UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"SISTEMA ADMINISTRADOR DE APLICACIONES"

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

ALMA CECILIA RUIZ JIMENEZ
ANTONIO AMEZCUA CHAVEZ

DIRECTOR DE TESIS: ING. ADOLFO MILLAN NAJERA.

CIUDAD UNIVERSITARIA. 1998.

A Dios por el camino de oportunidades que me ofreció.

A mis dos grandes amores Omar y Yosafat, por llenar mi vida de motivos.

A mis padres Martha y Samuel por su amor, su apoyo y buen ejemplo.

A Dennise Alejandra por ser una persona tan especial.

A mis hermanos: Rosario, Alfonso, Samuel, Carlos, Mary, Pablo, Maru, Daniel, Carmen, Martha, Rafael y Lucy; y sus familias por su apoyo siempre presente.

A todos mis amigos por su valiosa opinión.

Alma Cecilia.

A Dios por permitirme llegar hasta estos momentos.

A mi mamá quien siempre me ha brindado cariño y apoyo incondicional.

Al recuerdo de mis abuelos y mi padre.

A todos mis familiares y amigos por su apoyo y confianza.

Antonio.

Agradecemos al Ing. Adolfo Millán Nájera por su valiosa cooperación en la realización de este trabajo.

OBJETIVO GENERAL

- El objetivo de este trabajo es realizar un *Sistema Administrador de Aplicaciones* que proporcione a los usuarios de una herramienta de acceso única a la información, utilizando *Bases de Datos Relacionales* en plataforma *Sybase, Arquitectura Cliente/Servidor, Metodología de Desarrollo en Espiral* y *Lenguajes de Cuarta Generación.*

OBJETIVOS ESPECÍFICOS

- Proporcionar a los usuarios del *acceso a la información* que requieren para realizar sus análisis de manera eficiente y confiable.
- Unificar el acceso a todas las aplicaciones internas para simplificar las tareas.
- Diseñar un esquema de privilegios para seguridad de la información.
- Controlar los accesos de los usuarios del sistema de manera fácil y rápida

ÍNDICE

Prólogo

Prólogo

El objetivo de este sistema es Administrar las Aplicaciones para darle a los usuarios los privilegios adecuados para el uso de la información que se maneja en una institución.

Proporcionar a los usuarios el acceso a la información que requieren para realizar sus análisis financieros de manera eficiente y confiable.

El sistema a realizar debe ser capaz de concentrar de manera organizada la información con la que cuenta la institución, y tenerla disponible para los usuarios.

Asignar privilegios a cada usuarios según sean sus necesidades de información, proporcionando las aplicaciones con las que cuenta actualmente la institución y simplificando el acceso.

El sistema debe tener control de los usuarios del sistema de manera fácil y rápida, debe ser un buen auxiliar en vigilar la seguridad de la información. El acceso deberá estar sujeto a restricciones que el dueño disponga, es decir, sólo se podrá accesar a esta información con autorización del dueño de dicha información, para lo cual se utilizará un esquema de privilegios, tal que solo puedan accesar a cierta información permitida.

El sistema les pedirá su identificación (clave de Red y Password), con lo cual sabrá cuáles son los atributos asignados para cada Base de Datos. La metadatos contienen la información de las Aplicaciones (información de los Usuarios, de los

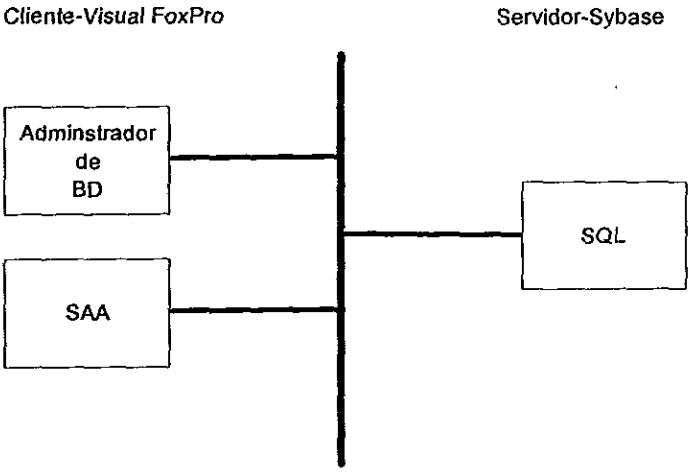
Rutas de acceso a las Aplicaciones, de los Atributos de cada usuario y de los Temas de origen) y se encuentra en una Base de Datos Relacional. En el capítulo II se obtendrá su diseño e implementación

Los usuarios del nuevo sistema quieren ver conjuntadas todas las aplicaciones para ver con mayor claridad la información; no es de su interés saber en donde se encuentra la información físicamente, sino tenerla a la mano en el momento de cualquier consulta.

Una vez creada nuestra Base de Datos existe la necesidad de darle mantenimiento, para que los datos que contengan sean reales, sin embargo esta es una tarea un poco laboriosa por tener que realizar conexiones al servidor Sybase e ir dando cierta prioridad al momento de ir dando de alta nuestros datos (como Altas de Usuarios, Altas de Aplicaciones, Asignarles atributos a un usuario para una aplicación especifica, o alguna otra) por lo que realizarla manualmente resultaría poco práctico. Entonces se plantea realizar un sistema que facilite dicha tarea. Se implementará en el capítulo III

Al terminar nuestro entorno ideal, se proseguirá a poner toda nuestra atención en la realización del Sistema Administrador de Aplicaciones, para realizar la explotación de nuestra metadatos; con las características que ya se han mencionado anteriormente. Se implementará en el capítulo IV

Como los datos se encuentran en Bases de Datos en una plataforma Sybase se realizarán el "Sistema Administrador de la Base de Datos" y el "Sistema Administrador de Aplicaciones" en una arquitectura Cliente-Servidor, ya que por sus características resultará conveniente aprovechar el procesamiento que nos ofrece el servidor, ya que de esta manera el flujo de los datos será el mínimo y por lo tanto representará un gran ahorro de recursos, entre otros de tiempo.



Las ventajas que se han encontrado al elegir este diseño es el hecho de que en la institución se tiene las herramientas necesarias tanto en Cliente como el Servidor, se considera que las herramientas son potentes y de gran velocidad, buena interfaz gráfica con el usuario y con sofisticadas funciones para realizar nuestras consultas y realizar informes, en particular se requiere de un servidor de gran capacidad de almacenamiento y de gran seguridad. Por otro lado el Cliente maneja inicio de sesiones, sintaxis nativa del servidor que es ODBC, realiza vistas remotas, podrá acceder a los grandes almacenes de datos que suelen estar disponibles en las Base de Datos de Servidor. También puede aprovechar las capacidades de seguridad y de procesamiento transaccional del Servidor Remoto.

Podemos tener la seguridad que las herramientas nos permitirán utilizar la funcionalidad del Servidor, con procedimientos almacenados y funciones intrínsecas basadas en el Servidor, tener absoluto control sobre las instrucciones Update, Delete e Insert de paso a través de SQL. Disponer de mayor control sobre las transacciones remotas; solicitar múltiples conexiones para un origen de datos. También es eficiente al trabajar con múltiples orígenes de datos ODBC solicitando una conexión con cada origen de datos al que desea acceder.

Capítulo I

Antecedentes

I.1 Definición del Problema

En una institución se realizan una serie de Aplicaciones para los usuarios; que realizan extracciones, alta y cambios de la información, además de obtener algunos reportes ya definidos y diseñados previamente para cada oficina según sus necesidades.

Estas Aplicaciones tienen un acceso separado mediante iconos y al entrar piden la identificación del usuario, ya que el acceso es restringido sólo a unos cuantos usuarios, generalmente sólo son los que realizaron la petición de la Aplicación.

Los usuarios que desconocen la información que existe en la institución, frecuentemente realizan capturas de dicha información, obtenida de otras fuentes, asumiendo el riesgo que esto conlleva. Pero sin embargo si todos tuvieran conocimiento de la información con la que se cuenta en la institución, seguramente eso no pasaría.

En realidad cada usuario responsable de la información no la esconde por mala fe, sino por protegerla de que llegase a ser dañada por algún error de otro usuarios, sin tener otra alternativa, porque no se tiene un sistema de información adecuado.

I.1.1 Situación Actual del Sistema

Las Aplicaciones examinan y recuperan los datos provenientes de las Bases de Datos, a partir de transacciones y operaciones que filtran, organizan y seleccionan los datos y los presenta en forma de información a los usuarios, proporcionándoles los medios para su toma de decisiones.

La información es un signo o conjunto de signos, que impulsan a la acción; se distingue de los datos porque éstos no son estímulos de la acción, sino simplemente cadenas de caracteres o patrones sin interpretar.

La toma de decisiones es un paso que conduce a la acción, se basa en la información, que puede definirse como datos organizados que reducen la incertidumbre en el momento de tomar decisiones.

I.1.2. Reglas del Negocio

La organización como sistema

Los objetivos de la organización son fijados por la gerencia cuando establece la misión básica, los objetivos estratégicos y los objetivos operativos.

Un concepto fundamental del enfoque de sistemas para la organización y la administración, es la relación recíproca de las partes o subsistemas de la organización.

El enfoque empieza con una serie de objetivos y se dedica al diseño del todo a diferencia del diseño de los componentes o subsistemas. Es decir, si se hace la planeación del desarrollo de cada una de las partes, de manera que logren sus objetivos; no es posible realizarla en forma independiente, pues aunque fuera óptima, no se está planeando el desarrollo de la empresa como un todo (enfoque sistemático).

Logrando un sistema que reúna una serie de etapas o pasos debidamente organizados para efectuar la presentación formal de datos o informes resumidos, preparados por una persona o grupo de ella, dentro de la institución para ayudar a la toma de decisiones en la institución.

Gerentes.

Son los responsables de la permanencia de la organización. Ejecutan actividades integradoras; son los encargados de la dirección, las misiones y los objetivos. Realizan los cambios de dirección para adaptar la empresa al ambiente. Constituyen asimismo el principal sistema de control. Establecen los métodos y procedimientos para asegurarse de que la organización consiga sus objetivos satisfactoriamente.

Colaboradores individuales.

Profesionales, técnicos, trabajadores de laboratorio o programadores. Los analistas de datos son personas que se ocupan de las transacciones o de la transmisión de datos. El personal operativo se encarga de tareas que son

primordialmente de carácter físico, como la operación de maquinaria, el control y el mantenimiento.

Elementos físicos.

Los elementos físicos son aquellos como el edificio, el equipo; apoyan los objetivos de la organización.

Subsistemas.

Organizaciones formales e informales que incluyen individuos, metas y relaciones.

Sistemas de información.

La gerencia establece los sistemas formales de información, los cuales son flujos, informes oficiales y medios físicos de la comunicación oficial. Los sistemas informales son la comunicación informal, contactos casuales o no planeados entre trabajadores y la comunicación rutinaria o ritualística en los grupos informales.

Sistemas funcionales o de proceso.

Patrones de flujo del trabajo, tareas y procedimientos establecidos formal o informalmente para lograr algún conjunto de objetivos relacionados.

Procesos de liderazgo.

Actividades que utilizan la motivación y logran la unidad organizacional en la búsqueda de los objetivos.

Planeación gerencial.

Preparación de planes escritos para la organización en su conjunto y para cada componente; exige integrar todas las actividades.

Control gerencial.

Supervisión y evaluación de rendimiento de la empresa, comparándola con los planes y estándares previamente establecidos. Imponer medidas correctivas si el desempeño se desvía demasiado de los planes y estándares.

Procesos de decisión.

La decisión debe tener en cuenta todos los resultados importantes de diversos cursos de acción. Deben tomarse las decisiones atendiendo a los intereses globales de la empresa, no sólo los del componente que afectan a corto plazo.

Se distinguen cuatro elementos que forman parte de una decisión: objetivos, información, predicción y evaluación.

Objetivos.- Se refiere a los elementos que indican hacia dónde se quiere ir o hacia dónde debe llevar la decisión de que se trate. Deberán llevar a metas que sean cuantificables, claras y reales.

Información.- Da a conocer a aquellos elementos que permiten entender la situación actual y evaluar la futura.

Predicción.- Se refiere al procedimiento que permite pronosticar o definir cuáles serán las posibles alternativas a seguir, así como los probables efectos sobre factores como costos y utilidades, organización y personal de la empresa.

Evaluación.- Se evalúan las predicciones considerando los objetivos fijados.

Funciones de los sistemas de la institución.

- Recolección de datos.
- Conversión de datos.
- Transmisión de datos.
- Almacenamiento de datos.
- Proceso de datos.
- Recopilación de información.

Toma de decisiones.

Una toma de decisiones totalmente objetivas y racionales requeriría información completa sobre todos los cursos posibles de acción y sobre sus resultados.

La capacidad humana de procesar información y los límites de la información disponible restringen el grado de racionalidad en la toma de decisiones (racionalidad limitada). La racionalidad limitada explica la toma de decisiones como: el examen secuencial de alternativas, heurísticas que limitan las opciones consideradas y la satisfacción suficiente.

Las heurísticas designan las reglas que reducen a unas cuantas las numerosas alternativas disponibles. La satisfacción suficiente es la búsqueda de la solución óptima que determina cuando alguna alternativa cumple con los requerimientos mínimos especificados.

La perspectiva de sistemas sobre la organización y la administración. En el enfoque de sistemas al diseño o análisis, es preciso identificar al sistema contestando antes las siguientes preguntas:

¿Cuáles son los objetivos del sistema?

¿Cuáles son los elementos y subsistemas?
¿Cuáles son los nexos o procesos integradores?
¿Qué es la estructura?

El sistema de información programado: consiste de aplicaciones que automatizan los documentos necesarios en las operaciones de oficina y decisiones de salida, anteriormente eran de naturaleza supervisora y que llevaban a cabo los seres humanos.

El componente de control del sistema de información se realiza por programa almacenado en la computadora, de manera que la toma de decisiones se logra automáticamente mediante los cálculos efectuados por la propia computadora.

El sistema de información de decisión: suministra información a los Directivos para que a base de la misma, puedan tomar decisiones. Esta información puede ser proporcionada independientemente, o bien, cuando hay una relación de hombre y máquina para la solución de problemas.

En este sistema se necesitan dar especificaciones para que el sistema proporcione las salidas requeridas por los Directivos, con la finalidad de que éstos tomen las decisiones adecuadas para satisfacer sus necesidades.

Si los sistemas de información de la Gerencia basados en computadoras están debidamente diseñados para la toma de decisiones, pueden constituir el adelanto que permitirá programar una cantidad cada vez mayor de decisiones administrativas, que ahora necesitan de tanto tiempo y esfuerzo.

I.1.3. ¿Por qué un Sistema Administrador de Aplicaciones?

El pronto, completo y confiable análisis de los datos es esencial, y se logrará si los usuarios tienen la información de manera disponible, confiable y clara en cualquier momento.

Ya se tiene una serie de Aplicaciones que hacen referencia a la información de manera confiable, pero a las cuales sólo tienen acceso unos cuantos usuarios, por lo que resulta común encontrarse esfuerzos duplicados, en la búsqueda y generación de la información.

El *Sistema Administrador de Aplicaciones* nace de la necesidad de información de los usuarios y se basa en la captación de información de los datos que

se tienen en la institución. Dicha información es generada por las Aplicaciones o subsistemas de información que apoyan a los diversos procesos que se realizan.

Un *Sistema Administrador de Aplicaciones* esencialmente reuniría un grupo de Aplicaciones que funcionan con objetivos comunes, logrando un sistema que une sus componentes.

Algunos aspectos importantes que deben contemplarse son:

- Debe administrar a los usuarios del Sistemas, apoyado en un esquema de privilegios.
- Debe contar con fuentes de datos únicas para evitar discrepancia.
- Debe mostrar información referente a los datos disponibles dentro de la institución de manera fácil y rápida.
- Debe proporcionar una estructura continua, formalizada y estructurada que reúne información, de las Aplicaciones disponibles, los usuarios de la institución y los privilegios que tienen para cada Aplicación. Se sirve de las mismas Aplicaciones ya desarrolladas.

El papel del *Sistema Administrador de Aplicaciones* consiste en informar a los usuarios que información se tiene disponible en la institución para que puedan hacer uso de ella evitando desarrollo de procesos similares.

El *Sistema Administrador de Aplicaciones* es un sistema que se sustenta en la relación que surge entre las personas, las computadoras y la información y se sirve de estos mismas elementos. Soportan un amplio espectro de tareas de la institución, más aún que los sistemas de procesamiento de datos, incluyendo el análisis y toma de decisiones.

Los usuarios del Sistema Administrador de Aplicaciones utilizan una *Base de Datos* compartida para tener acceso a la información.

Las Aplicaciones que se administrarán sirven para tomar decisiones, para lo cual analizan la información.

El *Sistema Administrador de Aplicaciones* explota una *Metainformación* (catálogos de usuarios, de aplicaciones, proyectos, temas, accesos, etc.), una serie de manuales y equipo de procesamiento de datos, que seleccionan, almacenan, procesan y recuperan datos (manipulación de datos y materia) para disminuir la incertidumbre de la información disponible, mediante el suministro único de información, en un tiempo más corto.

