más las redes internacionales de bases de datos.
- La creación de bases de datos relaciónales, hace que cualquier cliente pueda explotar la información
- Reduce el mantenimiento de los sistemas, debido al tipo de estructura del código que fue codificado.
- Hace más sencillo el mantenimiento de los sistemas, debido a que el código es orientado a objetos y es modular.

Beneficios para el Area Usuaria, en la producción del Sistema Cliente/Servidor

- Proporciona productos a todos los niveles de las organizaciones: operativo; procesos, gerencial; procesos e información y directivo; información.
- Hace más eficientes los procesos de información, debido al aumento en la productividad de automatizar cada vez más y mejor los proceso.
- Proporciona mayor tiempo de respuesta, gracias a la velocidad de los nuevos equipos.
- Crea un interface más abierta y amigable, por medio de interfaces GUI.
- Permite una mayor explotación de los proceso y de la información, gracias a la interface gráfica.

Todos estos beneficios que se muestran, da por ende que la empresa eleve su productividad, ya que mejora sus proceso y eso propicia que la empresa ahora piense en otras cosas o procesos , los cuales también son susceptibles de mejorar, y así se crea un circulo de mejora continua, para llegar a ser una empresa de calidad.

APENDICE A
ESQUEMA DE LA BASE DE DATOS

ESQUEMA DE LA BASE DE DATOS

Tablas, tipos de datos, restricciones (constraints), indices, llaves primarias (primary key) y laves foránes (foreing key)

```
CREATE TABLE CER_ANALISIS
      (pta_cve_pta              char(3) NOT NULL,
       pdt_cve_prd              char(4) NOT NULL,
   cea_anio_cer       tinyint NOT NULL
       CONSTRAINT reg_positivo2768 CHECK ( cea_anio_cer > 0),
       /*Integridad de datos (reglas del negocio)*/
       cea_num_analisis         char(8) NOT NULL,
       cea_fec_elaboracion_certf smalldatetime NOT NULL,
       cea_nom_elaboro          char(40) NOT NULL,
       cea_dsc_para_el_pie_certfic char(100) NOT NULL,
       CONSTRAINT PK_CER_ANALISIS       PRIMARY KEY  NONCLUSTERED
       (pta_cve_pta, pdt_cve_prd, cea_anio_cer, cea_num_analisis)
             WITH FILLFACTOR = 75,
       /*Indice clustered identificador*/
       CONSTRAINT descrito_en_CER_ANALISIS       FOREIGN KEY
(pdt_cve_prd)
              REFERENCES PRODUCTO,
       /*Integridad referencial */
       CONSTRAINT emitio_CER_ANALISIS       FOREIGN KEY
(pta_cve_pta)
              REFERENCES PLANTA
)
go
CREATE TABLE ESPECIFICACION
      (pta_cve_pta              char(3) NOT NULL,
       pdt_cve_prd              char(4) NOT NULL,
       cea_anio_cer             tinyint NOT NULL
       CONSTRAINT reg_positivo2769 CHECK ( cea_anio_cer > 0),
       cea_num_analisis         char(8) NOT NULL,
       esp_num_esp              tinyint NOT NULL
       CONSTRAINT reg_positivo2770 CHECK ( esp_num_esp > 0),
       esp_dsc_esp              char(100) NOT NULL,
       esp_unidad_dimensional   char(10) NOT NULL,
       esp_valor_obtenido       float NOT NULL,
       CONSTRAINT PK_ESPECIFICACION       PRIMARY KEY
NONCLUSTERED (pta_cve_pta, pdt_cve_prd, cea_anio_cer,
cea_num_analisis, esp_num_esp)
             WITH FILLFACTOR = 75,
```

```
      CONSTRAINT emplea_ESPECIFICACION      FOREIGN KEY
(pta_cve_pta, pdt_cve_prd, cea_anio_cer, cea_num_analisis)
            REFERENCES CER_ANALISIS
)
go

CREATE TABLE COMPANIA
     (cia_cve_compania         smallint NOT NULL,
      cia_raz_soc              varchar(100) NOT NULL,
      cia_domi_fiscal          varchar(100) NOT NULL,
      cia_domi_ofna_administrativ varchar(100) NOT NULL,
      cia_rfc                  char(15) NOT NULL,
      cia_lgr_origen           varchar(40) NOT NULL,
      cia_tel1_ofna_admva      varchar(20) NOT NULL,
      cia_tel2_ofna_admva      varchar(20) NULL,
      cia_tel3_ofna_admva      varchar(20) NULL,
      CONSTRAINT PK_COMPANIA      PRIMARY KEY (cia_cve_compania)
            WITH FILLFACTOR = 100
)
go

CREATE TABLE PRODUCTO
      (pdt_cve_prd              char(4) NOT NULL,
       cia_cve_compania         smallint NOT NULL,
       pdt_nom_espanol          varchar(60) NOT NULL,
       pdt_nom_ingles           varchar(60) NOT NULL,
       pdt_dsc_proceso_elaboracion varchar(255) NULL,
       CONSTRAINT PK_PRODUCTO       PRIMARY KEY (pdt_cve_prd)
            WITH FILLFACTOR = 100,
       CONSTRAINT usada_en_PRODUCTO       FOREIGN KEY
(cia_cve_compania)
            REFERENCES COMPANIA
)
go

CREATE TABLE PLANTA
      (pta_cve_pta              char(3) NOT NULL,
       pta_raz_soc_pta          char(100) NOT NULL,
       pta_domi_fiscal          char(50) NOT NULL,
       cia_cve_compania         smallint NOT NULL,
       CONSTRAINT PK_PLANTA       PRIMARY KEY (pta_cve_pta)
            WITH FILLFACTOR = 100,
       CONSTRAINT usada_en_PLANTA       FOREIGN KEY
(cia_cve_compania)
            REFERENCES COMPANIA
)
go
```

```
CREATE TABLE EMBARQUE_CLIENTE
      (cia_cve_compania          smallint NOT NULL,
       pta_cve_pta               char(3) NOT NULL,
       ide_fec_embarque          smalldatetime NOT NULL,
       cte_cve_cte               int NOT NULL,
       pdt_cve_prd               char(4) NOT NULL,
       emp_cve_emp               char(2) NOT NULL,
       ect_placa_camion_num_carro char(10) NOT NULL,
       pdn_num_anio              tinyint NULL,
       gtv_cve_gte_vta           char(3) NULL,
       pdn_num_ped               smallint NULL,
       ect_fec_estimada_arrivo   smalldatetime NOT NULL,
       ect_fec_fabricacion       smalldatetime NOT NULL,
       ect_cve_lote              char(14) NOT NULL,
       ect_tipo_equipo           char(20) NOT NULL,
       ect_peso_bruto_embarcado float NOT NULL
       CONSTRAINT reg_positivo2771 CHECK (
                 ect_peso_bruto_embarcado > 0),
       ect_peso_neto_embarcado  float NOT NULL
       CONSTRAINT reg_positivo2772  CHECK (
                ect_peso_neto_embarcado > 0),
       ect_cnt_emp_embarcado     int NULL
       CONSTRAINT reg_positivo2773  CHECK (
                  ect_cnt_emp_embarcado > 0),
       ect_tipo_fac              char(12) NULL
       CONSTRAINT typ_facturacion71
                CHECK ( ect_tipo_fac in ("UNICA","PERIODO",
                                         CONSIGNACION")),
       ect_sts              char(14) DEFAULT "XFACTURAR" NOT NULL
       CONSTRAINT typ_ect_sts40
              CHECK (ect_sts in ("XFACTURAR","FACTURADO PARC",
                                         "FACTURADO TOT" )),
       cea_anio_cer              tinyint NULL,
       cea_num_analisis          char(8) NULL,
       CONSTRAINT PK_EMBARQUE_CLIENTE       PRIMARY KEY
NONCLUSTERED
          (cia_cve_compania, pta_cve_pta, ide_fec_embarque,
           cte_cve_cte, pdt_cve_prd,
            emp_cve_emp,ect_placa_camion_num_carro)
              WITH FILLFACTOR = 75,
       CONSTRAINT tiene_EMBARQUE_CLIENTE   FOREIGN KEY
(cia_cve_compania, pdn_num_anio, gtv_cve_gte_vta, pdn_num_ped,
 pta_cve_pta)
                      REFERENCES PEDIDO_PTA_NAL,
       CONSTRAINT describe_EMBARQUE_CLIENTE FOREIGN KEY
(pdt_cve_prd)
                      REFERENCES PRODUCTO,
```

```
      CONSTRAINT envio_EMBARQUE_CLIENTE    FOREIGN KEY
(cia_cve_compania, pta_cve_pta, ide_fec_embarque)
                        REFERENCES INF_DIARIO_EMBARQ,
      CONSTRAINT describe_EMB_CTE          FOREIGN KEY
(pta_cve_pta, pdt_cve_prd, cea_anio_cer, cea_num_analisis)
                        REFERENCES CER_ANALISIS,
      CONSTRAINT recibe_EMBARQUE_CLIENTE   FOREIGN KEY
(cte_cve_cte)
                        REFERENCES CLIENTE,
      CONSTRAINT usado_en_EMBARQUE_CLIENTE FOREIGN KEY
(emp_cve_emp)
                        REFERENCES EMPAQUE
)
go

CREATE TABLE FACTURA_NAL
       (cia_cve_compania          smallint NOT NULL,
        lgf_serie_fac             char(3) NOT NULL,
        fan_num_fac               int NOT NULL
                                  CONSTRAINT reg_positivo3191
                                         CHECK ( fan_num_fac > 0),
        pdt_cve_prd               char(4) NOT NULL,
        pta_cve_pta               char(3) NOT NULL,
        cte_cve_cte               int NULL