Algunos elementos importantes que maneja el *Sistema Administrador de Aplicaciones* son los adelantos en la tecnología del hardware y del software que han

hecho que tenga mayor importancia las *Bases de Datos, cuyo uso ha ido difundiéndose* a través del tiempo. El sistema proporcionara ahorro del tiempo en la inicialización, ya que será un sólo inicio para el acceso a todas las Aplicaciones.

El Sistema Administrador de Aplicaciones será capaz de administrar las aplicaciones que los usuarios utilizan y por *consiguiente a los mismos* usuarios. De manera tal que todos los usuarios pudiesen tener conocimiento de la información existente en las *Bases de Datos*.

La información de las aplicaciones se encuentra clasificada de manera que los *usuarios puedan entender el tipo* de información de la que se trata y se les proporcione si la necesitarán para sus cálculos. Evitando así tener que volver a obtener esa misma información por *cada uno de las oficinas* que lo requirieran.

Pero para seguridad de la información el *acceso deberá estar sujeto* a restricciones que el usuario responsable disponga, es decir, sólo se podrá accesar a esta información con autorización. Para lo cual se utilizará un esquema de privilegios, *tal que sólo puedan accesar* a cierta información permitida. Por lo que el sistema pedirá la identificación (clave de Red y su Password), con lo cual sabrá, cuáles son los atributos asignados *para cada Base de Datos*. La información (Usuarios, Sistemas, Atributos, Temas, etc.) de dicho sistema estaría en una *Base de Datos Relacional* (se obtendrá su diseño e implementación en el *capítulo II*)

Es por ello que el proyecto, esta encaminado a lograr un mayor control en *los accesos que cada usuario* tiene a cada uno de las aplicaciones internas. Esto se refiere a que cada aplicación tiene acceso a una *Base de Datos* específica, y sólo los usuarios responsables *de dicha información* pueden otorgar permisos para accesar a algunas Aplicaciones que realicen transacciones (Altas, Bajas y Cambios), por seguridad de la propia información.

Sin embargo uno de los objetivos del Sistema es la *difusión de la información*, por lo que se permitirá el acceso a todas las aplicaciones que realicen tan sólo consultas de la información, logrando con ello el objetivo de la difusión y la protección de la *información*.

A la *Base de Datos en la que se encontrará* la información, será necesario darle mantenimiento, administrarla para que los datos que contengan sean reales, sin embargo esta es una tarea un poco laboriosa por tener *que realizar conexiones* al servidor Sybase; una cierta prioridad al momento de ir dando de alta nuestros datos (como Altas de Usuarios, Altas de Aplicaciones, Asignarles atributos a cada usuario *para cada aplicación* especifica, o alguna otra), por lo que realizarlas manualmente resultaría poco práctico. Entonces se plantea realizar un sistema que facilite dicha tarea (se implementará en el *capítulo III*).

Una vez implementado el ambiente adecuado *se iniciará* la realización *del Sistema Administrador de Aplicaciones* con las características que ya se han mencionado anteriormente (se implementará en el capítulo IV).

Como los datos se encontrarán en una *Base de Datos* en una plataforma Sybase se sugiere realizar las aplicaciones tanto del *Administrador de la Base de Datos* como del *Sistema Administrador de Aplicaciones* en una arquitectura Cliente-Servidor, ya que por sus características de procesamiento de la información en el servidor remoto, de manera tal que al transmitir el resultado del procesamiento sólo traería la información mínima requerida, resulta una gran ventaja en tiempo.

Las ventajas que se han encontrado al elegir este diseño es el hecho de que en la institución se tiene las herramientas necesarias tanto en Cliente como en Servidor, se considera que las herramientas son potentes y de gran velocidad, buena interfaz gráfica con el usuario y con sofisticadas funciones para realizar nuestras consultas y realizar informes, en particular se requiere de un servidor de una gran capacidad de almacenamiento y de gran seguridad. Por otro lado el Cliente será VisualFox que maneja inicio de sesiones, sintaxis SQL (Structured Query Language) nativa del servidor, realiza vistas remotas, por lo que podrá acceder a los grandes almacenes de datos que están disponibles en las *Bases de Datos* del servidor Sybase. También puede aprovechar las capacidades de seguridad y de procesamiento transaccional del servidor remoto.

Podemos tener la seguridad que las herramientas nos permitirán:

- Utilizar la funcionalidad específica del servidor, como procedimientos almacenados y funciones intrínsecas basadas en el servidor.
- Disponer de absoluto control sobre las instrucciones Update, Delete e Insert de paso a través de SQL.
- Disponer de mayor control sobre transacciones remotas.
- Solicitar múltiples conexiones para un origen de datos.
- También trabajar con múltiples orígenes de datos ODBC(Open Data Base Connectivity) solicitando una conexión con cada origen de datos al que desea acceder.

I.2 Marco Teórico

I.2.1¿Qué es un Sistema?

La palabra sistema es posiblemente el término más sobreutilizado y del que más se ha abusado en el léxico técnico. Hablamos de sistemas políticos y educativos, de sistemas de aviación y de fabricación, de sistemas bancarios y de ferrocarril. La palabra nos dice poco. Usamos el adjetivo que la describe para comprender el contexto en el que se usa. El diccionario Webster la describe así:

1.un conjunto u ordenación de cosas relacionadas de tal manera que forman una unidad o un todo orgánico; 2.un conjunto de hechos, principios, reglas, etc. clasificadas de tal manera que muestran un plan lógico uniendo las diferentes partes; 3. Un método o plan de clasificación y ordenación; 4. Una forma establecida de hacer algo; un método; un procedimiento.

Tomando prestada la definición anterior del diccionario Webster, definimos un sistema basado en computadora como.

Un conjunto u ordenación de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento de información.

Los elementos de un sistema basado en computadora incluyen los siguientes elementos:

Software: los programas de computadora, estructuras de datos y documentación asociada que sirven para realizar el método lógico, procedimiento o control requerido.

Hardware: los dispositivos electrónicos (por ejemplo, CPU, memoria) que proporcionan la capacidad de computación y los dispositivos electromecánicos (por ejemplo, sensores, motores, bombas) que proporcionan las funciones del mundo exterior.

Gente: los individuos que son usuarios y operadores del software y del hardware.

Bases de Datos: una colección grande y organizada de información a la que se accede mediante el software y que es una parte integral del funcionamiento del sistema.

Documentación: los manuales, impresos y otra información descriptiva que explica el uso y/o la operación del sistema.

Procedimientos: los pasos que definen el uso específico de cada elemento del sistema o el contexto procedimental en que reside el sistema

Una característica compleja de los sistemas basados en computadoras es que los elementos que componen un sistema pueden también representar un macro elemento de un sistema todavía mayor. Un macro elemento es un sistema basado en computadora que forma parte de un sistema basado en una computadora mayor.

I.2.2. Lenguajes de Cuarta Generación

A lo largo de la historia del desarrollo del software siempre se ha intentado generar programas de computadora cada vez con mayores niveles de abstracción. Esto se logra con las nuevas generaciones de los lenguajes de computación.

Se han creado varias herramientas para ayudar a que el desarrollo de los sistemas de información sea más rápido, barato y de mayor calidad.

La Primera Generación se inicia, con el establecimiento del lenguaje de máquina con una comunicación hombre-máquina, se "escribía" directamente el código binario de cada instrucción a realizar; casi simultaneo surge el lenguaje ensamblador, utilizando mnemónicos (palabras cortas cuyo significado alude a alguna acción específica, como ADD, MOVE y otras) como forma de interacción con la máquina.

La Segunda Generación inicia con la elaboración de lenguajes como Fortran, Cobol, Algol y Basic, cuyo uso era más específico y más sencillo. Todos ellos requieren el uso de algún compilador o intérprete que traduce el código del programa a lenguaje de máquina.

La Tercera Generación se inicia con la llegada de los lenguajes conocidos como Lenguajes de Alto Nivel, que permiten el uso de palabras más extensas y por lo que son más comprensibles para las personas, además de ampliar sus áreas de aplicación; algunos ejemplos de estos lenguajes son Pascal, C, Ada, Modula-2, Lisp, Prolog, APL y Smalltalk. Estos lenguajes cuentan con archivos de ayuda y editores con más funciones.

Los lenguajes de alto nivel se subdividen en dos grandes áreas: lenguajes de propósito general, cuando éstos se usan para generar programas de cualquier área de aplicación; y especializados, cuando se centran sobre una área en específico, por ejemplo LISP o Prolog que son usados dentro del área de la Inteligencia Artificial.

A principios de los años ochenta se crearon los lenguajes de Cuarta Generación los cuales fueron diseñados para la solución de problemas más específicos o precisos; algunas de sus características principales son: lenguajes interactivos, proporcionando varios medios para su uso, como ayuda en línea; instrucciones sencillas para acciones muy recurridas; actividades para el seguimiento de programas, tales como *Trace* y *Debug*; generan aplicaciones y se aprenden rápidamente; pueden realizar algún tipo de suposiciones inteligentes, esto es, elegir algunas soluciones cuando algunos aspectos o situaciones no le son especificados; permiten especificar la salida deseada sin tener que describir detalladamente los pasos que producen dicho resultado. Visual Fox es de este tipo ya que aunque maneja Wizard para construir la interfaz grafía, los métodos son tecleados aún con programación procedural.

En los inicios de los años 90's surgen los lenguajes de quinta generación 5GL que son programas desarrollados con una interfaz visual o gráfica. Para crear lenguajes de quinta generación es usual utilizar compiladores de lenguajes de 3GL o 4GL. Microsoft, Borland, IBM, y otras compañías desarrollan productos visuales con lenguajes de quinta generación, un ejemplo es el desarrollo de las aplicaciones en Java.

La programación visual te permite crear fácilmente jerarquía de clases orientadas a objetos y arrastrar iconos para ensamblar los componentes del programa. Microbrew AppWare y IBM VisualAge para Java son ejemplos de lenguajes de quinta generación.

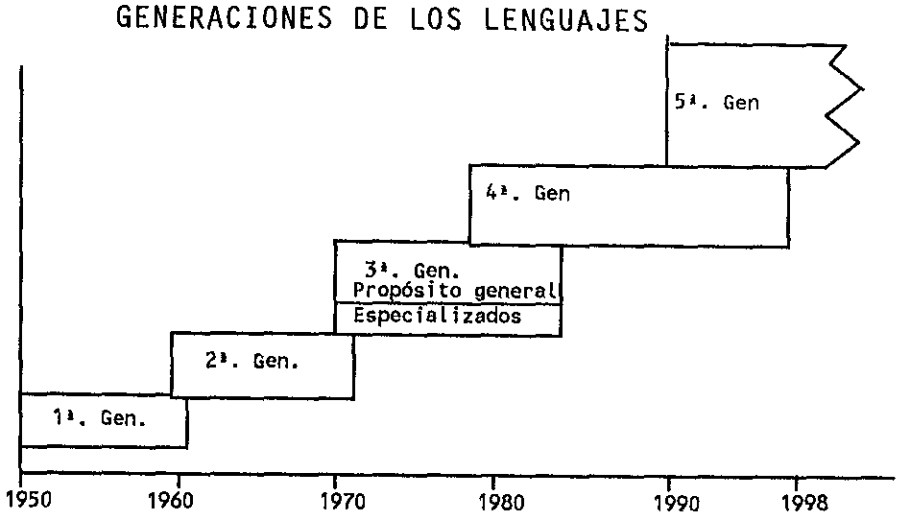


FIG. I.12. Evolución de los Lenguajes de Programación, dividida en generaciones.

Los lenguajes de Cuarta Generación han sido divididos en cuatro áreas de aplicación:

1. Aplicaciones de Bases de Datos.

Los lenguajes de este tipo interactuan en gran parte con un Sistema Administrador de Bases de Datos o *DBMS* (de sus siglas en inglés *Data Base Management System*) para manipulación, almacenamiento y recuperación de datos; existen dos modalidades en esta división:

Lenguaje de consulta que se enfocan a los lenguajes que proporcionan un conjunto de comandos muy simples que ayudan a un usuario no experto a accesar la Base de Datos fácilmente.

Generadores de reportes, que facilitan la elaboración de informes sencillos sin tener que conocer el funcionamiento del Sistema Manejador de Bases de Datos (DBMS), sino sólo un poco del lenguaje que nos facilitará la elaboración del informe.

2. Generadores de Programas.

Los generadores de programas proveen de los elementos para construir aplicaciones que involucran el software de la empresa y facilita la integración de la información como un sólo conjunto de datos. Normalmente tienen aplicación en procedimientos administrativos. Para aclarar el área que cubren éstos, hablaremos de

alguna empresa que cuenta con paquetes de software, tales como procesadores de texto, hojas de cálculo y otro tipo de herramientas, como paquetes administrativos; un generador de programas nos permitirá, por ejemplo, tomar la información procesada en la hoja de cálculo, llevarla a otra herramienta que genere un reporte gráfico con dicha información y permitir insertarla o agregarla dentro de un documento que se esté elaborando en el procesador de texto; manejado ello como un todo, sin tener que hacer los cambios manualmente de un paquete a otro; ya que el sistema se encargará de realizarlo de manera automática e invisible al usuario.

3. Soporte en la Toma de Decisiones.

Algunos lenguajes proporcionan información que ayuda a los gerentes o administradores en la tarea de tomar decisiones. Los lenguajes que cubren estos aspectos son denominados, bajo los términos de sistemas de información, lenguajes de planeación financiera, y en ellos se encuentran implementados los procedimientos matemáticos, estadísticos y de pronóstico; estos lenguajes forman parte de un tipo de sistema de información conocido como Sistema para el Soporte de Decisiones o DSS (Decision Support Systems).

4. Generadores de Prototipos o Aplicaciones.

Las herramientas para el desarrollo de nuevos sistemas o aplicaciones, son fáciles y rápidas y permiten ver la reacción de los usuarios finales ante estas ventajas. Los lenguajes para la creación de prototipos dan lugar a un desarrollo iterativo, en el que un prototipo se refina de manera sucesiva. Se apoyan en funciones preprogramadas en distintos módulos, y toman lo necesario, aquellas funciones que nos sirvan y las integran en un prototipo o versión experimental del sistema, de manera que se irá ajustando a las necesidades específicas del usuario hasta obtener el sistema que las satisfaga en su totalidad. Generalmente las herramientas CASE (Computer Aided Software Engineering) son contempladas dentro de este tipo de lenguajes.

Por las características que ofrecen los lenguajes de Cuarta Generación, como son ambientes gráficos, simplicidad en su uso y programación rápida de aplicaciones, el sistema se desarrollará en uno de estos lenguajes.

I.2.3. Bases de Datos Relacionales

I.2.3.1. Términos Básicos.

Existen algunos términos técnicos que son utilizados frecuentemente en las Bases de Datos, estos términos son definidos a continuación con el fin de lograr uniformidad y claridad en esta tesis.

Datos.

Un dato se define como la unidad mínima de información que se considera isomorfa, entendiendo con ello que conserva una estructura específica.

La descripción de los datos y de las relaciones que entre ellos existen adopta una de dos formas, dependiendo del ámbito donde son empleadas: *Lógica*, en donde el dato se define de acuerdo a la forma que éstos tienen para el usuario y/o aplicación que los ocupa; *Física*, que comprende la forma en que son almacenados físicamente (es decir, a nivel hardware). Para el usuario no es evidente que los registros, dentro de la unidad de disco no están en localidades contiguas de memoria auxiliar, pero él los maneja de dicha forma.

Byte.

El byte es un conjunto de bits (unidad mínima de representación de datos cuyos valores son 1 ó 0), generalmente 8 de ellos; mediante los cuales la computadora representa todos los datos; normalmente equivalente a un símbolo o carácter.

Item de datos.

Es el grupo de datos más pequeño que puede ser manejado de esa forma, tiene un número variable de bytes; se conoce más comúnmente como *campo o item de datos*; por ejemplo NOMBRE, INSTITUCION.

Agregado de datos.

Es una colección de campos de datos dentro de un registro, al que se nombra como un todo; por ejemplo FECHA puede ser un agregado de datos compuesto por los campos DIA, MES Y AÑO.

Registro.

Es una colección de campos y agregados de datos que representa una unidad de información, esto es, nos muestra los datos requeridos para un elemento específico e individual pero con los valores de campos únicos para él; por ejemplo un registro de información por cada institución, cada registro tiene la misma estructura de campos, pero con diferentes valores de contenido.

Archivo.

El archivo es el conjunto de registros, en donde éstos últimos comprenden todos el mismo número de campos. Se dice que es un conjunto de datos almacenados en una estructura bien definida; por ejemplo archivo del formulario X, archivo de datos de captación, etc.

I.2.3.2. Sistema Administrador de Bases de Datos DBMS (Data Base Management System).

Es el software (o conjunto de programas de computadora) que nos va a permitir la manipulación de nuestros datos, localizados dentro de una Base de Datos (conjunto de datos), en un entorno eficiente y conveniente dentro del cual nuestra información puede ser modificada, borrada, recuperada, etc., en forma segura, rápida y consistente. Los programas de la Base de Datos se subdividen en:

Administradores: en esta división se tienen los procedimientos para el manejo y control de los datos como tal, sin interferir o modificar su contenido o valor. Dentro de este tipo se realiza una definición de las estructuras de datos y sus relaciones, esto es, la forma de almacenamiento de nuestros datos y las relaciones que guardan entre sí.

De explotación: en esta sección se contempla el desarrollo de sistemas y/o programas que manipularán los valores de nuestros datos, como lo son los procedimientos de altas, bajas y cambios, que son de uso general.