                                  CONSTRAINT reg_positivo3192
                                         CHECK ( cte_cve_cte > 0),
        emp_cve_emp               char(2) NULL,
        fan_dsc_producto          varchar(255) NOT NULL,
        fan_precio_unitario       money NULL
                                  CONSTRAINT reg_positivo3193
                                         CHECK (
                                         fan_precio_unitario > 0),
        tpc_cve_moneda            char(3) DEFAULT "MN" NOT NULL,
        tpc_fec_cotiz             smalldatetime NOT NULL,
        cdp_cve_cnd_pago          char(3) NOT NULL,
        tvt_cve_termino_vta       char(3) NOT NULL
                                  CONSTRAINT typ_termv69
                                         CHECK (
tvt_cve_termino_vta in ("CFR",
"C&F","CIF","DAF","FAS","FOB","LAB","FOT","EXW")),
        fan_lgr_term_vta          varchar(30) NOT NULL,
        fan_fec_fac               smalldatetime NOT NULL,
        fan_fec_embarque          smalldatetime NOT NULL,
        fan_tipo_fac              char(12) DEFAULT "UNICA" NOT NULL
                                  CONSTRAINT typ_facturacion87
                                        CHECK ( fan_tipo_fac in
              ("UNICA","PERIODO",CONSIGNACION")),
        fan_peso_fac              float NULL
                                  CONSTRAINT reg_positivo3194
```

```
                                CHECK ( fan_peso_fac > 0),
        fan_peso_total_ped_x_cte float NULL
                                CONSTRAINT reg_positivo3195
                                CHECK (
                        fan_peso_total_ped_x_cte > 0),
        fan_total_peso_bruto    float NULL
                                CONSTRAINT reg_positivo3196
                               CHECK ( fan_total_peso_bruto > 0),
        fan_total_peso_neto     float NULL
                                CONSTRAINT reg_positivo3197
                                CHECK ( fan_total_peso_neto > 0),
        fan_total_cnt_emp       int NULL
                                CONSTRAINT reg_positivo3198
                                CHECK ( fan_total_cnt_emp > 0),
        fan_imp_neto_fac        money NOT NULL
                                CONSTRAINT reg_positivo3199
                               CHECK ( fan_imp_neto_fac > 0),
        fan_imp_iva_fac         money NOT NULL
                                CONSTRAINT reg_positivo3200
                                CHECK ( fan_imp_iva_fac > 0),
        fan_imp_total_fac       money NOT NULL
                                CONSTRAINT reg_positivo3201
                                CHECK ( fan_imp_total_fac > 0),
        cea_anio_cer            tinyint NULL
                                CONSTRAINT reg_positivo3202
                                CHECK ( cea_anio_cer > 0),
        cea_num_analisis        char(8) NULL,
        fan_sts                 char(11) DEFAULT "PROVISIONAL"
                                CONSTRAINT typ_fan_sts23
                                CHECK ( fan_sts in
         ("PROVISIONAL","DEFINITIVA","CANCELADA")),
        fan_fec_sts             smalldatetime NOT NULL,
        CONSTRAINT PK_FACTURA_NAL       PRIMARY KEY  NONCLUSTERED
(cia_cve_compania, lgf_serie_fac, fan_num_fac)
              WITH FILLFACTOR = 75,
        CONSTRAINT describe_FACTURA_NAL       FOREIGN KEY
(pta_cve_pta, pdt_cve_prd, cea_anio_cer, cea_num_analisis)
                REFERENCES CER_ANALISIS,
        CONSTRAINT describe_a_FACTURA_NAL       FOREIGN KEY
(cdp_cve_cnd_pago)
                REFERENCES CONDICION_PAGO,
        CONSTRAINT es_usado_FACTURA_NAL       FOREIGN KEY
(tvt_cve_termino_vta)
                REFERENCES TERMINO_VENTA,
        CONSTRAINT se_usa_en_FACTURA_NAL       FOREIGN KEY
(tpc_cve_moneda, tpc_fec_cotiz)
                REFERENCES TIPO_DE_CAMBIO,
        CONSTRAINT describe_algunas_FACTURA_NAL       FOREIGN KEY
(pdt_cve_prd)
```

```
                REFERENCES PRODUCTO,
        CONSTRAINT emisor_de_FACTURA_NAL        FOREIGN KEY
(lgf_serie_fac)
                REFERENCES LUGAR_FACTURACION,
        CONSTRAINT tiene_FACTURA_NAL        FOREIGN KEY
(cte_cve_cte)
                REFERENCES CLIENTE,
        CONSTRAINT usado_en_FACTURA_NAL        FOREIGN KEY
(emp_cve_emp)
                REFERENCES EMPAQUE,
        CONSTRAINT emite_FACTURA_NAL        FOREIGN KEY
(cia_cve_compania)
                REFERENCES COMPANIA
 )
go

CREATE TABLE EMB_ALM
       (cia_cve_compania          smallint NOT NULL,
        pta_cve_pta               char(3) NOT NULL,
        ide_fec_embarque          smalldatetime NOT NULL,
        alm_cve                   char(3) NOT NULL,
        pdt_cve_prd               char(4) NOT NULL,
        emp_cve_emp               char(2) NOT NULL,
        ebd_placa_camion_num_carro char(18) NOT NULL,
        ebd_fec_fabricacion       smalldatetime NULL,
        ebd_fec_estimada_arrivo   smalldatetime NOT NULL,
        ebd_peso_neto_embarcado   float NOT NULL
        CONSTRAINT reg_positivo3235
CHECK ( ebd_peso_neto_embarcado > 0),
        ebd_peso_bruto_embarcado float NOT NULL
        CONSTRAINT reg_positivo3236
CHECK ( ebd_peso_bruto_embarcado > 0),
        ebd_cnt_emp_embarcado     int NULL
        CONSTRAINT reg_positivo3237
CHECK ( ebd_cnt_emp_embarcado > 0),
        ebd_cve_lote              char(14) NOT NULL,
        ebd_tipo_equipo           char(20) NOT NULL,
        cea_anio_cer              tinyint NULL
       CONSTRAINT reg_positivo3238
CHECK ( cea_anio_cer > 0),
        cea_num_analisis          char(8) NULL,
        CONSTRAINT PK_EMB_ALM        PRIMARY KEY  NONCLUSTERED
(cia_cve_compania, pta_cve_pta, ide_fec_embarque, alm_cve,
pdt_cve_prd, emp_cve_emp, ebd_placa_camion_num_carro)
               WITH FILLFACTOR = 75,
        CONSTRAINT espera_EMB_ALM        FOREIGN KEY  (alm_cve)
                REFERENCES ALMACEN,
        CONSTRAINT tiene_EMB_ALM        FOREIGN KEY  (pdt_cve_prd)
                REFERENCES PRODUCTO,
```

```
        CONSTRAINT emite_EMB_ALM        FOREIGN KEY
(cia_cve_compania, pta_cve_pta, ide_fec_embarque)
                REFERENCES INF_DIARIO_EMBARQ,
        CONSTRAINT describe_EMB_ALM        FOREIGN KEY
(pta_cve_pta, pdt_cve_prd, cea_anio_cer, cea_num_analisis)
                REFERENCES CER_ANALISIS,
        CONSTRAINT usado_en_EMB_ALM        FOREIGN KEY
(emp_cve_emp)
                REFERENCES EMPAQUE
 )
go
CREATE TABLE INV_LOTE_ALM
       (alm_cve                    char(3) NOT NULL,
        pdt_cve_prd                char(4) NOT NULL,
        emp_cve_emp                char(2) NOT NULL,
        cia_cve_compania           smallint NOT NULL,
        pta_cve_pta                char(3) NOT NULL,
        ivb_fec_embarque_pta       smalldatetime NOT NULL,
        ivb_placa_camion_num_carro char(10) NOT NULL,
        ivb_fec_entrada_bodega     smalldatetime NOT NULL,
        ivb_existencia_neta_tm     float NOT NULL
        CONSTRAINT reg_positivo3307
        CHECK ( ivb_existencia_neta_tm > 0),
        ivb_existencia_bruta_tm    float NOT NULL
        CONSTRAINT reg_positivo3308
        CHECK ( ivb_existencia_bruta_tm > 0),
        ivb_existencia_cnt_emp     int NULL
        CONSTRAINT reg_positivo3309
        CHECK ( ivb_existencia_cnt_emp > 0),
        cea_anio_cer               tinyint NULL
        CONSTRAINT reg_positivo3310
        CHECK ( cea_anio_cer > 0),
        cea_num_analisis           char(8) NULL,
        CONSTRAINT PK_INV_LOTE_ALM        PRIMARY KEY (alm_cve,
pdt_cve_prd, emp_cve_emp, cia_cve_compania, pta_cve_pta,
ivb_fec_embarque_pta, ivb_placa_camion_num_carro)
                WITH FILLFACTOR = 80,
        CONSTRAINT usada_en_INV_LOTE_ALM        FOREIGN KEY
(cia_cve_compania)
                REFERENCES COMPANIA,
        CONSTRAINT describe_INV_LOTE_ALM        FOREIGN KEY
(pta_cve_pta)
                REFERENCES PLANTA,
        CONSTRAINT tiene_INF_LOTE_ALM        FOREIGN KEY  (alm_cve,
pdt_cve_prd, emp_cve_emp)
                REFERENCES INV_PRODUCTO_ALM,
        CONSTRAINT describe_LOTE_ALMACEN        FOREIGN KEY
(pta_cve_pta, pdt_cve_prd, cea_anio_cer, cea_num_analisis)
                REFERENCES CER_ANALISIS)
```