Estos sistemas se diseñan de manera tal que permiten un manejo de información muy grande; contemplando en ello, la estructura y manipulación de los datos, en forma segura, incluyendo autorizaciones de acceso a la Base de Datos y respaldo de la misma.

Debido a la importancia de la información dentro de todas las organizaciones; la *Base de Datos es un recurso* valioso, lo cual ha llevado al desarrollo de conceptos y técnicas de manejo de información en forma rápida, eficiente y segura.

El sistema de manejo de *Bases de Datos* permite también el desarrollo de aplicaciones o programas propias a las necesidades de la organización que lo emplea; por ejemplo, en una empresa bancaria pueden desarrollarse sistemas para hacer cargos y abonos a una cuenta, agregar cuentas, obtener saldos y generar los estados de cuenta; la conveniencia que ello representa es que al desarrollarse las aplicaciones dentro del mismo manejador de Bases de Datos, podemos tener programas similares desarrollados con un mismo lenguaje de programación.

Su objetivo es registrar y mantener datos relevantes a la organización, necesarios en los procesos de toma de decisiones; contempla funciones, rutinas y métodos de acceso, áreas de almacenamiento, de trabajo y de control, requerido para el tratamiento de la información.

Las funciones del sistema manejador de la *Bases de Datos* (DBMS *Data Base Management System*) son:

- Definición de los datos.
- Manipulación de los datos.
- Administración del almacenamiento de los datos.
- Integridad de la información.
- Seguridad de datos.
- Proporcionar herramientas como son utilerías para cargar y respaldar, monitoreo, ayudas de diseño; diccionario de datos; reporteador, etc.

I.2.3.3. Estructuras de Datos.

Una estructura de datos se define partiendo de un concepto importante: para el desarrollo de programas se tienen dos elementos, en primer lugar las estructuras de datos, que son las diferentes formas en que almacenaremos y representaremos nuestros datos dentro de la computadora; y los algoritmos, que son los procesos por los que serán manipulados nuestros datos, o el conjunto de pasos mediante el cual serán modificados para cumplir con el objetivo del programa desarrollado.

Entonces, una estructura de datos es la forma de almacenamiento de nuestros datos dentro de la computadora para ser utilizados por un programa, este almacenamiento es diseñado para tener mayor eficiencia.

La forma elemental de una estructura de datos es la variable o elemento básico, esta se define conforme a un tipo básico, el cual le brinda sus características y propiedades; esto es, una variable es un tipo de dato que un sistema o programa de computadora maneja de manera general, por ejemplo:

variable = integer, variable = real, variable = string

Que dentro de la computadora tienen ya una longitud en bytes predefinida, es decir, que al ser de uso común ya están definidos en el lenguaje de programación.

La estructura siguiente es el *registro*; definiéndose como un conjunto de tipos básicos (variables) denominados *campos del registro*, de manera que la longitud del registro se define por su número de campos, por ejemplo:

Nombre del Registro:	Campos		
Datos_Personales:	Nombre	Dirección	Teléfono
Tipo de Dato:	String	String	Num

En donde los campos se han definido como variables del tipo correspondiente, nuestro registro es Datos_Personales y cuenta con tres campos: Nombre, Dirección y Teléfono, en donde éstos son variables, nombre y dirección de tipo string y teléfono es un numérico.

Es posible definir, dentro de un registro, campos que a su vez, sean también registros (agregado de datos).

El siguiente en la lista es el *Archivo, constituido por registros* y almacenado en memoria auxiliar (discos, cintas, etc.).

Un *arreglo* agrupa elementos de un sólo tipo, por decir, de variables enteras, reales, caracteres, etc.; así como arreglos de registros. De manera general un arreglo es una lista de datos de un mismo tipo. *Un arreglo puede ser:*

- unidimensional (vector).
- multidimensional (matrices *desde 2 hasta n dimensiones).*

Un arreglo es almacenado en memoria real (internamente); puesto que el acceso a esta memoria (que también denominaremos memoria RAM) es de 1,000 a 100,000 veces más rápida que un acceso a memoria auxiliar (como son las unidades de disco o cinta, por ejemplo).

Una lista es una estructura de datos almacenada en memoria real que puede ser vista de forma análoga a un vector. En este caso nuevamente aplicamos los conceptos de vista lógica y física aplicadas a los datos.

Es evidente que las estructuras básicas van conformando estructuras de nivel superior, y éstas, a su vez, son elementos de otras más complejas de forma tal que se llega a una superestructura que engloba todas las anteriores en forma dinámica, no referimos a la *Base de Datos.*

I.2.3.4. ¿ Qué es una Base de Datos ?

En ocasiones una *Base de Datos* se concibe como un enorme receptáculo contenedor en el que un organismo guarda todos los datos procesables que reúne y al cual acuden muy diversos usuarios requiriendo dicha información (por ejemplo los archivos generados en *LOTUS* o en *FOX PRO* que son almacenados en un directorio de

la red y cuya información contenida puede ser accesada por los usuarios de nuestra institución). Una *Base de Datos* corresponde a una estructura mucho más compleja que implica la existencia de ciertas características, que a lo largo de la tesis se irán dando a conocer.

Un buen concepto inicial de lo que es una Base de Datos nos dice:

Una Base de Datos es una colección de datos almacenados e interrelacionados en conjunto sin redundancias perjudiciales o innecesarias; su finalidad es la de servir a una aplicación o más, de la mejor manera posible. Los datos se almacenan (o se estructuran) de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir, para modificar o extraer los datos almacenados.

La idea básica en la implantación de una Base de Datos es la de aprovechar los mismos datos para tantas aplicaciones como sea posible. Dicha *Base de Datos* permitiría no sólo la lectura de los datos almacenados, sino la continua modificación de los que son necesarios para el control de las operaciones. Asiendo posible la inspección de la *Base de Datos* en busca de las respuestas a los interrogantes que se plantean o de información para fines de planteamiento. La misma colección de datos servirá para varios departamentos, posiblemente sin tener en cuenta algunas fronteras administrativas; inclusive controlando los accesos concurrentes (varios usuarios al mismo tiempo).

La Base de Datos debe prestarse a una fácil reestructuración siempre que haya que agregarle nuevos tipos de datos o utilizarla para nuevas aplicaciones. Esta reestructuración no debe originar la necesidad de volver a escribir los programas de aplicación y, en general, no debe ser fuente de trastornos.

La facilidad con que pueda modificarse la Base de Datos tendrá siempre un efecto directo sobre la capacidad de desarrollar nuevas aplicaciones del procesamiento de datos dentro del organismo que la explota.

A menudo se habla de *independencia de datos* como uno de los atributos destacados de la *Base de Datos*. Esta idea implica que los datos y los programas de aplicación que de ellos se sirven son mutuamente independientes, de manera que unos u otros puedan ser modificados sin afectar los restantes.

Los datos deben ser independientes también de su forma de almacenamiento; esto es, los datos pueden ser almacenados en diversos dispositivos diseñados para ello, pero la percepción que un usuario tenga de estos datos, debe ser independiente de esta forma de almacenamiento.

La Base de Datos esta organizada de modo que los datos que contiene pueden ser alcanzados de muchas maneras diferentes (que son las vistas lógicas) y ser utilizados para responder a una diversidad de interrogantes.

Lo primero es entonces, almacenar para cada entidad todos los datos de interés, no sólo los que se necesitan para una determinada aplicación.

Para el futuro inmediato, las Bases de Datos más valiosas serán probablemente las que se estructuran para satisfacer conjuntos prescritos de operaciones, en vez de las que se prevean para una búsqueda no estructurada.

Las características deseables de la Base de Datos son:

- Consistencia, esto es, tener una sola versión válida de los datos.
- No redundancia, los datos no deben repetirse (con excepciones).
- Integridad, la información se encuentre correcta y completa.

I.2.3.5. Objetivos de los Sistemas de Bases de Datos.

De alguna manera estos objetivos se plantearán con base a las desventajas que presentan los sistemas de procesamiento de archivos; de donde las más relevantes son:

- Redundancia e inconsistencia de los datos: puesto que los programas se desarrollan por diferentes programadores es posible tener datos repetidos en varios archivos; esta redundancia aumenta costos de almacenamiento y tiempos de acceso; es posible además, tener datos distintos (no actualizados) dentro de esta información redundante por lo que nuestra información es inconsistente y además poco confiable.

- Dificultad para tener acceso a los datos: cuando es requerida una información específica sobre algunos registros de la Base de Datos, si no existe el programa de aplicación que la extraiga, se requerirá desarrollarlo, o en su defecto, será necesario extraerla manualmente; y ninguna de estas dos opciones nos brinda la eficiencia y rapidez requeridas.

- Aislamiento de los datos: puesto que los datos se almacenan en diversos archivos, éstos pueden tener diferentes estructuras por lo que para una nueva aplicación es difícil obtener los datos apropiados.

- Usuarios múltiples. muchos sistemas permiten que la información se actualice en forma simultánea, no existe una forma de controlar la concurrencia en la modificación de la información, de tal manera que esta se vuelve inconsistente.

- Problemas de seguridad: en sistemas de procesamiento de archivos es difícil imponer y controlar las restricciones en el acceso de la información a los distintos usuarios; esto es de vital importancia en sistemas tales como los bancarios, en

donde información confidencial no debe ser alterada por personas ajenas al sistema; no todas las personas deben tener acceso a toda la información y mucho menos a poder modificarla.

- Problemas de integridad: los valores de los datos que se guardan en la Base de Datos deben satisfacer ciertos tipos de limitantes de consistencia; el sistema debe verificar el cumplimiento de estas limitantes; en ocasiones el agregar las limitantes a los programas de aplicación se vuelve una tarea compleja y poco práctica.

En forma concreta, los objetivos primarios de las Bases de Datos son:

- Uso múltiple de datos.
- Al modificar la Base de Datos no será necesario modificar las aplicaciones.
- Bajo costo.
- Menor proliferación de datos.
- Desempeño.
- Claridad.
- Facilidad de uso.
- Flexibilidad.
- Rápida atención de interrogantes no previstos.
- Facilidad para el cambio.
- Precisión y coherencia.
- Reserva.
- Protección contra pérdida o daño.
- Disponibilidad.

Los objetivos secundarios que persigue:

- Independencia física y lógica de datos.
- Redundancia controlada.
- Adecuada rapidez de acceso y exploración.
- Normalización de los datos dentro de un organismo.
- Diccionario de datos.

- Interfaces de alto nivel con los programadores.
- Lenguaje del usuario final.
- Controles de integridad.
- Fácil recuperación en caso de fallo.
- Ayudas para el diseño y la supervisión.
- Migración o reorganización automática.

Es por todo ello que los sistemas de Bases de Datos ofrecen mecanismos que nos permiten cumplir con todo esto y más; sobre todo de forma rápida y mucho más eficiente. El "Sistema Administrador de Bases de Datos" nos ofrece técnicas para el control de seguridad e integridad, y en ocasiones hasta métodos para eliminar información redundante e inconsistencia.

Las ventajas que ofrece el enfoque de Bases de Datos son:

- Información centralizada en donde todos los datos necesarios se almacenan dé tal forma que la información será única y será accesada por todos los usuarios y sus programas.
- Reduce redundancia.
- Evita inconsistencia.
- Conserva la integridad.
- Se aplican restricciones de seguridad.
- Facilita el uso múltiple de la información (distribución paralela).
- Reduce tiempo de desarrollo y mantenimiento de las aplicaciones.
- Independencia de datos (inmunidad de las aplicaciones a los cambios de la estructura de almacenamiento y del método de acceso).

I.2.3.6. Usuarios de la Base de Datos.

*Soporte Técnico:

- Instalan y dan mantenimiento al software del DBMS
- Controlan el espacio de almacenamiento.
- Solucionan problemas de software.

*Operación:

- Inicia la operación diaria del DBMS
- Monitoreo de la operación.
- Montar y desmontar cintas y discos.

*Administrador de la Base de Datos:

- Diseño y definición de la Base de Datos.
- Monitoreo del desempeño.
- Respaldo y recuperación de la información.
- Definición de esquemas de seguridad.

*Programador o analista de aplicaciones:

- Cargar datos.
- Escribir programas con el lenguaje de programación que proporciona el DBMS.
- Definir modelos de datos.

*Usuario final:

- Analizar resultados.
- Ejecutar programas.
- Definir consultas y reportes.

*Administrador de datos:

- Planeación estratégica de la información.
- Desarrollo del modelo conceptual de la Base de Datos.
- Coordinador.

I.2.3.7. Diseño de la Base de Datos.

Modelos de Datos.

Un modelo consiste en unidades lógicas de datos (entidades) y expresa las relaciones entre ellas de acuerdo con la interpretación del mundo real. En el entorno de Bases de Datos, una entidad se define como un objeto distinguible a partir del cual se puede registrar información.

El modelado de datos abarca tres aspectos: El *Modelo Conceptual*, en el que se indica el modelo de la organización; es independiente del DBMS y es la herramienta de comunicación entre los usuarios de la Base de Datos. El *Modelo Lógico* es la traducción del modelo conceptual en función del DBMS; los más usados son los modelos reticulares o de red, jerárquico y relacional, siendo este último el más importante. El *Modelo Físico* abarca la distribución de datos, métodos de acceso y técnicas de indexación.

Modelos Lógicos.

A) Modelo Jerárquico:

Este se basa en el concepto de árboles manejado en las estructuras de datos. El modelo jerárquico representa siempre una relación de uno a varios.

Existen manejadores comerciales que usan este modelo, debido a que es relativamente simple y fácil de usar, reduce la dependencia de los datos.

Sus desventajas: no permite implementar fácilmente relaciones varios a varios, las operaciones de inserción y supresión son extremadamente complejas, sus comandos son de procedimiento, sólo podemos accesar a un nodo hijo sólo a través de su padre.

B) Modelo de Red o Reticular.

Permite modelar correspondencias muchos a muchos; este modelo interconecta las entidades en una red. La diferencia significativa entre el modelo jerárquico y el de red estriba en que en este último, cualquier tipo de registro puede participar en cualquier número de conjuntos, jugando el papel de un tipo de registro propietario o miembro.

Sus desventajas: es complejo la reorganización de la Base de Datos, es posible perder la independencia de los datos.

C) Modelo Relacional.

En este modelo, las relaciones son visualizadas como tablas, esto es: Una tabla es una relación (archivo); un atributo, es una columna (campo); y una "combinación" de estos campos, conocido también como tupla, es un renglón (registro).

Las propiedades de las relaciones o tablas son:

1. Cada entrada de la tabla representa un campo de datos, es decir, no hay grupos repetitivos y cada renglón tiene el mismo número de campos.
2. Las tablas son homogéneas por columna.
3. Cada columna tiene un nombre propio.
4. Cada renglón tiene un identificador único.
5. El orden de los renglones no es significativo.
6. El orden de las columnas no es significativo.
7. El producto de una operación relacional siempre será una relación.

El modelo relacional está basado en una serie de tablas. El usuario puede consultar estas tablas, insertar nuevas "tuplas", eliminarlas y actualizarlas.

Los lenguajes de consulta relacionales permiten realizar consultas sobre nuestros datos sin tener que desarrollar un programa para ello, cuentan con un conjunto de procedimientos para realizar algunas consultas sencillas de forma directa; en caso de algunas más complicadas, permiten entonces sí, su uso dentro de un programa.

Estos lenguajes son concisos y formales y no son apropiados para usuarios casuales del sistema de *Base de Datos*; es por ello que los sistemas comerciales han empleado otro tipo de lenguajes, donde se incluyen construcciones para la actualización, inserción y eliminación de información; los lenguajes comerciales más conocidos y empleados entre otros son: SQL, Quek y QBE.

Los diferentes usuarios de una *Base de Datos* compartida pueden aprovechar vistas lógicas individualizadas de la *Base de Datos*.

La normalización es un proceso que se aplica a nuestro sistema de *Base de Datos* al diseñarlo para eliminar las dependencias funcionales, es decir, las limitantes del conjunto de relaciones legales; es un proceso de paso a paso que permite reemplazar relaciones entre datos con relaciones de la forma plana bidimensional (creación de tablas); las tablas deberán organizarse de tal manera que no se pierda ninguna de las relaciones existentes entre datos.

La normalización atañe a las estructuras lógicas - las vistas de los datos propias del usuario - y no a la forma como los datos se representan físicamente.

Las ventajas de la normalización son: simplicidad, facilidad de consultas no planeadas, independencia de datos, fundamentos teóricos útiles. La principal y quizá única desventaja es que genera un desperdicio de memoria secundaria

Las ventajas de representación de datos en tercera forma normal, esto es, una representación en la que no se repiten registros, se tienen uno o varios campos que forman nuestra 'llave' de acceso a cada registro, donde los campos restantes son dependientes completos de nuestra llave, y si además no existen dependencias indirectas (transitivas) de nuestros atributos o campos con respecto a otros; son :

1. Facilidad de uso a través de las tablas.
2. Flexibilidad, permitiendo fácilmente obtener la información requerida de la Base de Datos por medio de operaciones de álgebra relacional.
3. Precisión: las relaciones (tablas) tienen un significado preciso y son manipuladas adecuadamente por el álgebra relacional.
4. Seguridad: pueden trasladarse atributos sensibles a una relación aparte, con sus propios controles de autorización.
5. Relacionabilidad: permite relacionar diferentes archivos o conjuntos de tuplas de forma flexible.
6. Facilidad de implementación.
7. Independencia de datos fácil de lograr.
8. Lenguaje para la manipulación de datos, basado en estructuras del álgebra y cálculo relacional.
9. Claridad.

I.2.3.8. Bases de Datos Distribuidas.

Este concepto indica que los datos se almacenan en varias computadoras (por ejemplo una red); un sistema de este tipo consiste en un conjunto de localidades, cada una de las cuales puede participar en la ejecución de transacciones que accesen a datos de una o varias localidades, la principal diferencia con respecto a las Bases de Datos Centralizadas es que en estas últimas los datos residen en una sola localidad y en las distribuidas, se encuentran en varias.

Cada localidad esta consciente de la existencia de las demás y cada una permite ejecutar transacciones locales y globales.

Las ventajas de las bases de datos distribuidas son:

Capacidad para compartir y accesar la información de manera eficiente y confiable: cada administrador local podrá tener un grado de autonomía diferente, que se conoce como autonomía local.

Confiabilidad y disponibilidad: en caso de fallas es probable seguir trabajando.

Agilización del procesamiento de consultas: es posible subdividir las consultas en subconsultas que se ejecuten en paralelo en distintas localidades.

Las desventajas son:

Mayor complejidad para garantizar una coordinación adecuada entre localidades; esto se refleja en: costo de desarrollo de software, mayor posibilidad de errores, mayor tiempo extra de procesamiento.

A) Diseño de Bases de Datos Distribuidas.

Un sistema puede ocultar los detalles de la distribución de la información en la red; denominada *transparencia de la red*; ésta se relaciona con la autonomía local, pues es el grado hasta el cual los usuarios del sistema pueden ignorar los detalles del diseño del sistema distribuido.

La transparencia y autonomía comprenden: nombres de los datos, repetición y fragmentación de datos (que se explican a continuación), y localización de los fragmentos y copias.

Repetición: el sistema mantiene varias copias idénticas de la relación, cada una en una localidad diferente.

Fragmentación: dividimos la relación en varios fragmentos almacenados en diferentes localidades.

Repetición y Fragmentación: dividimos nuestra relación en fragmentos y almacenamos varias copias de esos fragmentos.

Una transacción debe cumplir con la propiedad de atomicidad (se ejecutan todas las instrucciones de la transacción o no se ejecuta ninguna). El manejador de transacciones tiene como función asegurar que estas conserven su atomicidad; el manejador de transacciones es local.

Para comprender como puede estructurarse un manejador de este tipo, se define un modelo abstracto para un sistema de transacciones. Cada localidad contiene:

- Manejador de transacciones: ejecución de transacciones que accesan los datos de esa localidad.
- Coordinador de transacciones: Coordina la ejecución de éstas.

Cada manejador se encarga de mantener una bitácora para la recuperación; participar en un esquema de control de *concurrencia apropiado* para coordinar la ejecución en paralelo de las transacciones que se ejecuten en esa localidad. El coordinador debe *iniciar la ejecución de la transacción* y coordinar la terminación de la transacción.

Las fallas que pueden presentarse en este tipo de bases de datos son, *además de los normales* (como son falta de almacenamiento, etc.):

- Falla total de una localidad.
- Interrupción de una línea de *comunicaciones*.
- Pérdida de mensajes.
- Fragmentación de la red.

Como es de esperarse, es difícil detectar cuál de estas fallas es la que se *presentó en un momento* dado, pero existen mecanismos que nos permiten conocer la causa.

B) Control de Concurrencia.

En primera instancia, se cuenta con los modos de candado *compartido* y *exclusivo*. *Un candado compartido* permite hacer una lectura de un dato, pero no puede escribirlo o modificarlo y en un candado de modo exclusivo, el dato puede ser *leído* y escrito por la transacción.

Las diferencias entre un esquema de trabajo en una red LAN (o de área local) empleando un DBMS (manejador de sistemas de *Bases de Datos*) y un *Servidor*:

En un *Back End* estamos realizando la *selección de los datos* requeridos en nuestro servidor (denominaremos servidor en este caso, a la localidad dentro de la red de donde estamos tomando *nuestros datos*), y *en nuestra* terminal, en el *Front End*, tenemos la presentación apropiada del resultado de la transacción ejecutada.

A diferencia del esquema en donde todos los datos son transmitidos y luego *se realizan las transacciones* u operaciones correspondientes para obtener la información deseada; hay que mencionar que aunque este método resulta menos *práctico* (aparentemente) tiene sus aplicaciones, por ejemplo cuando no contemos con sistemas de control de concurrencia adecuados, o cuando se requieren realizar un número muy grande de transacciones sobre el mismo conjunto de datos, pues al tener nuestros datos completos, el tiempo empleado en *la ejecución de estas* operaciones será mucho menor; pero considerando también que no somos el único usuario y que debemos permitir el uso de los *datos a los demás*.

I.2.3.9. SYBASE.

Este manejador fue diseñado con una arquitectura cliente servidor, propia de las redes; el servidor SQL opera bajo una conexión de este tipo y contiene tanto la *Base de Datos* como el manejador; el cliente realiza solicitudes de acceso a esos datos.

Servidor Sybase SQL.

Se encarga del manejo de múltiples *Bases de Datos* y multiusuarios, mantiene el mapeo de la descripción lógica al almacenamiento físico de datos, empleando la memoria RAM. Compila y ejecuta instrucciones de T-SQL y regresa los resultados a las aplicaciones de los clientes.

El manejo multiusuario lo realiza el servidor y no el sistema operativo; con ello reduce significativamente el tráfico en la red.

Dentro del ambiente de Sybase, existen múltiples *Bases de Datos* y pueden ocupar varios discos; el Diccionario de Datos se almacena en cada *Base de Datos*.

Transact SQL o T-SQL es una versión mejorada del ANSI estándar; incluye definición, manipulación y control de datos. Las mejoras de SYBASE sobre esta son: instrucciones de flujo de control, uso de tipos de datos definidos y adicionales, utilerías y procedimientos almacenados. Es un lenguaje de programación que crea, mantiene y relaciona los objetos de una Base de Datos.

A nivel sistema operativo cada cliente tendrá una tarea separada para cada presentación (Front End) de su aplicación, habrá una sola tarea para el servidor, dependiendo de cuántos clientes tenga conectados.

Para la conexión en RED, el servidor será referenciado de acuerdo al nombre lógico que tenga asignado.

Data Workbench.

Es la sección de Sybase que corre y edita consultas (queries); procesos batch con consultas; uso de la lista de historia; salvar y cargar comandos de sistema operativo. Es la interfaz de 'ventanas' del servidor SQL, comprende: editor SQL, editor VQL, resultados, reporteador, entrada de datos, diccionario de datos, historial, ayuda, copiado de tablas.

Las tablas contienen columnas donde se tienen tipos de datos, reglas y defaults; pueden tener índices; la Base de Datos puede tener vistas, procedimientos y *triggers* basados en tablas. Sybase asocia reglas y defaults con columnas.

El Diccionario de Datos contiene información detallada sobre la *Base de Datos* y sus objetos.

La indexación *se usa para proveer eficiencia* y unicidad. Sybase soporta dos tipos de indexación:

Clustered Index: que ordena físicamente un sólo índice por tabla, los apuntadores al archivo están en orden. Opera *con árboles binarios.*

Nonclustered Index: se ordena la tabla y se agrega secuencialmente; no reordena los datos físicamente.

Con los términos clustered y *nonclustered se hace referencia* a una agrupación o no agrupación de índices de nuestras *Bases de Datos.*

Antes de continuar es necesario agregar tres conceptos que se manejan en *la teoría Bases de Datos Relacionales:*

- Llave Primaria: una o más columnas que identifican de manera única un renglón de una tabla.
- Llave Foránea: una o más columnas que son llave primaria en otra tabla.
- Llave común: una o más columnas que permiten *relacionar tablas.*

Las llaves operan a un nivel lógico y los índices a un nivel físico.

Un *batch* es un conjunto de instrucciones SQL que se usan juntas y se ejecutan como un grupo, una después de la otra. En ISQL se ejecutarán hasta recibir una palabra go. El servidor SQL revisa los batch y si encuentra un error, no ejecuta ninguna instrucción.

Procedimientos Almacenados.

Es una colección de *instrucciones SQL almacenadas en la Base de Datos* que pueden ejecutarse a través de un nombre; pueden aceptar y regresar parámetros y *valores, así como invocar a otros* procedimientos; corren más rápido que los comandos interactivos o que un batch.

Sus ventajas: reducen el tráfico de la red, almacenados en versión reducida; en su primera ejecución generan un plan de consultas (query); refuerzan consistencia de información; proporcionan nivel extra de seguridad; permiten el *desarrollo* de aplicaciones modulares; reduce el error; las transacciones complejas y delicadas pueden automatizarse; son recursivos; *permiten hacer referencia a otras Bases de Datos.*

Triggers

Es un tipo especial de procedimientos almacenados; activados automáticamente por el servidor cuando una tabla es actualizada, borrada o agregada; permiten mantener consistencia e integridad. Maneja integridad referencial, datos duplicados, derivación de columnas (o columnas derivadas); maneja restricciones más complejas que una regla no maneja. El concepto como tal no tiene traducción, así que generalmente se hace mención de él como *Trigger*.

Transacción

Serie de instrucciones T-SQL que son tratadas como un solo evento: El manejador de transacciones tiene tres componentes: instrucciones de control de transacciones para tratar instrucciones como unidad; el servidor almacena todas las modificaciones para garantizar recuperabilidad; el servidor proporciona páginas de bloqueo de tablas.

El bloqueo es manejado automáticamente por el servidor SQL; se tienen comandos compartidos y exclusivos.

Un deadlock ocurre cuando dos procesos tiene un candado exclusivo sobre una página que el otro proceso necesita; y ambos no pueden liberar su bloque hasta que tengan la información siguiente. El servidor lo detecta automáticamente y aborta la transacción. Para su minimización se usan: realizar la terminación de transacciones rápidamente; minimizar el uso de bloqueo completo y usar procedimientos almacenados para controlar la ventana de vulnerabilidad.

Recuperación de Fallas.

Generalmente se realiza una copia de la *Base de Datos* o de las transacciones, esto con el fin de asegurar la recuperación después de una falla.

A) Open Client.

Es la interfaz del servidor SQL con lenguajes de tercera generación 3GL; incluye una librería de BD (*Base de Datos*), un conjunto de rutinas y macros para que la aplicación interactue con el servidor; provee una interfaz consistente con programas en lenguaje C, no requiere un precompilador del host. Para la incorporación de las librerías del servidor y BD, un programa puede actuar como cliente y servidor.

B) Open Server.

Es una librería de programación con funciones en lenguaje C para ligar un programa y hacerlo un servidor; permite unificar una aplicación con sistemas de *Bases* de Datos múltiples o múltiples hosts. Provee un DBMS transparente, un mecanismo para accesar funciones fuera de la *Base de Datos*; es la interfaz estándar entre hosts y sistemas.

Una aplicación de Open Server es un manejador de eventos, en la que los eventos son asociados a un manejador para su ejecución.

Como hemos podido ver, Sybase ofrece muchas ventajas sobre VISUALFOX, dado que proporciona un lenguaje de consultas relacionales que facilitan la extracción de la información correspondiente sin necesidad de desarrollar nuevos programas para ello; pero también hay que mencionar que no todo se reduce a simplificar nuestras operaciones de acceso a la información, pues así como provee herramientas para mejorar seguridad, consistencia, independencia, integridad y demás; requiere un control mayor, desarrollo de programas mejor estructurados e implica un nuevo número de complicaciones en cuanto a la administración de la Base de Datos.

I.2.4. Arquitectura Cliente / Servidor

I.2.4.1. Conceptos Generales

Un sistema de Base de Datos en una plataforma Cliente/Servidor esta relacionada con la existencia de una red LAN (Local Area Network), en la cual cierto número de computadoras de escritorio (clientes) y algunas computadoras especiales (servidores) están conectadas. Las computadoras clientes preguntan por una gran variedad de servicios proporcionados por los servidores.

Al servidor que ejecuta programas para dar servicio a una Bases de Datos se le conoce como Servidor de Bases de Datos. El poder de la plataforma de Cliente/Servidor radica en el concepto de división de tareas. El cliente es una máquina de Front-End que interactúa directamente con los usuarios finales (cubre el aspecto de la presentación de los datos). El Servidor de Bases de Datos que es la máquina Back-End maneja dos tareas primordiales para los clientes (proporciona la capacidad de cómputo) acceso y actualización de la Bases de Datos.

En nuestro sistema el Front End lo representa el equipo PC que será nuestra interfaz con el usuario, y el Back End, serán los servidores, que en este caso es equipo Sun Solaris y un Sistema Administrador de Bases de Datos llamado SYBASE.

Conceptualmente, la arquitectura Cliente/Servidor son parte de la filosofía de Sistemas Abiertos (Open Systems) bajo la cual todo tipo de computadora, sistema operativo, protocolos de red y otro tipo de software y hardware puedan interactuar y trabajar en armonía para alcanzar el objetivo establecido por el usuario. El objetivo de los sistemas abiertos es alcanzar la interoperabilidad, es decir un estado de

comunicación múltiple y heterogénea que contribuyen a la conclusión de tareas comunes.

Para poder explicar esto, necesitamos realizar una clasificación o subdivisión de un sistema, de manera tal que permita aclarar fácilmente las diferencias entre las distintas arquitecturas, con cuyo nombre estamos designando a metodologías tanto físicas como lógicas que se emplean en el procesamiento de datos.

Una aplicación consta de seis partes fundamentales:

- Servicios de Presentación: es a través de éstos que un usuario introduce un conjunto de datos de entrada y por medio de éstos mismos, recibe una respuesta.
- Lógica de Presentación: contempla el conjunto de protocolos o elementos de comunicación lógica entre el usuario y la computadora.
- Lógica de la Empresa o de Aplicación: se refiere a todos aquellos elementos como políticas de la empresa, decisiones estructuradas, cálculos y operaciones que la aplicación debe tomar en cuenta.
- Lógica de Datos: abarca los comandos y operaciones que expresamos para que el Sistema Administrador de Bases de Datos o DBMS (Data Base Management System) nos proporcione sólo la información que necesitamos.
- Servicios de Datos: son las acciones que ejecuta o lleva a cabo el DBMS.
- Servicios de Archivo: generalmente engloba las funciones de manejo de archivos correspondientes a los diversos sistemas operativos.

Comparación con *otras Plataformas de* Bases de Datos.

Bases de Datos Centralizadas.

En una computadora central corren funciones fundamentales como la interfaz para el usuario, la lógica de las aplicaciones o el control de la integridad de los datos. Las terminales prácticamente no tienen poder de procesamiento y sólo se limitan a la comunicación con la computadora digital.

Sus ventajas radican en la gran capacidad y la operación en modo multiusuario. El esquema de seguridad es menos complejo al tener todo concentrado.

Pero por otro lado, puede llegar a ser muy caro, y en particular en cuestiones de mantenimiento

PC's de Redes de Area Local (LAN)

En esta plataforma se tienen *Bases de Datos* en Computadoras Personales (PC) corriendo en un sistema operativo de red. El acceso compartido de los datos es manejado por un servidor de archivos. Pero el procesamiento de los programas relacionados con las *Bases de Datos* corre en las PC's.

Los sistemas de Bases de Datos basados en esta plataforma incrementan la flexibilidad y velocidad de procesamiento.

Aunque todo depende de las características de las PC's; la seguridad y control de la integridad se complica y aumentan las tareas de administración.

La arquitectura Cliente/Servidor brinda una mayor flexibilidad de mantenimiento y velocidad del desempeño de aplicaciones. A pesar de la distribución de datos y aplicaciones en esta plataforma, se puede tener conocimiento de la ubicación da cada cosa, sin caer en un desorden. Se reduce el tiempo para desarrollo de aplicaciones debido a que se puede optar por emplear herramientas que sean familiares.

I.2.4.2. Roles del Servidor

Un servidor conjunta una serie de procesos agrupados de la siguiente forma:

Servicios de servidor.

- Acceso a archivos y localización de los recursos.
- En el caso de un Sistema Manejador de Bases de Datos, se encarga de la administración del sistema y la seguridad de los datos.

Lógica de la Aplicación

- Permite controlar la Integridad de los Datos y la Integridad Referencial
- Maneja transacciones
- Respaldos y recuperaciones

Conexiones de Red

- Controlar los dispositivos de comunicación.
- Dar transparencia a la conexión, a su mantenimiento y su terminación.

Dependiendo del elemento que se quiere explotar se pueden tener varios tipos de servidores, los más comunes son:

- Servidores de Archivos

- Servidores de Impresión
- Servidores de Bases de Datos
- Servidores de Comunicación

En particular el Servidor de *Bases de Datos* se enfoca a la administración de estructuras de *Bases de Datos* con una filosofía relacional. Las funciones encomendadas a estos servidores cubren los siguientes puntos:

- Integridad de Datos
- Integridad Referencial
- Manejo de Transacciones
- Seguridad de Datos
- Respaldo y recuperación.

La *integridad* de datos de una Bases de Datos se mantiene, mientras éstos sean correctos y completos durante su acceso.