Disparadores (triggers) Integridad referencial

```
create trigger COMPANIA_UPD on COMPANIA for UPDATE as
                                                    /* Mon Sep 04
16:32:54 1995 */
begin
declare  @numrows    int,
         @cambio     float,
         @actual     float,
         @compromiso float,
         @errno   int,
         @errmsg  varchar(255)
select @numrows = @@rowcount
if @numrows = 0
  return

/* PRODUCTO        Mon Sep 04 16:32:53 1995 */
if exists ( select * from  deleted, PRODUCTO
              where PRODUCTO.cia_cve_compania =
deleted.cia_cve_compania )
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: COMPANIA por ser usado en
PRODUCTO.'
  goto error
  end

/* PLANTA        Mon Sep 04 16:32:54 1995 */
if exists ( select * from  deleted, PLANTA
              where PLANTA.cia_cve_compania =
deleted.cia_cve_compania )
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: COMPANIA por ser usado en
PLANTA.'
  goto error
  end

create trigger PRODUCTO_UPD on PRODUCTO for UPDATE as
                                                    /* Mon Sep 04
16:32:54 1995 */
begin
declare  @numrows    int,
         @cambio     float,
         @actual     float,
         @compromiso float,
         @errno   int,
         @errmsg  varchar(255)
select @numrows = @@rowcount
```

```
if @numrows = 0
  return

/* CER_ANALISIS        Mon Sep 04 16:32:54 1995 */
if exists ( select * from  deleted, CER_ANALISIS
            where CER_ANALISIS.pdt_cve_prd = deleted.pdt_cve_prd
)
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: PRODUCTO por ser usado en
CER_ANALISIS.'
  goto error
  end

create trigger PLANTA_UPD on PLANTA for UPDATE as
                                                    /* Mon Sep 04
16:32:54 1995 */
begin
declare  @numrows    int,
         @cambio     float,
         @actual     float,
         @compromiso float,
         @errno   int,
         @errmsg  varchar(255)
select @numrows = @@rowcount
if @numrows = 0
  return
                                                    /* CER_ANALISIS
Mon Sep 04 16:32:54 1995 */
if exists ( select * from  deleted, CER_ANALISIS
            where CER_ANALISIS.pta_cve_pta = deleted.pta_cve_pta
)
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: PLANTA por ser usado en
CER_ANALISIS.'
  goto error
  end


create trigger CER_ANALISIS_DEL on CER_ANALISIS for DELETE as
                                                    /* Mon Sep 04
16:32:54 1995 */
begin
declare  @numrows int,
         @validcnt int,
         @errno   int,
         @errmsg  varchar(255)
select @numrows = @@rowcount
```

```
if @numrows = 0
  return
                                                    /* ESPECIFICACION
Mon Sep 04

16:32:54 1995 */

/*Borrado en Casacada de Certificado a Especificaciones*/
    delete ESPECIFICACION
      from ESPECIFICACION,deleted
      where ESPECIFICACION.pta_cve_pta = deleted.pta_cve_pta and
            ESPECIFICACION.pdt_cve_prd = deleted.pdt_cve_prd and
            ESPECIFICACION.cea_anio_cer = deleted.cea_anio_cer and
            ESPECIFICACION.cea_num_analisis =
deleted.cea_num_analisis

return

error:
    raiserror @errno @errmsg
    rollback transaction
end
                                                    /* Fin
CER_ANALISIS_DEL*/
go


create trigger CER_ANALISIS_UPD on CER_ANALISIS for UPDATE as
                                                    /* Mon Sep 04
16:32:55 1995 */
begin
declare  @numrows     int,
         @cambio      float,
         @actual      float,
         @compromiso float,
         @errno    int,
         @errmsg   varchar(255)
select @numrows = @@rowcount
if @numrows = 0
  return
                                                    /* FACTURA_NAL
Mon Sep 04 16:32:54 1995 */
if exists ( select * from  deleted, FACTURA_NAL
              where FACTURA_NAL.pta_cve_pta = deleted.pta_cve_pta
and
                    FACTURA_NAL.pdt_cve_prd = deleted.pdt_cve_prd
and
                    FACTURA_NAL.cea_anio_cer =
deleted.cea_anio_cer and
```

```
                    FACTURA_NAL.cea_num_analisis =
deleted.cea_num_analisis )
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: CER_ANALISIS por ser usado
en FACTURA_NAL.'
  goto error
  end
                                                    /* HIS_LOTE_ALM
Mon Sep 04 16:32:54 1995 */
if exists ( select * from  deleted, HIS_LOTE_ALM
  where HIS_LOTE_ALM.pta_cve_pta = deleted.pta_cve_pta and
        HIS_LOTE_ALM.pdt_cve_prd = deleted.pdt_cve_prd and
        HIS_LOTE_ALM.cea_anio_cer = deleted.cea_anio_cer and
        HIS_LOTE_ALM.cea_num_analisis = deleted.cea_num_analisis )
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: CER_ANALISIS por ser usado
en HIS_LOTE_ALM.'
  goto error
  end
                                                    /*
EMBARQUE_CLIENTE        Mon Sep 04 16:32:54 1995 */
if exists ( select * from  deleted, EMBARQUE_CLIENTE
  where EMBARQUE_CLIENTE.pta_cve_pta = deleted.pta_cve_pta and
        EMBARQUE_CLIENTE.pdt_cve_prd = deleted.pdt_cve_prd and
        EMBARQUE_CLIENTE.cea_anio_cer = deleted.cea_anio_cer and
        EMBARQUE_CLIENTE.cea_num_analisis =
deleted.cea_num_analisis )
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: CER_ANALISIS por ser usado
en EMBARQUE_CLIENTE.'
  goto error
  end
                                                    /* INV_LOTE_ALM
Mon Sep 04
```

```
16:32:55 1995 */
if exists ( select * from  deleted, INV_LOTE_ALM
       where INV_LOTE_ALM.pta_cve_pta = deleted.pta_cve_pta and
         INV_LOTE_ALM.pdt_cve_prd = deleted.pdt_cve_prd and
         INV_LOTE_ALM.cea_anio_cer = deleted.cea_anio_cer and
         INV_LOTE_ALM.cea_num_analisis = deleted.cea_num_analisis
)
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: CER_ANALISIS por ser usado
en INV_LOTE_ALM.'
  goto error
  end
                                                        /* EMB_ALM
Mon Sep 04 16:32:55 1995 */
if exists ( select * from  deleted, EMB_ALM
              where EMB_ALM.pta_cve_pta = deleted.pta_cve_pta and
                    EMB_ALM.pdt_cve_prd = deleted.pdt_cve_prd and
                    EMB_ALM.cea_anio_cer = deleted.cea_anio_cer
and
                    EMB_ALM.cea_num_analisis =
deleted.cea_num_analisis )
  begin
  select @errno  = 30005
  select @errmsg = 'No puede cambiar: CER_ANALISIS por ser usado
en EMB_ALM.'
  goto error
  end

return

error:
    raiserror @errno @errmsg
    rollback transaction
end
                                    /* Fin CER_ANALISIS_UPD */
go
```

VISTAS PARA LAS CONSULTAS DEL CLIENTE

```
/* Vista entre Certificado de Análsis y Embarque Cliente*/
CREATE VIEW VST_CER_CTE
AS
SELECT CEA.pdt_cve_prd, PDT.pdt_nom_espanol,
       CEA.pta_cve_pta, PTA.pta_raz_soc_pta,
       CEA.cea_anio_cer, CEA.cea_num_analisis,
       CEA.cea_fec_elaboracion_certf,
       CEA.cea_nom_elaboro,
       CEA.cea_dsc_para_el_pie_certfic,
       ECT.cia_cve_compania, CIA.cia_raz_soc,
       ECT.ide_fec_embarque,
       ECT.ect_placa_camion_num_carro,
       ECT.emp_cve_emp, EMP.emp_nom_emp,
       ECT.cte_cve_cte, CTE.cte_raz_soc,
       ECT.ect_peso_neto_embarcado

FROM   CER_ANALISIS CEA, EMBARQUE_CLIENTE ECT,
       PRODUCTO PDT, CLIENTE CTE, PLANTA PTA, COMPANIA CIA,
       EMPAQUE EMP
WHERE  CEA.pdt_cve_prd = ECT.pdt_cve_prd AND
       CEA.pta_cve_pta = ECT.pta_cve_pta AND
       CEA.cea_anio_cer = ECT.cea_anio_cer AND
       CEA.cea_num_analisis = ECT.cea_num_analisis AND
       CEA.pdt_cve_prd = PDT.pdt_cve_prd AND
       CEA.pta_cve_pta = PTA.pta_cve_pta AND
       ECT.cia_cve_compania = CIA.cia_cve_compania AND
       ECT.emp_cve_emp = EMP.emp_cve_emp AND
       ECT.cte_cve_cte = CTE.cte_cve_cte

/*Vista entre Certificado de Análisis y Embarque Almacén*/
CREATE VIEW AS VST_CER_ANALSIS_EMB_ALM
AS
SELECT CEA.pta_cve_pta, CEA.pdt_cve_prd, CEA.cea_anio_cer,
       CEA.cea_num_analisis,
       CEA.cea_fec_elaboracion_certf, CEA.cea_nom_elaboro,
       CEA.cea_dsc_para_el_pie_certfic,
       CEA.cia_cve_compania, EBD.ide_fec_embarque,
       EDB.emp_cve_emp,
       EBD.ebd_placa_camion_num_carro, EBD.alm_cve,
       EBD.ebd_fec_fabricacion,
       EBD.ebd_fec_estimada_arrivo,
```