La *integridad referencial* se refiere a la consistencia presentada por los datos y sus relaciones en las tablas que conforman la *Base de Datos*.

Las *transacciones* son la unidad lógica de trabajo en un servidor, es una operación o una secuencia de instrucciones SQL sobre la *Bases de Datos*; y puede cumplirse o fallar, este hecho es fundamental ya que nos da un cierto margen de seguridad, al saber de la factibilidad de realizar una operación pero sin afectar físicamente la *Bases de Datos* de forma inmediata.

La *seguridad de los datos* se basa en restricciones tanto en el acceso a los datos como a las operaciones que se realicen sobre éstos. Esta protección va a niveles de consulta, la alteración o destrucción de los datos.

El *respaldo* se llega a dar a varios niveles. El concepto de Espejo implica la duplicidad del contenido de una *Bases de Datos* en otro disco. Así en caso de que el disco de la Base primaria falle, se recurre al secundario. Se respaldan físicamente datos y/o el log de transacciones según la periodicidad que se establezca.

I.2.4.3. ¿Qué es el Cliente?

Dentro del Cliente se llevan cabo los siguientes procesos:

Servicios de Presentación

- Proporciona una interfaz al usuario final
- Maneja entrada y salida del usuario

Lógica de aplicaciones

- Mantener la integridad de datos y la integridad referencial.
- Controlar la aplicación.

Conexiones de red

- Establecer y terminar las ligas con la red.
- Crear, mandar y recibir mensajes.

Existen diversos tipos de aplicaciones en un cliente como lo son:

A) Interfaz con el usuario.

Permiten la interacción del usuario final con el sistema a través de la pantalla. El usuario da una instrucción vía una línea de comando o por medio de una interfaz gráfica. La información solicitada se puede desplegar desde la forma más informal hasta en un reporte detallado profesionalmente.

B) Herramientas de consulta y reporteo

Proporcionan los medios de consulta y reporteo de información en base a las especificaciones del usuario. En una consulta se pueden establecer que información se quiere extraer y que condiciones debe cumplir. Además se puede incluir algún cálculo adicional a los datos solicitados. El resultado obtenido de la consulta se puede guardar en diferentes formatos de almacenamiento o mostrar en pantalla o papel un reporte personalizado.

C) Herramientas de acceso Front End

Estas aplicaciones son sistemas comerciales enfocados a la manipulación de información. En la mayoría de los casos los usuarios ven cubiertas gran cantidad de sus necesidades pero tienen que ocupar tiempo para adaptarse a ellas. Aunque por otro lado el desarrollo de aplicaciones dedicadas disminuye si el usuario esta satisfecho con el alcance de estas herramientas. El uso generalizado de herramientas *Front End* como Excel, Dbase, Access, etc., nos encamina al terreno de los *Sistemas Abiertos*.

D) Análisis de datos

En esta área se tienen aplicaciones orientadas al soporte de decisiones que implican proyecciones o tendencias. También se consideran los sistemas basados en DataWareHousing, que es un concepto en el que se organiza y almacena periódicamente la información necesaria para el procesamiento analítico.

E) Herramientas de desarrollo

Para el desarrollo de aplicaciones con plataformas *cliente/servidor* se tienen herramientas como Power Builder y Visual Fox, que permite al diseñador crear *Front End* de una forma más estructurada y menos compleja gracias a su interfaz gráfica y sus librerías.

I.2.4.4. Comunicación Cliente/Servidor

El proceso de comunicación conjunta algunos componentes del cliente y el servidor. A nivel hardware incluye conectores de red, cable, repetidores y concentradores, que pueden estar bajo alguna de las siguientes topologías: Ethernet, Token Ring o Arcnet.

El software necesario debe contener un sistema operativo de red que proporcione los servicios, la red de comunicación que ligue los servicios y el proceso de mensajes que enlace la lógica de las aplicaciones.

Se origina con el uso de Mainframes (computador central con alta capacidad de procesamiento y almacenamiento), cuyas características principales son: procesamiento multiusuario centralizado; interacción a través de terminales; personal especializado para la administración del sistema. El procesamiento se realiza en su totalidad en un procesador central que es accesado por medio de terminales que generalmente *no* ofrecen facilidades gráficas de interacción y/o presentación de información.

El siguiente paso es la arquitectura de Servidor de Archivos, en el que los archivos son transferidos del computador central a las terminales, en donde se maneja la presentación, lógica de Empresa y funciones de la *Base de Datos*. Una de sus ventajas es que permite la creación de una interfaz gráfica, facilitando la interacción usuario máquina; pero una desventaja considerable es la carga de información en la Red.

El desarrollo de la arquitectura Cliente-Servidor se basó en la eliminación de estos problemas (carga de la red e implementación de la interfaz gráfica para facilitar el uso del equipo de cómputo); separando la aplicación en sus distintas partes y localizándolas en donde resultan ser más efectivas.

Los componentes principales son un *Cliente*, que invoca los servicios ofrecidos por un Servidor, que es el otro componente, el cual a su vez puede ser *Cliente* de algún otro Servidor.

Existen diversas variantes en cuanto a la estructura del *Cliente-Servidor*, basándose en las partes de la *aplicación residentes* en cada uno de los elementos *Cliente-Servidor*.

La primera opción es poner los servicios de datos y los servicios de archivos en el Servidor y los servicios de presentación, la lógica de presentación, la lógica de la empresa y la lógica de *datos en el Cliente*; este modelo se denomina Servidor de Datos Remoto. Proporciona la mejor escalabilidad, pero genera una gran carga en el poder de procesamiento del Cliente y en el ancho de banda de la red; genera además problemas de administración o manipulación, debido a que el *código* de las aplicaciones se encuentra en cada Cliente, lo que hace difícil algún tipo de modificación *en el código*.

En el modelo de Lógica Compartida, se almacenan secciones de la Lógica de Empresa en el Servidor, de manera que reducimos *una gran proporción* de la carga de información en la red.

Si toda la lógica de la empresa está en el Servidor, de manera que en el Cliente sólo tenga los Servicios de Presentación y la Lógica de Presentación, entonces nuestro modelo se conoce como *Arquitectura Cliente-Servidor* de Presentación Remota.

La Lógica de la Aplicación o de la Empresa es puesta en el Servidor empleando procedimientos almacenados, es decir, colecciones de lenguaje procedural que incluyen acceso a la *Base de Datos* (lenguajes de consulta, como SQL); este tipo de procedimientos mejoran el accesos a la *Base de Datos*, así como la integridad de la aplicación.

Los problemas principales del modelo de Presentación Remota son: gran carga en la *administración de un número* grande de Clientes; complicada implementación de la Lógica de la Empresa usando procedimientos almacenados; además de considerar que el uso de éstos últimos afecta la escalabilidad de nuestro sistema.

Los *problemas a nivel* general de los tres modelos de Cliente Servidor anteriores son: el procesamiento en lote y los usuarios interactivos interfieren unos con otros; la habilidad del DBMS de trabajar con *Commits* de dos fases (un método de ejecución de transacciones que asegura una marcha *atrás en caso de error* o falla en nuestras operaciones) se ve limitado en su eficiencia por la existencia de múltiples *DBMS. Estos tres modelos* se agrupan en un grupo denominado Arquitectura Cliente/Servidor de Dos Hilos.

En la Arquitectura de Tres Hilos de un Cliente Servidor, el Cliente está dedicado a la Lógica de Presentación y Servicios, y cuenta con una API (*Application Programming Interface*) Interfaz de Programación de Aplicaciones para invocar a la aplicación en un nivel intermedio. El *Servidor de Bases de Datos* está dedicado a Servicios de Datos y de Archivos. El nivel intermedio es una aplicación del Servidor donde se ejecuta la Lógica de Empresa y es ahí donde la Lógica de Datos invoca los Servicios de Datos; puede manejar las transacciones y asegurar la integridad de la *Base de Datos Distribuida*; puede manejar consultas asíncronas; centralizar la Lógica de aplicación para facilitar la administración y un cambio de manejador; proporciona el acceso a los recursos, basándose en nombres, en lugar de locaciones o lugares en la red.

En la FIGURA I.13. se muestran gráficamente las distintas arquitecturas durante la evolución del procesamiento de datos.

EVOLUCIÓN DE LAS ARQUITECTURAS DE PROCESAMIENTO DE DATOS

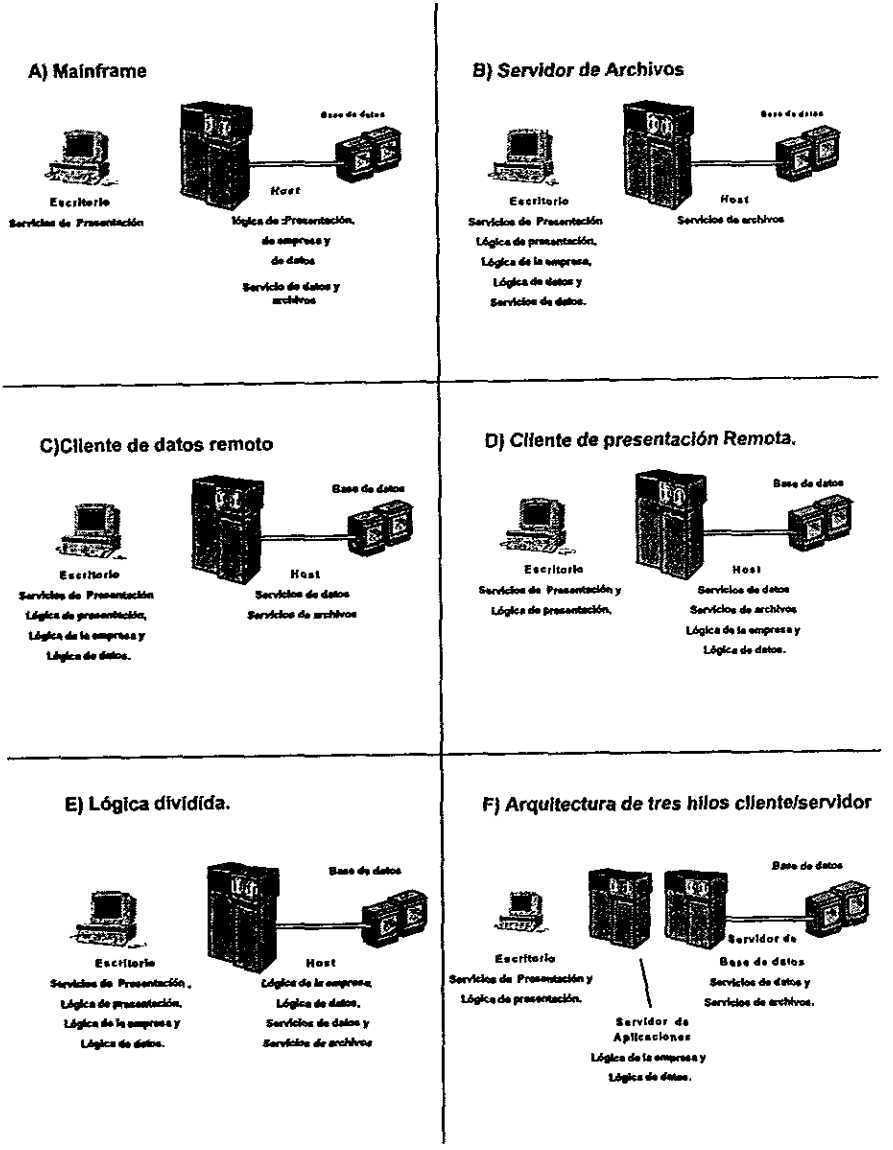


FIG. I.13. Arquitecturas de Procesamiento de Datos.

Las ventajas con las que cuenta la Arquitectura Cliente Servidor en general son:

- Desconcentración de aplicaciones.
- Aprovechamiento eficiente de Recursos.
- Avance hacia ambientes distribuidos.
- Flexibilidad debido a la independencia del Cliente y el Servidor.
- Consistencia.
- Acceso a nuevas tecnologías, lo que nos lleva a la apertura y estandarización, y a la desvinculación de los elementos de un sistema.

Las desventajas actuales son:

- Altos costos en Software y Hardware.
- Falta de personal capacitado en el área de administración del sistema.
- Herramientas para manejo y administración aún en desarrollo.
- Alto costo de mantenimiento.

1.2.5. Metodologías de Desarrollo de Sistemas.

Los problemas que se presentan en la construcción de sistemas de información no son simples versiones a gran escala de los que se presentan al crear pequeños programas de cómputo. Se puede hacer la analogía entre un puente para tránsito vehicular sobre un río y uno peatonal sobre un arroyo. Aunque pertenecen a la clase puente y tienen algunas características comunes, un ingeniero civil no diseñaría el primero de ellos por la simple ampliación del puente peatonal. La complejidad de los programas es tal que una persona puede comprenderlos con facilidad y retener en su mente todos los detalles del diseño y la construcción. Las especificaciones pueden ser informales y el efecto de las modificaciones evidenciarse de inmediato. Por otro lado, los grandes sistemas son tan complejos que resulta imposible para cualquier individuo recordar detalles de cada aspecto del proyecto. Se necesitan técnicas más formales de especificación y diseño; debe documentarse apropiadamente cada etapa del proyecto y es esencial una cuidadosa administración.

De manera formal la ingeniería de software permite establecer y usar principios de ingeniería para obtener un software económico y eficiente. Dentro de este campo de trabajo se manejan tres elementos: los métodos, las herramientas y procedimientos.

Los métodos proporcionan la técnica del como construir el software. Involucra una serie de tareas como: planeación, estimación, requerimientos, análisis, diseño de estructura de datos, algoritmos y arquitectura de programas, código, pruebas y mantenimiento.

Las herramientas dan un soporte automatizado o semiautomatizado a los métodos, un ejemplo es CASE (Computers Aided Software Engineering).

Los procedimientos ligan los métodos y las herramientas para poder tener un desarrollo del software racional y a tiempo. Los procedimientos definen la secuencia en la cual los métodos serán aplicados.

En el caso de este proyecto se plantea el uso de una metodología sin la ayuda de una herramienta automatizada de la cual no se dispone.

1.2.5.1. Ciclo de Vida Clásico.

Al igual que otros sistemas a gran escala, los grandes sistemas de software requieren un tiempo considerable para su desarrollo y permanecen en uso durante un tiempo aún mayor. En este período de desarrollo y uso pueden identificarse varias etapas, que juntas constituyen lo que se llama el ciclo de vida del software. El modelo inicial ha tenido diversos perfeccionamientos y variaciones. Todos ellos se pueden incluir en un modelo del ciclo de vida como el siguiente:

Análisis y definición de necesidades.

- Los servicios, restricciones y objetivos del sistema se establecen consultando a los usuarios. Una vez acordados, deben definirse de una manera comprensible, tanto para los usuarios como para el personal de desarrollo.

- Diseño del sistema o del software.

- El diseño de software es el proceso de representar las funciones de cada sistema a fin de poder transformarlo con facilidad en uno o más programas de cómputo.

- Codificación y pruebas de módulos.

- El diseño del software se materializa como un conjunto de programas o módulos escritos en algún lenguaje de programación, las pruebas de módulos implican la comprobación de que cumplen con las especificaciones.

Pruebas del Sistema.

- Los módulos integran y prueban como un sistema completo para asegurar que se cubren con las necesidades del software, después de las pruebas, el sistema de software se envía al usuario, en la fase de pruebas el que desarrolla el sistema debe convencer a los usuarios de que el sistema cumple con todas sus necesidades. Sin embargo, las actividades de verificación y confirmación deben ser cubiertas en las primeras etapas del ciclo de vida. Los grandes sistemas de software no son objetos estáticos, existen en un ambiente sujeto a cambios constantes y que los analistas de sistemas pueden no comprender mejor, los sistemas deberán adaptarse a esos cambios o ir perdiendo utilidad, hasta que se acaba desechado. Este proceso es lo señalado como evolución del software.

Operación y mantenimiento.

- Esta fase suele ser la más larga del ciclo de vida. Se instala el sistema y se pone en uso práctico. La actividad de mantenimiento implica corregir errores que no se descubrieron en las primeras etapas del ciclo de vida, mejorar la aplicación de los módulos del sistema y aumentar los servicios de éste a medida que se detectan nuevas necesidades.

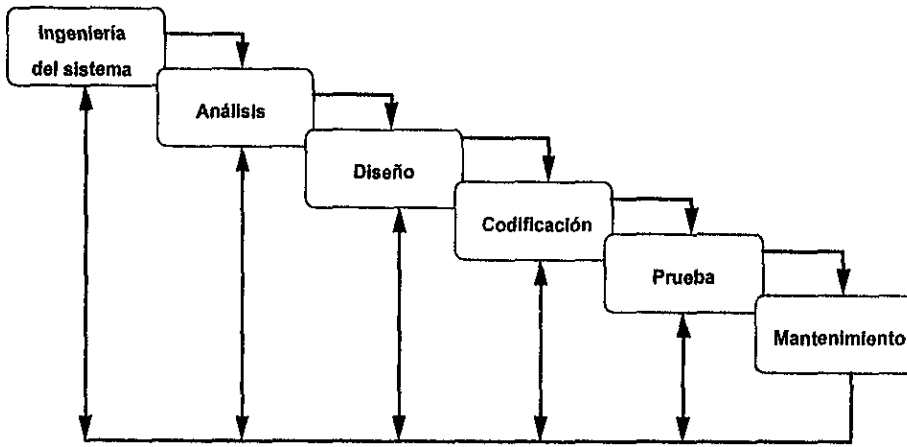


FIG. I.14. Etapas del Ciclo de Vida Clásico en el desarrollo de sistemas de información.