VISTAS PARA LAS CONSULTAS DEL CLIENTE

```
/* Vista entre Certificado de Análsis y Embarque Cliente*/
CREATE VIEW VST_CER_CTE
AS
SELECT CEA.pdt_cve_prd, PDT.pdt_nom_espanol,
       CEA.pta_cve_pta, PTA.pta_raz_soc_pta,
       CEA.cea_anio_cer, CEA.cea_num_analisis,
       CEA.cea_fec_elaboracion_certf,
       CEA.cea_nom_elaboro,
       CEA.cea_dsc_para_el_pie_certfic,
       ECT.cia_cve_compania, CIA.cia_raz_soc,
       ECT.ide_fec_embarque,
       ECT.ect_placa_camion_num_carro,
       ECT.emp_cve_emp, EMP.emp_nom_emp,
       ECT.cte_cve_cte, CTE.cte_raz_soc,
       ECT.ect_peso_neto_embarcado

FROM   CER_ANALISIS CEA, EMBARQUE_CLIENTE ECT,
       PRODUCTO PDT, CLIENTE CTE, PLANTA PTA, COMPANIA CIA,
       EMPAQUE EMP
WHERE  CEA.pdt_cve_prd = ECT.pdt_cve_prd AND
       CEA.pta_cve_pta = ECT.pta_cve_pta AND
       CEA.cea_anio_cer = ECT.cea_anio_cer AND
       CEA.cea_num_analisis = ECT.cea_num_analisis AND
       CEA.pdt_cve_prd = PDT.pdt_cve_prd AND
       CEA.pta_cve_pta = PTA.pta_cve_pta AND
       ECT.cia_cve_compania = CIA.cia_cve_compania AND
       ECT.emp_cve_emp = EMP.emp_cve_emp AND
       ECT.cte_cve_cte = CTE.cte_cve_cte

/*Vista entre Certificado de Análisis y Embarque Almacén*/
CREATE VIEW AS VST_CER_ANALSIS_EMB_ALM
AS
SELECT CEA.pta_cve_pta, CEA.pdt_cve_prd, CEA.cea_anio_cer,
       CEA.cea_num_analisis,
       CEA.cea_fec_elaboracion_certf, CEA.cea_nom_elaboro,
       CEA.cea_dsc_para_el_pie_certfic,
       CEA.cia_cve_compania, EBD.ide_fec_embarque,
       EDB.emp_cve_emp,
       EBD.ebd_placa_camion_num_carro, EBD.alm_cve,
       EBD.ebd_fec_fabricacion,
       EBD.ebd_fec_estimada_arrivo,
```

```
      EBD.ebd_peso_neto_embarcado,
      EBD.ebd_peso_bruto_embarcado,
      EBD.ebd_cnt_emp_embarcado, EBD.ebd_cve_lote,
      EBD.ebd_tipo_equipo
FROM CER_ANALISIS CEA, EMB_ALM EBD
WHERE CEA.cea_anio_cer = EBD.cea_anio_cer AND
      CEA.cea_num_analisis = EBD.cea_num_analisis AND
      CEA.pdt_cve_prd = EBD.pdt_cve_prd AND
      CEA.pta_cve_pta = EBD.pta_cve_pta
```

```
       EBD.ebd_peso_neto_embarcado,
       EBD.ebd_peso_bruto_embarcado,
       EBD.ebd_cnt_emp_embarcado, EBD.ebd_cve_lote,
       EBD.ebd_tipo_equipo
FROM CER_ANALISIS CEA, EMB_ALM EBD
WHERE CEA.cea_anio_cer = EBD.cea_anio_cer AND
       CEA.cea_num_analisis = EBD.cea_num_analisis AND
       CEA.pdt_cve_prd = EBD.pdt_cve_prd AND
       CEA.pta_cve_pta = EBD.pta_cve_pta
```

/*Vista para Certifcado de Análisis con Factura*/

```
CREATE VIEW VST_FACTURA_CER
AS
SELECT CEA.pta_cve_pta, CEAS.pdt_cve_prd, CEA.cea_anio_cer,
CEA.cea_num_analisis,
                CEA.cea_fec_elaboracion_certf,
CEA.cea_nom_elaboro, CEA.cea_dsc_para_el_pie_certfic,
                FAN.cia_cve_compania, FAN.lgf_serie_fac,
FAN.fan_num_fac, FAN.cte_cve_cte, emp_cve_emp,
                FAN.fan_dsc_producto, FAN.fan_precio_unitario
FROM CER_ANALISIS CEA, FACTURA_NAL FAN
WHERE CEA.pta_cve_pta = FAN.pta_cve_pta AND
               CEA.pdt_cve_prd = FAN.pdt_cve_prd AND
               CEA.cea_anio_cer = FAN.cea_anio_cer AND
              CEA.cea_num_analisis = FAN.cea_num_analisis
```

/*Vista para certificado de Análisis con Inventario lote bodega*/

```
CREATE VIEW VST_INV_LOTE-ALM-CER
AS
SELECT CEA.pta_cve_pta, CEA.pdt_cve_prd, CEA.cea_anio_cer,
CEA.cea_num_analisis,
                CEA.cea_fec_elaboracion_certf,
CEA.cea_nom_elaboro, CEA.cea_dsc_para_el_pie_certfic,
                IVB.alm_cve,  IVB.emp_cve_emp,
IVB.cia_cve_compania, IBV.ivb_fec_embarque_pta,
FROM CER_ANALISIS CEA, INV_LOTE_ALM IVB
WHERE CEA.pta_cve_pta = IVB.pta_cve_pta AND
                CEA.pdt_cve_prd = IVB.pdt_cve_prd AND
                CEA.cea_anio_cer = IVB.cea_anio_cer AND
                CEA.cea_num_analisis = IVB.cea_num_analisis
```

/*Vista para Certifcado de Análisis, Producto y planta*/

```
CREATE VIEW VST_CER_ANALISIS
AS
SELECT CEA.pdt_cve_prd,PDT.pdt_nom_espanol,
       CEA.pta_cve_pta, PTA.pta_raz_soc_pta,
       CEA.cea_anio_cer, CEA.cea_num_analisis,
       CEA.cea_fec_elaboracion_certf,
       CEA.cea_dsc_para_el_pie_certfic, CEA.cea_nom_elaboro
FROM CER_ANALISIS CEA, PLANTA PTA, PRODUCTO PDT
WHERE CEA.pta_cve_pta = PTA.pta_cve_pta AND
      CEA.pdt_cve_prd = PDT.pdt_cve_prd
```

APENDICE B
CODIGO DEL CLIENTE Y ESTANDARES

ESTANDARES DE REALIZACION

Nombres de objetos

Objeto	Prefijo	Ejemplo
Form Window	frm	frmNombre
Table Window	tbl	tblNombre
QuestWindow	tbl	tblNombre
DialogBox	dlg	dlgNombre
MDI window	mdi	mdiNombre
Data field	df	dfNombre
Multiline field	ml	mlNombre
Push botton	pb	pbNombre
Radioboton	rb	rbNombre
Check box	cb	cbNombre
Option button	ob	obNombre
List box	lb	lbNombre
Combo box	cmb	cmbNombre
Picture	pic	picNombre
Horizontal scroll bar	hsb	hsbNombre
Vertical scroll bar	vsb	vsbNombre
Custom control	cc	ccNombre
Column	col	colNombre

Cuando declaremos un objeto, el nombre de este debe de llevar el prefijo correspondiente, seguido del nombre o naturaleza del mismo.

Nombre de objetos que hagan refencia a la base de datos

Nombre para atributos

Para atributos, se escribe el prefijo del tipo de objeto, seguido de un underscore y el nombre tal cual del atributo:

df_cte_cve_cte

Nombre para tablas

Para tablas, se escribe el prefijo, seguido del nombre de la tabla:
tbl**CLIENTE**

Nombre para objeto que hagan procesos

Nombre Proceso	Significado
Alta	Insert a una o más tablas
Borra	Delete a una o más tablas
Consulta	Select a una o más tablas o vistas
Cambio	Update a una tabla
Copia	Insert a una tabla
Aplica Alta	Aplica el insert
Aplica Borra	Aplica el Delete
Aplica Cambio	Aplica el Update
Aplica Copia	Aplica el Insert
Cancela	Cancela la acción en cuestión
Limpia	Inicializa el estado de una forma

Si el objeto hace referencia a un proceso, como es el caso de un push botón el nombre será indicando el prefijo, seguido del nombre del proceso (Alta, Baja, Cambio, Consulta, Copia).

Ejemplo

pbAlta	*pb pushbotton*	*Proceso de Alta*
pbConsulta	*pb pushbotton*	*Consulta Proceso*
pbAplicaAlta	*pb pushbotton*	*AplicaAlta Proceso*

Nombre para objetos de control

Los objetos de control son aquellos que son utilizados para coordinar el flujo de la forma o mdi en cuestión, por lo que pueden ser cualquier tipo de objetos. El nombre de estos se define con el prefijo correspondiente, seguido de un nombre nemónico apropiado para la función que realice.

Ejemplo

cbPedidoPta *cb check box PedidoPta es un check box para determinar el tipo de pedido(Planta)*
cbPedidoAlm *cb check box PedidoAlm es un check box para determinar el tipo de pedido (Almacen)*

Nombre de variables

Tipo de dato	Prefijo	Ejemplo
Boolean	b	bNombre
Date/Time	d	dNombre
File handle	fh	fhNombre
Long String	lstr	lstrNombre
Number	n	nNombre
Sql Handle	hsql	hsqlNombre
String	str	strNombre

Cuando declaremos una variable el nombre de esta debe llevar el prefijo correspondiente a su tipo, seguido de una palabra que lo describa.

Nombre de variables que hagan refencia a la base de datos

Nombre para atributos

Para atributos, se escribe el prefijo, seguido de un underscore y el nombre tal cual del atributo:

str_cte_cve_cte

Nombre para tablas

Para tablas, se escribe el prefijo, seguido del nombre de la tabla:

strCLIENTE

Nombre para variables de control

Las variables de control son aquellas utilizadas para coordinar el flujo de la forma o mdi en cuestión. El nombre de estos se define con el prefijo correspondiente, seguido de un nombre nemónico apropiado para la función que realice.

Ejemplo

nInd *n numérico Ind Índice*
strPaso *str string String Auxiliar para asignación*

Nombre de variables que hacen referencia a objetos

Estas variables se usan para asociarlas a un objeto, las cuales son necesarias para identificar ciertas características del objetos, como el estado de activo o inactivo, abierto o cerrado, etc. El nombre se define con el prefijo del tipo de objeto, seguido del nombre del objeto.

Ejemplo:

bFrmCLIENTE *variable booleana para la Forma* ***frmCLIENTE.***

Nombres de mensajes, objetos y funciones

Nombre de Mensajes

Los mensajes son eventos que podemos definir en el programa para ser enviados a un objeto. Para definir su nombre:

SAM_User Nombre

Deben de comenzar con el prefijo 'SAM_User', seguido del nombre del mensaje.

Nombre de clases

Para declarar un objeto en el editor de clases, el nombre debe ser:

cPrefijoNombre

Se define la 'c' de clase, seguido del tipo prefijo del tipo de objeto y por último el nombre de la clase
Ejemplo:

cCmbConsulta *clase de un combo box que realiza un Query de consulta a un campo de una tabla*

Nombre de funciones

Para declarar funciones el nombre debe ser; el prefijo de 'SalUser' seguido del nombre de la función.

Ejemplo:

SalUserInsert *Función que realiza un insert a una tabla*

Variables incluidas en nuestro ambiente de desarrollo

Nombre de Variable	*Descripción*
hSqlSyb1	Variable de tipo sql handle para identificar al cursor del cliente que abre una canal de comunicaciones entre el y el servidor
hSqlError	Variable de tipo sql handle para identificar al cursor del cliente, el cual atrapa los errores del servidor al cliente.
strSelect	String de una clausula Select
strWhere	String de una cláusula Where para un insert, select, delete o update
strInto	String de una cláusula Into para insert
strInsert	String de una cláusula Insert
strDelete	String de una cláusula delete
strUpdate	String de una cláusula Update
strValues	String de una cláusula Values para un insert o update
strCampo[1:40]	Matriz de tipo string para almacenar los nombres de las columnas de una tabla para construir el 'where' de cualquier cláusula SQL.
strValor[1:40]	Matriz de tipo string para almacenar los valores de los objetos de la interface, para construir el 'where' de cualquier cláusula SQL
strCampoUp[1:40]	Matriz de tipo string para almacenar los nombres de las columnas de una tabla, las cuales serán modificados en un Update
strValorUp[1:40]	Matriz de tipo string para almacenar los valores de las columnas que serán actualizadas en un Update.
strCampoWhere[1:40]	Matriz de tipo string que almacena las columnas de una tabla de Sybase, los cuales servirán para construir un 'where'.

Mensajes incluidos en nuestro ambiente de desarrollo

Mensaje	*Descripción*
SAM_UserEnable	Habilita el objeto
SAM_UserDisable	Desabilita el objeto
SAM_UserLimpia	Limpia los campos de una forma
SAM_UserDelete	Asigna los valores para hacer una cláusula Delete
SAM_UserInsert	Asigna los valores para hacer una cláusula Insert
SAM_UserSelect	Asigna los valores para hacer una cláusula Select
SAM_UserUpdate	Asigna los valores para hacer una cláusula Update
SAM_UserSelectHijo	Asigna los valores para hacer una cláusula Select de un tabla detalle
SAM_UserShow	Muestra el objeto
SAM_UserHide	Oculta el objeto
SAM_UserEdoTrabajo	Deja a la forma en un estado de trabajo

Objetos o Clases

Nombre de la Clase	Descripción	Tipo de Objeto
cBackgroundVQ	Etiqueta (Titulo) dentro de una frm	Background Text
cBotonCerrar	Para cerrar frm o tbl	Pushbutton
cBotonTam	Define características de Tamaño para pb de los programas.	Pushbutton
cCmbDina	Hace consultas dinámicas sobre una Columna (limitado)	Combo Box
cColDateHija	Definir columna fecha en table hija	Column
cColNumHija	Definir columna numérica en table hija	Column
cColStrHija	Definir columna string en table hija	Column
cDfDateVQ	Definir campos. fecha	Data Field
cDfNumVQ	Definir campos numéricos	Data Field
cDfStrVQ	Definir campos string	Data Field
cFormDatosGenerales	Contiene los datos (no PK) del objeto principal de la aplicación.	Form Window
cMDIQuimico	Prototipo de pantalla para realizar una aplicación.	MDI Window
cPbAlta	Botón que define un estado de Alta a la aplicación	Pushbutton
cpbAplicaAlta	Botón que realiza la función de armar un Insert (Alta)	Pushbutton
cpbAplicaBorra	Botón que ejecuta la función de Delete (Borrar)	Pushbutton
cpbAplicaCambio	Botón que ejecuta la función de Update (Actualización)	Pushbutton
cpbAplicaCopia	Botón que realiza la función de Insert (Alta de Copia)	Pushbutton
cPbBorra	Botón que define un estado de borrado para la aplicación.	Pushbutton
cPbCambio	Botón que define un estado de cambio para la aplicación	Pushbutton
cPbConsulta	Botón que ejecuta una consulta de aplicación en cuestión	Pushbutton
cPbCopia	Botón que define un estado de copia a la aplicación	Pushbutton
cPbCopiaHija	Realiza una copia de un rengión válido en una tabla hija	Pushbutton
cPbDetalle	Botón base para la creación de botones asociados a las tabla hijas, para abrirlas o cerrarlas	Pushbutton