1.2.5.2. Construcción de Prototipos.

La mayor parte de los costos de los sistemas no resultan de errores en el sistema, sino en el cambio en las necesidades. Por tanto, para reducir los costos de mantenimiento y en consecuencia, los costos totales del ciclo de vida, debe establecerse una expresión más exacta de las necesidades reales del usuario. Esto ha conducido a las propuestas de que el modelo del ciclo de vida analizado antes, debe ser desechado y reemplazado por un modelo más evolutivo de desarrollo de software.

Este modelo de *Construcción de Prototipos* se base en la idea de que al usuario se le debe presentar, lo antes posible, un prototipo, debe construirse de manera que pueda ser modificado con facilidad. *Tan pronto* se construye el prototipo, se presenta el usuario, que experimenta en él y retroalimenta información a los analistas. Después el prototipo se modifica hasta que el usuario está satisfecho con el sistema.

El modelo prototipo puede tomar tres formas:

- Un prototipo en papel que retrate lo *más claramente* a la interacción usuario-máquina y permita al usuario entender como tal interacción ocurrirá.
- Un prototipo para implementar un subconjunto de funciones requeridas por el sistema.
- Un programa existente que realice parte de todas las funciones deseadas pero con otras características que puedan mejorarse en un nuevo esfuerzo de desarrollo.

La secuencia de eventos para la metodología del prototipo es la siguiente:

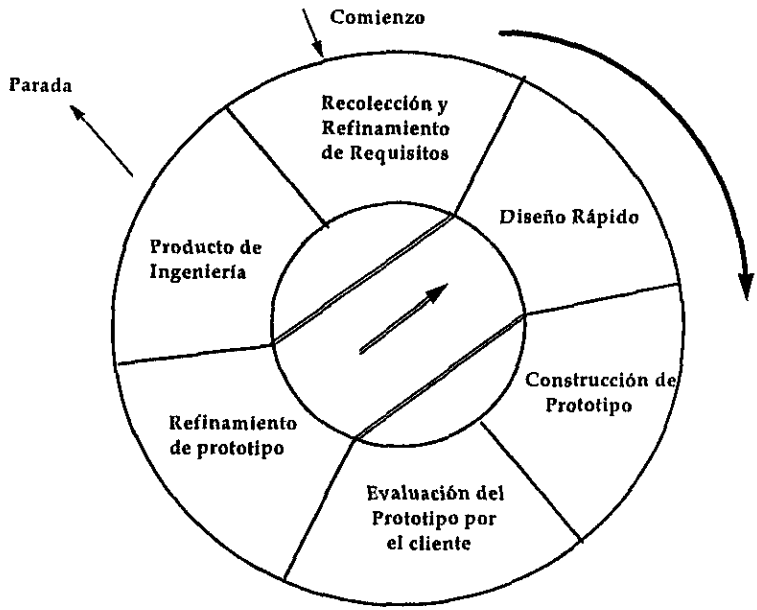


FIG. I.15. Construcción de Prototipos (en este modelo, las aplicaciones a desarrollar tienen una evolución cíclica).

1.2.5.3 Modelo en Espiral.

El modelo espiral fue desarrollado con base a las mejores características del ciclo de vida clásico y del modelo del prototipo, además de incluir un análisis de riesgo no contemplado en otras metodologías.

Este modelo agrupa cuatro principales actividades representadas en cuadrantes, como se observa en la figura I.16.

Identificación.

Se determinan los requerimientos iniciales y un proyecto cuyo plan se ira ajustando en base a los comentarios del cliente.

Diseño.

Se empieza a plantear a ciertos niveles las alternativas de solución.

Construcción.

Se implementan en un prototipo desde las ideas iniciales hasta los modelos particulares con todo detalle.

Evaluación.

Se realiza un análisis del riesgo de llevar a cabo el proyecto y se confronta con el cliente desde resultados parciales hasta las pruebas finales del sistema completo.

En cada iteración alrededor de la espiral (empezando del centro hacia afuera a trabajar), progresivamente versiones más completas del software son construidas. Revisando cada uno de los cuadrantes se observa un trabajo que va de *lo más general a lo particular.*

Cuadrante de identificación

Necesidades del Negocio

Se define el entorno del negocio y los objetivos relacionados con desarrollo del nuevo sistema. Se plantean los planes de prueba para establecer los criterios de aceptación del sistema.

Se establecen los alcances y objetivos de la primera construcción de ensayo de un subconjunto del sistema.

Por ultimo se revisan los requerimientos definidos para empezar a trabajar con la siguiente etapa.

Necesidades del Sistema.

Se identifican las áreas del negocio involucradas en el sistema, se establecen los resultados más importantes que deben ser obtenidos por el sistema cuando este se declare listo tanto por los usuarios como por los desarrolladores.

Una vez seleccionadas las tareas y datos que serán implementados en la primera construcción, se revisan los objetivos y el cumplimiento de los estándares que hayan sido fijados.

Requerimientos del los Subsistemas.

Los límites entre las funciones relacionadas de los sistemas son establecidos. Además los resultados básicos que *cada subsistema debe obtener son definidos.*

Se seleccionan los requerimientos que intervendrán en la segunda construcción y finalmente se revisan nuevamente las necesidades.

Requerimientos Finales

Se examinan y validan todos los requerimientos unitarios de los segmentos previos y permite el ajuste oportuno de los requerimientos del sistema y subsistema.

Cuadrante de diseño

- Diseño Conceptual: las tareas e información son bosquejadas como parte del nuevo sistema y se presenta el primer intento por representar (descomponer, transformar y modelar los procesos) a detalle la información, se piensa en un modelo conceptual de la interfaz con el usuario y se plantean mecanismos de seguridad, también se debe considerar en forma conceptual la factibilidad de los requerimientos a nivel de servicios, elasticidad, periodos aceptables de desempeño, etc.

- Diseño Lógico: el diseño conceptual es llevado a entidades lógicas con atributos y tomando en cuenta mecanismos para garantizar la integridad de la información, los procesos son implementados en transacciones que pueda manejar el sistema; la interfaz con el usuario es diseñada en su presentación de transacciones delineadas y se modifican los modelos de datos y procesos para verificar su factibilidad de los requerimientos junto con las consideraciones pertinentes ante ajustes posteriores.

- Diseño Físico: se diseña la distribución y los tipos de datos de la información; se detalla el sistema en subsistemas y sus procesos, para la interfaz con el usuario se establecen las pantallas y reportes junto con detalles derivados de las observaciones del cliente.

- Diseño Final: se completan todos los modelos y representaciones del sistema en detalle suficiente para soportar la construcción final.

Cuadrante de construcción

Prueba del Concepto: se prueba que también el diseño conceptual puede manejar los requerimientos importantes del negocio.

Primera Construcción: evalúa los requerimientos y diseño lógico del sistema propuesto, elaborando el diseño, codificación y prueba de un software basado en requerimientos específicos del negocio, sistema y tecnología.

Segunda Construcción: se construye y prueba el código para el sistema.

Construcción Final: se termina la construcción de todas las funciones del sistema y se prueba su adecuado desempeño.

Cuadrante de evaluación

Análisis de Riesgo: se revisa y evalúa sobretodo las ventajas del desarrollo del sistema planeado en este ciclo; se obtiene la información para revisarse en las futuras construcciones junto con la recomendación del como proceder.

Primera Evaluación: se evalúa el progreso del desarrollo del sistema planteado en el aspecto de requerimientos del software, diseño lógico y primera construcción; con la experiencia obtenida de las evaluaciones de los requerimientos, diseños y software del segundo ciclo se realizan las recomendaciones para el siguiente.

Segunda Evaluación: examina las actividades del tercer ciclo para evaluar el progreso y hacer las recomendaciones pertinentes.

Prueba Final: usuarios y clientes prueban el sistema para su aceptación total, previa planeación de las pruebas y procedimientos para conducirlas; libera los planes de puesta en marcha del software junto con su documentación.

La metodología de espiral es el modelo más aproximado a la realidad del desarrollo de sistemas de amplia escala. Permite al desarrollador y al cliente entender y reaccionar al riesgo en cada nivel de evolución. Se usa el modelo del prototipo como forma de disminuir el riesgo pero algo más importante, es el hecho de que el analista de sistemas pueda aplicar el prototipo en cualquier etapa de evolución del producto.

Se mantiene una secuencia sistemática apegada al ciclo de vida clásico pero incorporando un trabajo iterativo que refleja mejor el mundo real. Una especial consideración se debe hacer sobre el riesgo a diferentes niveles del proyecto, si es bien apreciado, reduce la probabilidad de una situación crítica.

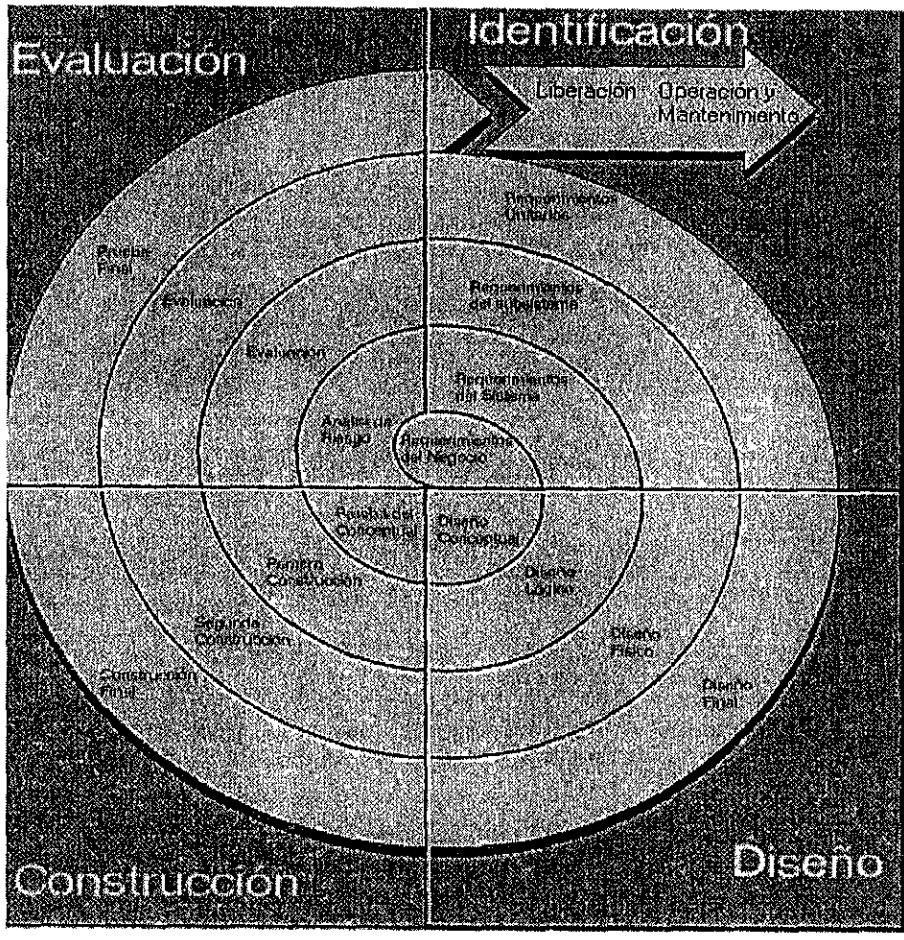


FIG. I.16. Enfoque Evolutivo presentado en el modelo en espiral, resultado de la combinación adecuada de las metodologías del ciclo de vida clásico y construcción de prototipos.

1.2.5.4 Técnicas de Cuarta Generación.

El término Técnicas de Cuarta Generación (T4G) involucra a un amplio conjunto de herramientas de software que automáticamente generan código fuente partiendo de las especificaciones del sistema. La importancia de T4G para la

ingeniería del software, se enfoca en la habilidad de especificar software a una máquina a una nivel muy cercano al lenguaje natural.

Algunas herramientas que apoyan al modelo T4G son: lenguajes no procedurales para consulta de Base de Datos, generador de reportes, manejo de datos, definición de interacción de pantallas y generación de código, capacidades gráficas de alto nivel y capacidades de hoja electrónica.

Aunque estas herramientas existen su aplicación ha sido muy específica. No existe una herramienta de T4G que pueda ser utilizada en forma amplia en el desarrollo de todas las categorías de sistemas de información.

Muchas empresas que trabajan con T4G limitan a aplicaciones de sistemas de información de negocios o administrativos, con buenos resultados para aplicaciones pequeñas e intermedias, pero para desarrollos de software que por su tamaño demanden mayores esfuerzos, el tiempo de análisis, diseño y prueba es negativo en comparación al tiempo ahorrado al eliminar código.

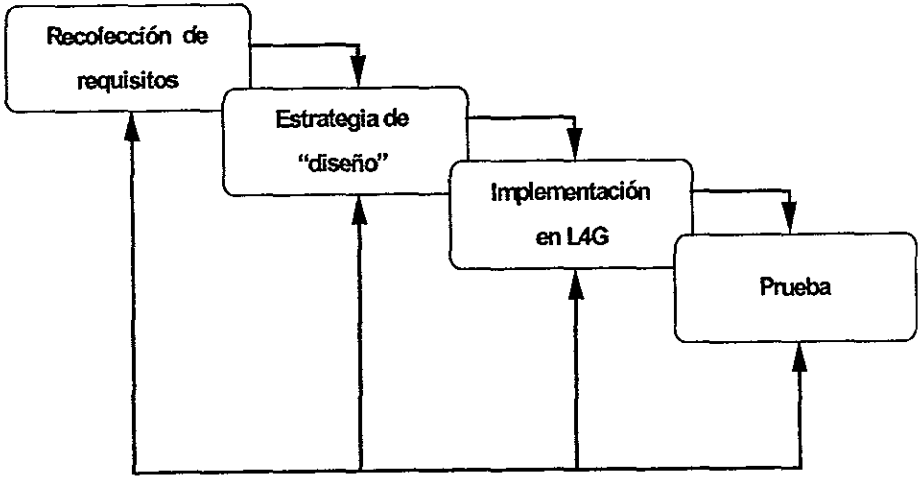


FIG. I.17. En las técnicas de cuarta generación el número de pasos a seguir se reduce, porque el trabajo a realizar se simplifica de manera considerable, pero no debemos olvidar que el diseño no debe perder ningún detalle.

Capítulo II

Creación de la Base de Datos Relacional

II.1. Requerimientos del Sistema

Se requiere tener de manera integrada la información de las Aplicaciones que se utilizan en la institución y a las cuales pueden accesar los usuarios para realizar alguna consulta o extracción de la información. Ya que los mismos usuarios plantean gran número de requerimientos de información unilateralmente que en algunos puntos son coincidentes; pero, por diversos problemas principalmente de comunicación, no se ha logrado una difusión de la información disponible en la institución.

La información se encuentra de la siguiente manera: existen Temas generales de los cuales para su estudio se crean diversos Proyectos que tienen una tarea específica para lo cual se auxilia de una serie de Aplicaciones que se realizan en la institución, todas ellas tienen un acceso independiente, por lo que los usuarios que no conocen el proyecto no tienen conocimiento de las partes que lo forman. Para lo cual se creara una estructura de Bases de Datos Relacionales en la que se tenga reflejada la estructura de la información, y su relación con los usuarios.

Los Temas tienen un nombre (clave) con el que se reconocen y una descripción más completa para una mayor comprensión de la información que se manejará; con la clave además se hace referencia en los Proyectos.

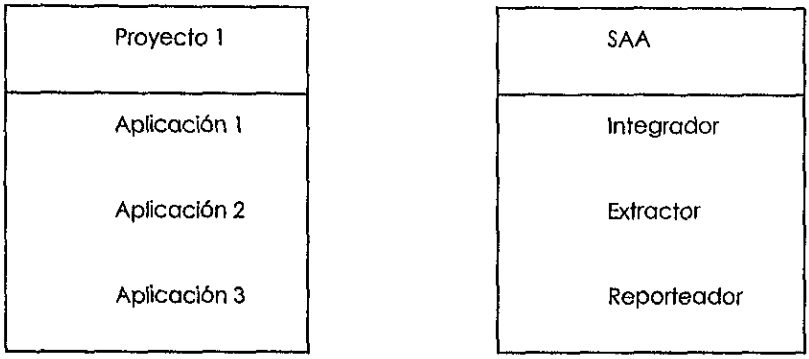
Los Proyectos se identifican con un nombre (clave); una descripción; un tema asociado, de manera que resulta más clara la comprensión del origen de la información; y una oficina responsable de la información, la cual tiene mayor

conocimiento de la información debido a que fue quien la solicitó; con la clave se hace referencia al Proyecto en las Aplicaciones.

Las Aplicaciones se identifican con un nombre (clave), una descripción, un tipo de Aplicación (integradores, reporteadores y extractores), una ruta (esta puede ser interna o externa, esto tiene relación con el tipo de usuario), y en caso especial de que se trate de algún integrador mantener el control de la exclusividad, para no permita que alguien más interfiera en el proceso; para mayor control e información se tiene el dato de la oficina responsable de la información.