Funciones

Funciones	Descripción
SalUserDialog	Función la cual despliega un dialog box con un mensaje. Parámetros : strNota Mensaje a desplegar
SalUserDelete	Función para armar y ejecutar un transacción que hará un delete de una tabla padre y hasta 4 tablas hijas. Parámetros: strTablaHija1 Nombre de la tabla hija 1 strTablaHija2 Nombre de la tabla hija 2 strTablaHija3 Nombre de la tabla hija 3 strTablaHija4 Nombre de la tabla hija 4 strTablaPadre Nombre de la tabla Padre nTabla Número de tablas a borrar nCampoPK Número de campos de la PK de la tabla padre
SalUserDragDrop	Función para hacer un drag and drop de las columnas de una tbl de consulta a los campos correspondientes de la frmDatosGenerales
SalUserUpdate	Función para armar una instrucción Update Parámetros strTabla Nombre de la tabla a actualizar nNumeroCampo Número de campos a actualizar nCampoPK Número de campos que forman la PK de la tabla
SalUserUpdateTran	Función para ejecutar una transacción la cual fue armada por las funciones: SalUserUpdate, SalUserInsert y/o SalUserDelete Esta función regresa un valor booleano, si es verdadero, la transacción fue llevada con éxito, de lo contrario será falsa
SalUserInsert	Función para armar una instrucción Insert Parámetros String: strTabla Nombre de la tabla a insertar Number: nNumeroCampo Numero de campos a insertar Boolean: bTipoTabla Si es verdadero la inserción es de tabla padre, sino es hijo
SalUserInsertTran	Función para ejecutar una transacción de clausuras Insert, las cuales fueron armadas por la función: fSalUserInsert. Esta función regresa un valor booleano, si es verdadero, la transacción fue llevada con éxito, de lo contrario será falsa

CODIGO FUENTE DE CERTIFICADOS DE ANALSIS

```
cMDIQuimico: mdiCERTIFICADO_ANALISIS
  Menu
    Popup Menu: &Archivo
      Menu Item: &Guardar
      Menu Item: &Imprimir Certificado
      Menu Item: &Salir
    Named Menu: menuEdit
    Popup Menu: &Funciones
      Menu Item: &Alta
      Menu Item: &Borra
      Menu Item: Ca&mbio
      Menu Item: &Consulta
      Menu Item: Co&pia
      Menu Item: &Limpia
      Menu Item: Copia a &Forma
      Menu Item: Aplica&r
    Popup Menu: &Detalles
      Menu Item: &Especificaciones
    Named Menu: menuMDIWindows
    Popup Menu: &Ayuda
      Menu Item: &Indice
      Menu Item: &Procedimiento
      Menu Item: A&cerca de...
  Contents
    cdfstrVQPK: dfProducto
    cCmbDinaPK: cmbProducto
    cGroupBoxClassVQPK: Producto
    cdfstrVQPK: dfPlanta
    cCmbDinaPK: cmbPlanta
    cGroupBoxClassVQPK: Planta
    cdfnumVQPK: dfClave_del_lote
    cdfstrVQPK: df_clave_lote_num  ! Clave del lote numero certificado
    bkgIdentificador: Clave del Lote
    cPbDetalle: pbEspecificacion   ! Boton para ver la especificaciones en mditblEspecificaiones
    cMDIQuimico: pbCopia
    cMDIQuimico: pbLimpia
    cMDIQuimico: pbAlta
      Message Actions
        On SAM_Click
          ! Sección de Estado de las banderas
          Set bEdoConsulta=FALSE
          Set bEdoAlta=TRUE
          Set bEdoTrabajo=FALSE
          ! Sección de estado de los objetos de la forma
          Call SalSendClassMessage(SAM_Click, 0,0 )
          Call SalSendMsgToChildren( frmCERTIFICADO_ANALISIS, SAM_UserEnable, 0,0 )
          ! Sección de fecha
          Call SalDateToStr( frmCERTIFICADO_ANALISIS.dtFec_de_emision,strCampo0)
          If strCampo0 = ''
          Set frmCERTIFICADO_ANALISIS.dtFec_de_emision=SalUserConstructDate( )
```

```
        Set dfecha=SalUserConstructDate( )
        If sEspecificaciones = 'si'
          Call SalSendMsg(mditblEspecificacion, SAM_UserEdoAltaDet, 0,0 )
        Else
          Call SalSendMsg(pbEspacificacion, SAM_Click,0, 0 )
          Call SalDisableWindow( pbEspacificacion )
          Call SalSetFocus( dfProducto )
cMDIQuimico: pbBorra
cMDIQuimico: pbConsulta

cMDIQuimico: pbAplicaAlta
  Message Actions
    On SAM_Click
      Call SalDateToStr(frmCERTIFICADO_ANALISIS.dfFec_de_emision,strCampo0)
      Set nOpcErr=0  ! Inicializa rutina de errores
      ! Sección de validación de lo campos de la forma
      If dfProducto = ''
        Call SalUserDialog( 'Falta definir la Clave del Producto' )
      Else If dfPlanta = ''
        Call SalUserDialog( 'Falta definir la Clave de la Planta' )
      Else If SalNumberToStrX(dfClave_del_lote,0) = ''
        Call SalUserDialog( 'Falta definir la  Clave del Lote' )
      Else If df_clave_lote_num = ''
        Call SalUserDialog( 'Falta definir la  Clave del Lote' )
      Else If strCampo0 = ''
        Call SalUserDialog( 'Falta definir la  Fecha de Emisión' )
      Else If frmCERTIFICADO_ANALISIS.ml_descripcion = ''
        Call SalUserDialog( 'Falta definir la  descripción' )
      Else If frmCERTIFICADO_ANALISIS.dfElaboro = ''
        Call SalUserDialog( 'Falta definir el nombre de quien elaboro' )
      Else If sEspecificaciones = 'no'
        Call SalUserDialog( 'Falta definir las Especificaciones' )
      Else If SalTblSetRow( mditblEspecificacion,TBL_SetLastRow ) = 0 and
        mditblEspecificacion.col_esp_unidad_dimensional = '' and
        SalNumberToStrX(mditblEspecificacion.col_ep_valor_obtenido,0) = '' and
        mditblEspecificacion.col_des_especificacion= ''
       ! Esta condición indica que la matriz de especificaciones esta vacia o no
        Call SalUserDialog( 'Falta definir las Especificaciones ' )
       Else
        ! Aplicación del Alta
        ! Sección para asignar los valores a las matrices para hecer el insert
         Call SalSendMsg( mditblEspecificacion.pbAplicaInsertaAltaHija, SAM_Click,0,0)
         Set nIndIns = 1
         Call SalSendMsgToChildren( mdiCERTIFICADO_ANALISIS, SAM_UserInsert, 0,0 )
         Call SalSendMsgToChildren( frmCERTIFICADO_ANALISIS, SAM_UserInsert, 0,0 )
        ! Sección del armado del insert Padre
         Call SalUserInsert('CER_ANALISIS', 7, TRUE )
        ! Sección de la Ejecución de la transacción
         If SalUserInsertTran( )
            ! Sección de estado de trabajo, despues de un alta
            Call SalSendClassMessage(SAM_UserAplicaAlta, 0,0 )
            If sEspecificaciones = 'si'
              Call SalSendMsg(mditblEspecificacion, SAM_UserEspecifAlta, 0,0)
            Set bEdoAlta=FALSE
```

```
                Set bEdoTrabajo=TRUE
    cMDIQuimico: pbAplicaBorra
    cMDIQuimico: pbAplicaCambio
    cMDIQuimico: pbAplicaCopia
    cMDIQuimico: pbCambio
     cMDIQuimico: pbSalir
Contents
   ctblHija: mditblEspecificacion
   ctblConsultas: mditblCertificado
   Table Window: mditblFactura
   Table Window: mditblEmbAlm
   Table Window: mditblExistencia
   cFrmDatosGenarales: frmCERTIFICADO_ANALISIS

Dialog Box: dlgConsulta
```

GLOSARIO

API	(Application Program Interface).Software que proporciona a las aplicaciones servicios de re.
Arquitectura Cliente/Servidor	Es un modelo computacional para sistemas, el cual es de tipo distribuido, en donde intervienen tres elementos, el Cliente, la Red y el Servidor.
Atributo	Es una característica de una Entidad.
Base de Datos	Conjunto de datos, los cuales guardan una relación entre si.
Campo	Es la transformación de un atributo de análisis a diseño.
Cliente	Es el software almacenado en una PC, el cual tienen la función de llevar la interface de control y despliegue de datos en un sistema, así mismo realiza peticiones de servicios a un servidor, tales como datos, procesos, comunicaciones, etc.
Diagrama de Descripción de Entidades	Modelo gráfico que describe los atributos de las entidades.
Diagrama de Flujo de Datos	Modelo gráfico que describe el flujo de los datos entre los procesos de un contexto.
Diagrama de Flujo de Eventos	Modelo gráfico que describe el ciclo de estados de una entidad u objeto.
Diagrama de Flujo de Objetos	Modelo gráfico que describe la afectación de objetos por los eventos de un contexto.
Diagrama Entidad-Relación	Modelo gráfico que describe los atributos de las entidades y su relación entre cada una de estas.
Disparadores (Triggers)	Es un procedimiento almacenado que se ejecuta cuando se da una acción de Update, Insert y Delete.

Entidad	Es un elemento que describe información de interés al contexto del sistema en cuestión.
Filosofía Cliente/Servidor	Es la conceptualización de la forma en la cual funciona el proceso cliente-servidor.
GUI	(Grafic User Interface). Son las Interfaces Gráficas del Usuario, que son usadas por el cliente, para la operación del sistema.
Indice	Medio lógico para hacer acceso a un tabla de una base de datos.
Indice Clustered	Indice el cual forma parte de la estructura de datos de una tabla, y el cual se utiliza para ordenar en forma ascendente o descendente los registros de una tabla.
Indice NonClustered	Estructura de datos, la cual se utiliza para la localización de un dato en una tabla.
Integridad de datos	Son una serie de reglas y criterios a seguir para que los datos de un sistema, sean coherentes respecto a las reglas y requerimientos del negocio.
Integridad referencial	Son una serie de reglas y criterios a cumplir para que los datos de un sistema, se mantengan coherentes respecto a las relaciones entre las tablas de la base de datos.
Llave Foránea (FK) Foreing Key	Es una o más columnas de una tabla, la cual o cuales son llaves primarias de otra tabla, esta FK, puede o no pertenecer a la llave primaria de la tabla donde es foránea.
Llave Primaria (PK) Primary Key	Es una o más columnas de una tabla, la cual identifica a un renglón de la tabla, y asegura su unicidad y que no acepte valores nulos.
Middleware	Conjunto de hardware y software que proporciona servicios de red a un cliente.
Modelo Dinámico	Concepción lógica de los estados que puede tener un objeto o entidad de un sistema.

Modelo entidad/objeto	Concepción lógica de elementos y relaciones de un contexto o sistema.
Modelo Funcional	Concepción lógica del flujo de datos que intervienen en los procesos de un sistema dado.
Paradigma Orientado a Objetos	Forma de conceptualizar a los elementos de un sistema, en elementos independientes, que funcionan por si mismos.
Procedimientos Almacenados (store procedures)	Procedimientos en código de un manejador de base de datos, que ejecutan ciertas tareas cuando son invocados.
Programación Orientada a Eventos	Método de programación de software, el cual basa su funcionamiento en base a la ejecución de procesos, cuando sucede un evento o condición.
Programación Orientada a Objetos	Método de programación de software, el cual basa la construcción de programas en base a objetos independientes que tienen su propio funcionamiento y datos.
Prototipo	Modelo de software base, para el desarrollo de las aplicaciones de un sistema.
Red	Conjunto de nodos interconectados, generalmente computadoras o dispositivos, los cuales tienen el objetivo de compartir recursos.
Relación	Es la asociación que existe entre dos entidades.
Restricciones (Constraint)	Es una regla o condición que debe cumplir una o más columnas de una tabla.
Servidor	Software almacenado en una equipo de alta capacidad de proceso, el cual presta servicios de proceso, de almacenamiento de datos, de impresión, de comunicación, etc.
SIPSA	Servicios Industriales Peñoles, S.A.

Sistema	Conjunto de elementos que interactuan entre si, para llegar a la consecución de un objetivo común.
Sistema abierto	Son aquellos sistemas, los cuales tienen una interacción con su medio ambiente, generando salidas y recolectando entradas.
Sistema centralizados	Sistema el cual centra en un solo nodo sus procesos y datos.
Sistema Cliente/Servidor	Sistema de información automatizado, el cual distribuye sus tareas entre el cliente y el servidor, los cuales se comunican a través de una red.
Sistema de tiempo real	Sistema capaz de influir en un medio ambiente casi inmediatamente, cuando entra a el un insumo.
Sistema distribuido	Sistema el cual distribuye sus datos y procesos en varios nodos.
Sistema en línea	Sistema que realiza procesos en forma interactiva con el usuario.
Sistema cerrado	Sistema el cual no tiene interacción con su medio ambiente.
Tabla	Es la transformación de una entidad de análisis a un elemento de almacenaje físico de información, en una base de datos.
Tipos de datos	Es un dominio de información definido en una base de datos.

BIBLOGRAFIA

BIBLIOGRAFIA

- **Analsis y Diseño Orientado a Objetos**
  Martin, James
  Prentice-Hall
  México- Enclewood Clifss
  pp

- **Análisis Estructurado Moderno**
  Yourdon, Edward
  Prentice Hall
  México- Enclewood Clifss
  735pp

- **Manual Fast Track to SYBASE System 10 Tomo I**
  Sybase de México
  México D.F:

- **Manual Fast Track to SYBASE System 10 Tomo II**
  Sybase de México
  México D.F:

- **Manual Tunning and Performence Sybase**
  Sybase de México
  México D.F.

- **Testing Client/Server**
  Goglia, Patricia
  QUED Publishing Group
  Boston
  315pp

- **Cliente/Server Strategies**
  Davis Vaskevitch
  Info Word

- **Manual de Análsis y Diseño**
  FACTUM, S.A. de Mancera Hnos
  México D.F.

HEMEROGRAFIA

- **Personal Computing Mexico**
  Cliente-Servidor: Moda o Tendencia?
  Noviembre 1993
  pp.26

- **Window sources**
  Karen Watterson.
  Data on Demand
  Febrero 1994.
  pp.212

- **DBMS DataBase & Client/Server Solutions**
  Development Methodologies
  Steps to Client/Server Success
  Mayo 1995
  pp 48