Las oficinas se identifican con una clave y cada oficina tiene un responsable, de manera, que en algún momento sabemos con quien nos podemos dirigir para alguna aclaración de las Aplicaciones

Dentro de un Proyecto existen varias Aplicaciones, a las cuales se trata de encasillar a todos con una misma estructura, para darle al programador más facilidad de comprensión al programar de manera mas estructurada y clasificada, como se muestra en el diagrama siguiente.



Para el acceso a las Aplicaciones los usuarios se identifican con una clave y un password, además de tener un registro en el que se tienen algunos datos como el nombre, el número de conexiones que puede realizar simultáneamente; de igual manera para el nuevo Sistema se requerirá controlar lo anterior además de los días y horas de acceso permitidas, e identificar de que tipo de usuario se trata, dependiendo de la clasificación siguiente:

- usuarios internos: los cuales verán todos estos proyectos

- usuarios externos: los cuales solo verán los proyectos que les concierne

Un tipo especial de usuarios internos son aquellos que no tienen límite en el número de conexiones que puede realizar, *normalmente estos* privilegios se le asignan a los desarrolladores por cuestiones de pruebas múltiples.

Una tarea muy importante, será controlar los accesos de los usuarios y sus privilegios, mediante un protocolo de privilegios, con el cual se sabrá si un usuario puede accesar información especifica. Los privilegios que se otorgaran a los usuarios podrán ser de: lectura escritura y/o modificación, y serán otorgados por el responsable de la información.

Para control del Sistema *se deberá registrar* los accesos de cada usuario al sistema y no permitirle el acceso cuando a agotado ya sus conexiones, también *registrar a que aplicaciones* ha entrado, que día y que operación realizó; de esta manera se le permitirá a los usuarios:

- Realizar accesos controlados al Sistema.
- Controlar las operaciones que pueden realizar dentro de los mismos.
- Mostrar un esquema de la estructura de la *información con la que* se cuenta.
- Que la información este disponible para todos los usuarios, evitando que se tengan distintas fuentes.
- *Darle mayor agilidad al acceso* de las aplicaciones salvaguardardado la información a la cuál podrán tener acceso por este medio.

Con estos requerimientos se realizará el esquema de las Bases de Datos para el "Sistema Administrador de Aplicaciones", que permita concentrar *en un solo punto* los datos de modo que los sistemas ya existentes o nuevos, tomen dicha información y se ahorren la tarea de mantener al día la *información en forma* independiente.

II.2. Análisis de Datos

En el análisis de los datos se van dando solución a los requerimientos especificados anteriormente, para llegar así al diseño del modelo entidad-relación-atributo que es la representación más clara de las reglas del negocio.

Nos auxiliaremos de técnicas propias de "diseño lógico" o "diseño conceptual" o "modelado de datos".

II.2.1. Entidad, Llaves y Relaciones.

Algunos conceptos útiles en el transcurso de esta tesis serán los siguientes:

Entidades.

Entidad: persona, lugar, objeto, concepto, actividad o evento de interés para la empresa por ejemplo: *Empleado, Departamento.*

Tipo de entidad: es el nombre de la entidad, usualmente se escribe en mayúsculas, pero no es la única manera, eso depende de los estándares que se definan, en este caso se pondrá en mayúsculas sólo la primera letra de cada palabra y si la entidad estuviera formada de dos palabras cada una tendría la primera en mayúsculas, por ejemplo: *AtributoUsuario*.

Instancia de la entidad: es una *ocurrencia*, usualmente se escribe en minúsculas.

Cuando nos refiramos a la "Entidad" significa "tipos de entidad": nombre de la entidad (regularmente conviene diseñar *las entidades en tablas*).

Todas las entidades son sustantivos, pero no todos los sustantivos son entidades, ignorar los sustantivos que:

Denotan un dato específico.

No tiene relación con tu empresa

A las entidades se les crean una llaves primarias Pk (de sus siglas en inglés Primari key) con la que se facilitara realizar *un mejor diseño* de los datos, para determinarlas, se sugiere:

Sea un identificador único para cada instancia (atributo o conjunto de atributos).

Características necesarias: único, no cambie en el tiempo.

Una características deseables es que sea Estable y controlado por el administrador de datos

Un caso especial de las Entidades son Entidades Dependientes y sus características son:

No existen sin una entidad padre (dependencia).

Siempre tiene relación con el padre (*normalmente muchos-uno*).

La llave primaria de la entidad dependiente incluye la llave del padre y columna de definición

Subentidad:

Es un subconjunto de otra entidad, llamada superentidad. Se crean subentidades importantes de una entidad, que tengan algo especial.

Herencia:

Se ceden las llaves primarias, atributos y relaciones aplicadas a superentidades para todas las subentidades.

Para crear una superentidad alrededor de entidades similares se tendrán que tomar en cuente los siguientes puntos:

Las entidades similares tienen atributos en común, hay que asignar atributos en común para superentidades, y atributos especiales para subentidades, y considerar llaves artificiales primarias para una nueva superentidad

Clase:

Grupo de subentidades mutuamente exclusivas-alineadas verticalmente; a cada clase le corresponde clasificar un atributo de la superentidad. Una superentidad puede tener varias clases no relacionadas.

En nuestro caso en especial no encontramos subentidades, pero se mencionan por análisis.

A continuación ilustraremos nuestras entidades requeridas:

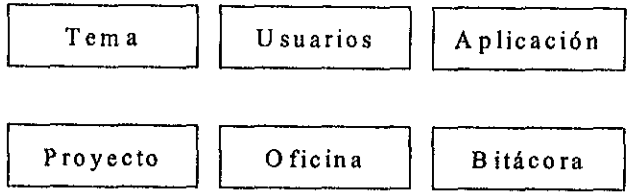


Diagrama de Entidades

Relación

Relación: acuerdo entre dos entidades

Usuario - Responsable - Oficina

Usuario - *AsignadoA* - *Aplicación*

Tipo de relación: enunciados acerca del tipo de la entidad, usualmente escrito en mayúsculas, o en un estándar, aquí sólo se *manejarán mayúsculas* al inicio de palabra

Usuario -Responsable - Oficina

Instancia de la relación: enunciado *acerca de* la instancia de la entidad, escrito en minúsculas

Yosafat Cisneros - Maneja - Operaciones

Usualmente cuando se *refiere a* '*Relación*' significa 'Tipo de relación', y está llega a ser una llave foránea Kf (de las siglas en ingles Key foreign) en diseño relacional.

Todas las relaciones son verbos pero no todos los verbos son relaciones, ignorar los verbos siguientes:

- que describen procesos *más que datos*
- *que no tiene relación* con tu empresa

A continuación se ilustran las relaciones que encontramos en nuestras Entidades:

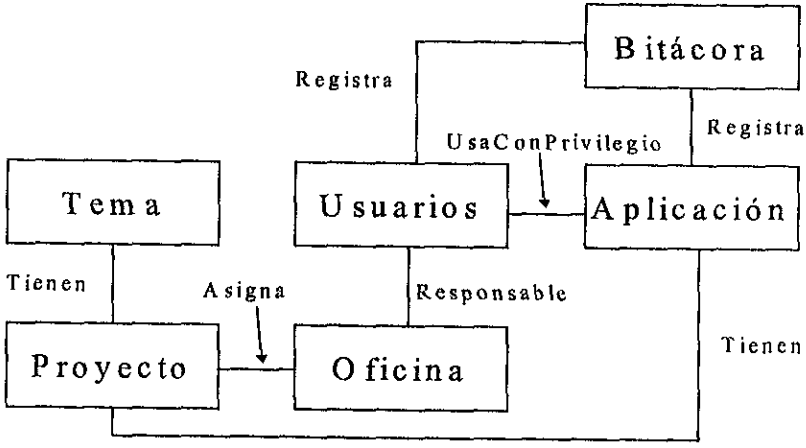


Diagrama de Entidad Relación

Y se leen de la siguiente manera:

A las Oficinas se les Asignan Proyectos
A los Proyectos se les Asignan Oficinas

Los Usuarios se Registran en la Bitácora
En la Bitácora se Registran los Usuarios

Las Oficinas tienen un Responsable Usuario
Los Usuarios son Responsables de las Oficinas

Las Aplicaciones se Registran en la Bitácora
En la Bitácora se Registran las Aplicaciones

Los Usuarios Usan con Privilegios las Aplicaciones
Las Aplicaciones se Usan con Privilegios por los Usuarios

Los Temas Tienen Proyectos

Los Proyectos Tienen Tema

Los Proyectos Tienen Aplicaciones

Las Aplicaciones Tienen Proyecto

Como podemos ver en la relación *Usan_con_Privilegios* es una relación especial que contiene más atributos que la simple relación, por lo tanto ésta la convertiremos en otra entidad que llamaremos Privilegios.

Cardinalidad

La cardinalidad es el número de instancias u ocurrencias en una relación. Por ejemplo la cardinalidad de:

Usuarios - Usan con Privilegios - Aplicaciones

¿ Cuántos Usuarios pueden Usar con Privilegios una Aplicación ? muchos

¿ Cuántas Aplicaciones pueden Usar con Privilegios un Usuario ? muchos

*Cuando se trata de una Relación muchos a muchos indica que lo más conveniente es convertir la Relación en una nueva entidad, y se dice que se rompe la cardinalidad M-M.

La cardinalidad de

Proyecto - Tiene - Tema

¿ Cuantos Proyectos Tiene un Tema? muchos

¿ Cuantos Temas Tiene un Proyecto? uno

*esto es llamado la máxima cardinalidad y es la que utilizaremos, pero también hay otra llamada la mínima cardinalidad.

Cardinalidad Mínima: es la mínima que puede existir.

La mínima cardinalidad de:

Usuario -Responsable - Oficina

¿ cuántos Usuarios deben ser Responsables de una Oficina ? uno

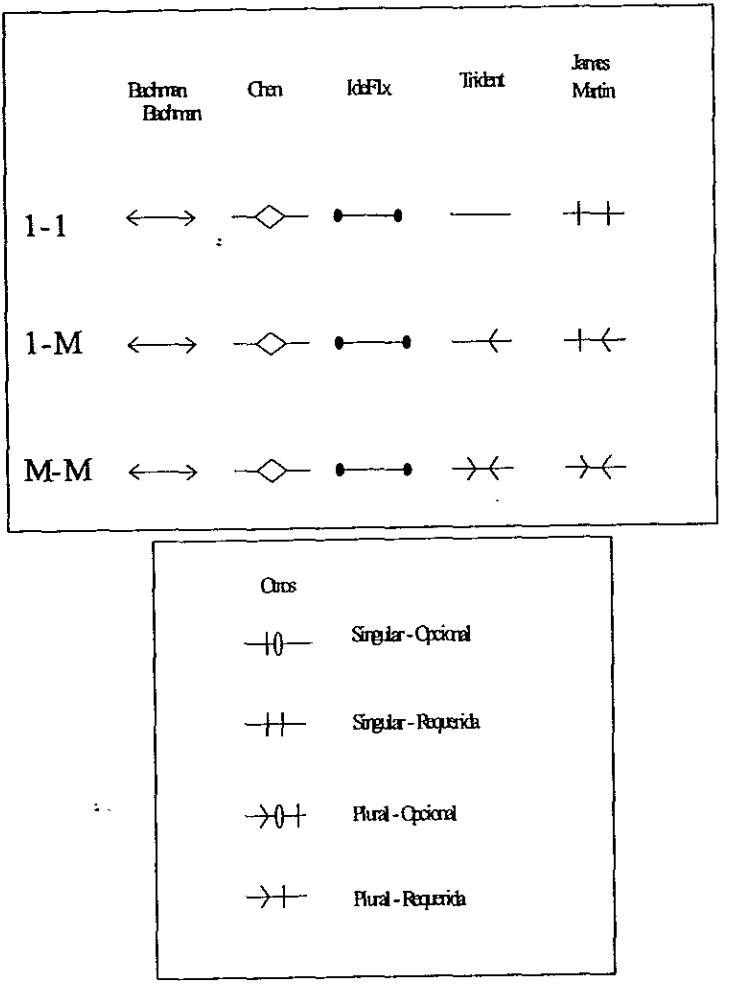
¿ De cuantas Oficinas deben ser Responsables cada usuario ? cero

* la cardinalidad en una regla especial de integridad y refleja las políticas del negocio

Terminología:

- Singular = máximo uno (un solo valor 1-1)
- Plural = máximo muchos (varios valores a una ocurrencia 1-M)
- Opcional = mínimo cero
- Requerido = mínimo uno (llave primaria)

Dependiendo de los autores encontraremos diferentes notaciones para la cardinalidad.



Para decidir un nombre formal de las Relaciones tomar en cuenta los siguientes puntos:

- Usar voz activa cuando sea posible.
- Lista los sinónimos en el diccionario y elimina los homónimos.

El nombre de la relación incluye el nombre de la entidad (puede ser borrado, de ser obvio).

Usuario - Usa -Privilegios

*Al escribir la descripción usar sentencias completas y ejemplos.

Las relaciones en N-sentido involucran a más de dos entidades. Observa los enunciados con frases preposicionales conteniendo una entidad.

A los usuarios se les da de alta en varios proyectos con ciertos atributos específicos.

Relación recursiva: es un enunciado acerca de una entidad.

Usuario - Jefe - Usuario

Documente los roles de las entidades en las relaciones recursivas (ejemplo supervisor, subordinado).

Una relación indirecta es redundante a otras dos relaciones. Usualmente se excluye las relaciones indirectas del diccionario.

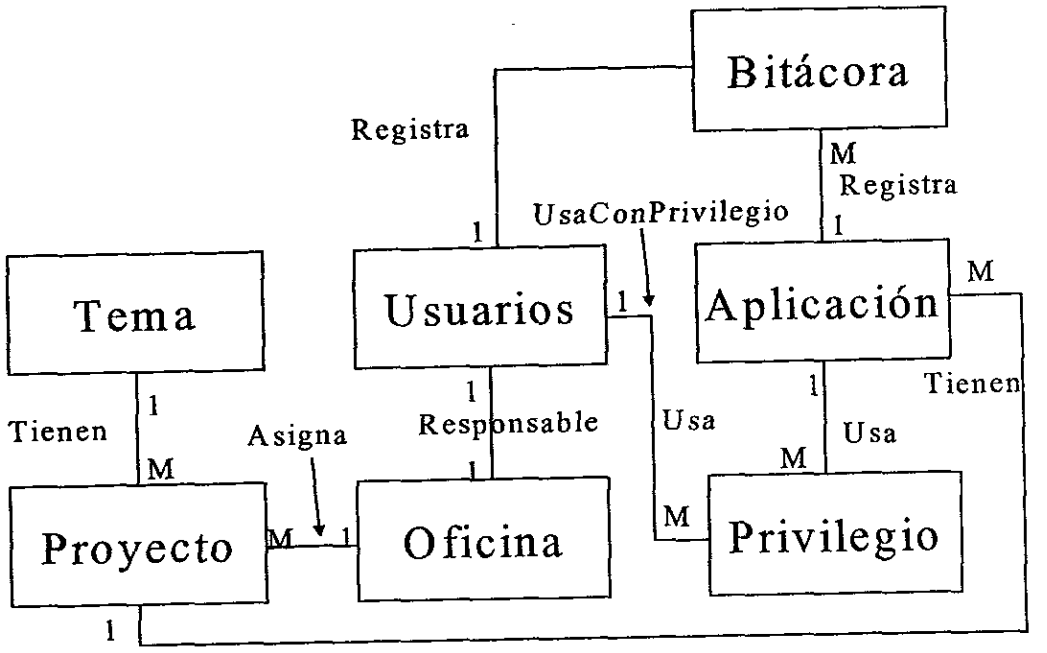


Diagrama de Entidades-Relación con Cardinalidad

Atributos

Atributos:

Describe una propiedad de una entidad o relación.

Tipo de atributo:

Nombre del atributo, usualmente escrito en mayúsculas o con otro estándar, aquí se utilizará con sólo mayúscula al inicio de cada palabra.

Instancia de un atributo: valor individual, escrito en minúsculas, "atributo" significa "tipo de atributo".

No confundir ¿Relación o Atributo?

Cuando un atributo asocia la "llave" primaria de alguna entidad es realmente una relación.

Las aplicaciones tienen rutas de acceso, tipos y códigos de responsables de oficina.

No confundir ¿Relación o Entidad?

Los verbos pueden tener forma de sustantivos y viceversa.

Si la llave primaria consiste de llaves primarias de otras entidades, es una relación.

No confundir relaciones con entidades o atributos; los atributos designan entidades y relaciones. Si la llave primaria de una entidad consiste de otras llaves primarias, puede ser una relación.

Usualmente los atributos se transforman en columnas en el diseño relacional.

Un atributo es un sustantivo que denota un dato especifico (*nombre, fecha, cantidad, valor_monetario*).

Las llaves son atributos que cumplen con las características necesarias, tanto de llaves primarias como de llaves foráneas.

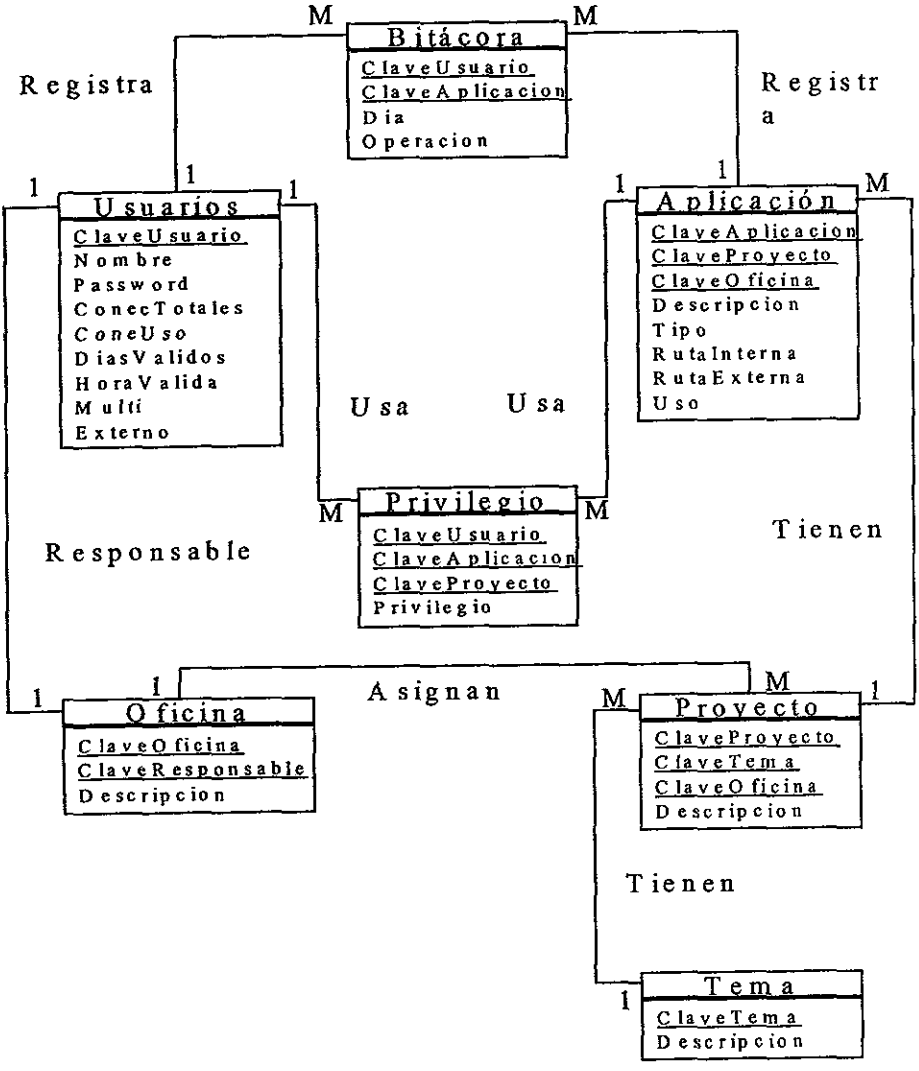


Diagrama de Entidad Relación Atributo

En el anterior diagrama se muestran el diagrama completo de entidad-relación-atributo con sus llaves subrayadas.

Para el total diseño de nuestra Base de Datos resta el ver las características de los tipos de datos:

nombre (nombre, conexión, monto).

tipo físico (carácter, entero, flotante, binario).

número de caracteres o dígitos.

valor permitido (ejemplo: color in [read, green, blue).

reglas de validación (ejemplo: Tipo IN LIST[I,R,E]).

valor default (ejemplo: Exteno = 0).

tipos de datos compuestos: varios tipos relacionados en un paréntesis ejemplos:

DATETIME = (Año, Mes, Día, Hora, Minuto, Segundo)

TELEPHONE=(Código_País, Código_Área, Número, Extensión)

FULLNAME=(Nombre, Apellido_Paterno, Apellido_Materno)

Algunas sugerencias para decidir un nombre formal de un atributo son las siguientes:

Un nombre formal de un atributo puede ser:

ClaveUsuario, DíasVálidos, ClaveResponsable

El tipo de dato puede tomarse de un estándar definido por las reglas de tu negocio, por ejemplo:

- Que cada palabra inicie en mayúsculas.

- Eliminar homónimos y documentar los sinónimos.
- Que el nombre sea lo más descriptivo posible.

Es recomendable que en la documentación que baya realizando se tomen en cuenta los siguientes punto:

- Escribir una descripción de cada uno de los Atributos.
- Usar sentencias completas y ejemplos.
- Especificar unidades para valores: distancia, tiempo y peso.

Observar el siguiente Posesivo:

El horario de los usuarios.

Observar el Verbo Propietario:

Los usuarios tienen claves de acceso.

Observar las siguientes Preposición:

El horario de los usuarios.

La Ruta de ejecución para la aplicación.

Los usuarios tienen acceso a las aplicaciones con unos atributos particulares.

Nosotros también asignamos claves de acceso (a los usuarios esta implícito)

Como podemos observar en las entrevistas con los usuarios se escuchan los Atributos, identificando a las Entidades y Relaciones. Si buscas un posesivo, un verbo propietario o una preposición y entonces sabrás cuáles son los Atributos.

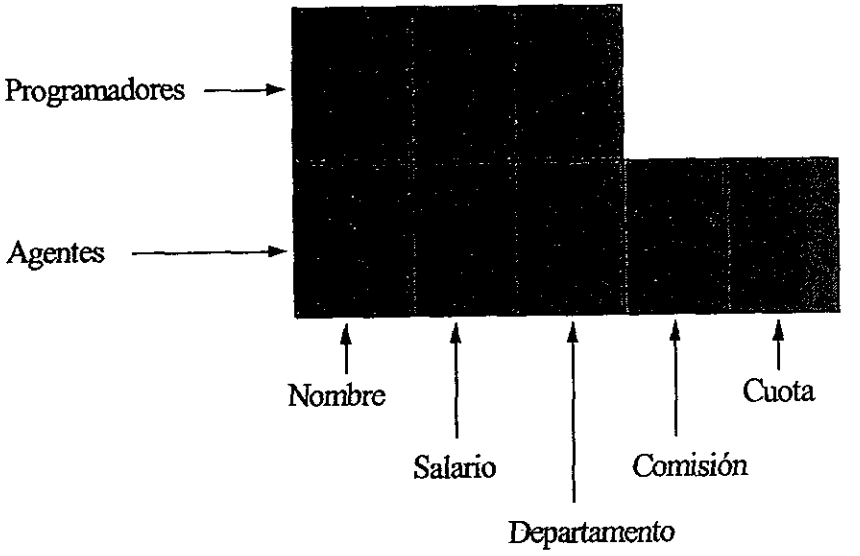
Cuando se trata de Subentidades, el atributo indirecto no es realmente una propiedad de su padre. Un caso especial de relación indirecta son los *Atributos Derivados*, éstos son calculados desde otros atributos por lo regular éstos se utilizan con cantidades

Compensación = SalarioHora * 2000+ Bono

El Atributo Primitivo se ingresa directamente y para los Atributos Derivados es recomendable se documente la formula si el atributo es importante o la derivación es oscura. Aunque de preferencia considere eliminar el atributo derivado, ya que en cualquier momento puede generarse y en cambio si alguno de los Atributos Primitivos cambia tienes que tener ciertas reglas de integridad definidas en tu tabla o de otra manera se encontrará incongruencia.

Los Atributos Opcionales son aquellos que pueden venir o no, cuando esto sucede se encontrarán muchos huecos en la tabla en ese momento será recomendable que se utilicen Subentidades.

Los Atributos Opcionales tienen la mínima cardinalidad cero. Los Atributos son opcionales porque su valor es desconocido o inaplicable. Generalmente se convierten, cuando opcional significa inaplicable.



Problemas de datos L-Shaped

En este caso que muestra el diagrama anterior encontramos huecos que dejan los casos de inaplicables, por lo que resulta conveniente descomponerla en subentidades.

Si existen datos históricos (que reflejan además del estatus actual, el pasado o el futuro) en las Entidades, Relaciones y Atributos, aplican entonces:

Para Entidad o Relación:

- Añada la fecha o el atributo fecha/hora para un punto o un estado continuo
- Añada (FechaInicio, FechaFinal) para estados segmentados.

Para Atributos:

Para los atributos con forma compuesta ponga el nombre de los campos que lo componen o una descripción con la que sea fácilmente identificable.

CantidadSalario, FechaEfectivo

Documente los estados pasados/futuros en descripción.

Añada DATE, DATETIME, o los atributos *FechaInicio, Fecha_final.*

Nuestro Análisis de Datos nos ha llevado al siguiente diagrama de los datos final:

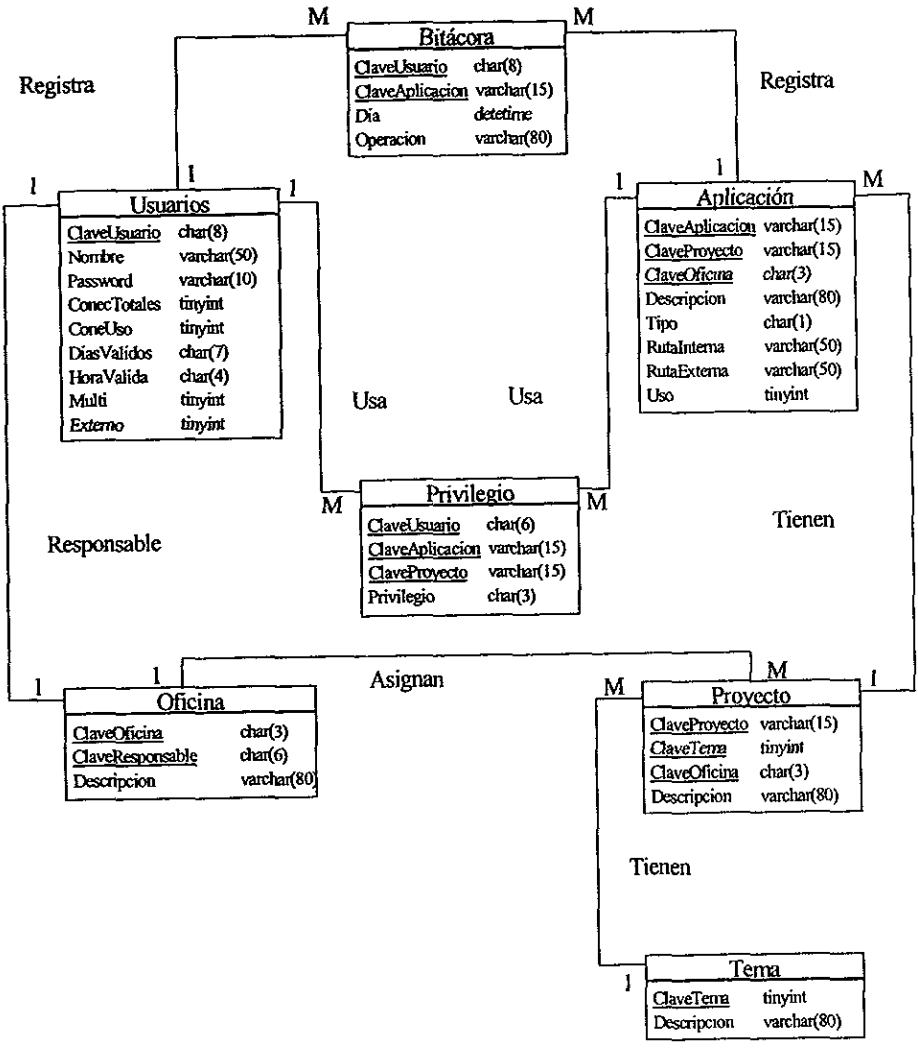


Diagrama Final del Análisis de Datos

II.3. Diseño de la Base de Datos Relacional

Aquí se esquematiza y detalla el diseño, se hace la representación o modelo de los datos, siguiendo los siguientes pasos:

Diseño relacional: diagrama entidad relación (normalizar): se definen tablas, columnas y llaves (eliminar redundancia inútil).

Reducir redundancia para simplificar la administración y mejorar integridad.

Desempeño satisfactorio.

Recurrir al "diseño lógico".

Diseño físico: depende del ambiente y la implementación.

Especificar la organización interna de tablas en el disco (almacenar estructuras, índices, particiones, asignar tabla a dispositivos).

Metas: optimizar el rendimiento.

¿Qué es una Base de Datos Relacional?

Una definición simple dice:

"Datos tabulados con restricciones, proyecto y operaciones de unión(Join)"

Una definición práctica:

"El sistema soporta SQL"

Pero la mejor definición dice:

"Sistema que opera conforme a los principios de Modelo Relacional"

El Modelo Relacional

La estructura de datos esta formada por los siguiente elementos: Tabla, Archivo, Columna, Tipos de Datos, Valor Nulo, Vista.

Los Operadores que se pueden utilizar son: Restricción, Proyecto, Asociación, *Unión, Diferencia, Intersección, Producto,* División.

Las Reglas de Integración se realizan con: Llaves Primarias y Ajenas, Integridad Referencial y Entidad.

(Nota: corresponde a las disciplinas de Ingeniería de Software)

Terminología:

FORMAL	COMÚN	PHYSICAL
Relación	tabla	archivo
Tupla	renglón	registro
Atributo	columna	campo
Dominio	tipo de dato	tipo de dato

Ejemplo de SQL:

```
create table Oficina(ClaveOficina        char(3),
                     ClaveResponsable          char(8),
                     Descripcion         varchar(80))
```

TABLAS DE EJEMPLO

ClaveOficina	ClaveResponsable	Descripción
A13	YCR	Oficina de Operaciones
B17	CBO	Oficina de Requerimientos

Definición Formal de una Tabla.

La tabla consiste de un encabezado y un cuerpo. El encabezado esta compuesto de un juego de columnas C1...Cn y datos de tipo D1...Dn. El cuerpo esta cambiando un juego de REGISTROS {(C1:V1)...(Cn:Vn)} - todos los valores Vi están en datos tipo Di

Otras características:

Los registros y las columnas no tienen un orden (relacional) inherente.

Los registros y las columnas duplicadas están desautorizadas.

Exactamente un valor por celda de archivo/columna (esto se logra si las tablas son normalizadas)

II.3.1. Tipos de Datos.

Componentes de los Tipos de Datos:

Representación física (Integer, Char, Varchar, Decimal)

Nombres definidos por el usuario (*DíasHábiles*)

Juegos de valores definidos por el usuario y validación de reglas (*DíasHábiles=[1,2,3,4,5,6,7]*)

Tipos de datos complejos y compuestos (*Polígono, Arreglo, FechaHora*)

Funciones definidas por el usuario (Área(Polígono) Nombre(Apellidos Paterno))

Chequeo automático para los datos compatibles en expresiones.

Valores Nulo.

Símbolo especial, independiente del tipo de dato, significa *desconocido* o *inaplicable* y requiere de Nuevas tablas de verdad, que se muestran enseguida.

AND	TRUE	FALSE	NULL
TRUE	true	false	null
FALSE	false	false	False
NULL	null	false	null

OR	TRUE	FALSE	NULL
TRUE	true	true	null
FALSE	true	false	null
NULL	true	null	null

X	NOT(X)
TRUE	false
FALSE	true
NULL	null

**Por Definición:* el resultado de cualquiera de estas operaciones: < = > + - * / es nulo cuando cualquier argumento es nulo.

Valores por Defaul.

Son valores que se asignan a los tipos de datos cuando se sabe por anticipado cual será su valor, los valores normales de los tipos de datos representan cada uno valores estandarizados. Existen algunos problemas con los Defaults, por ejemplo ¿qué pasa si todos los valores son significativos? hay que tomar en cuenta que:

Existirían posibles errores con las funciones de SQL.

No puede eliminar nulos desde SQL.

Cuando más se recomienda el uso de defaults es para datos tipo carácter.

Uso de nulos para datos de tipo numérico cuando las funciones de SQL son importantes.

Operadores.

El modelo relacional tiene 8 operadores: Restricción, Proyecto, Join, Unión, Diferencia, Intersección, Producto, División. Los operadores son las bases de la álgebra relacional implementada en SQL. Ejemplos de la restricción:

```
SELECT *
FROM Proyecto
WHERE ClaveOficina = 'A15'
```

ClaveProyecto	ClaveTema	ClaveOficina	Descripción
SAA	Archivos	A15	Sistema Administrador de Archivos
MEL	Laborales	A15	Manejo de Elementos Laborales
FIC	Capitalización	A15	Fondo de Inversiones de Capitalización

Las Reglas de Integridad se inician con las llaves y de éstas podemos decir lo siguiente:

Llave: una columna o grupo de columnas usadas para identificar un registro.

Llave simple: una llave formada de una columna simple

Llave compuesta: una llave formada con múltiples columnas

¡Una llave no es lo mismo que un índice!

Las llaves *identifican* registros (diseño relacional)

Los índices establecen registros (diseño físico)

Los tipos de llaves en el Modelo Relacional son: Primarias, Foráneas, Candidatas y Alternativas

Llaves Primarias: solo garantiza el modo para identificar positivamente los registros

UPDATE Atributo

SET Privilegio = 'R'

WHERE ClaveUsuario = 'A15DGR'

Llaves Primarias: deben ser únicas (2 registros no pueden tener el misma valor de Llave Primaria al mismo tiempo), Mínimas (todas las columnas son necesariamente únicas), Nunca Nulas (ninguna columna puede ser Nula). Las extracciones pueden fallar cuando la llave es parcial o totalmente Nula.

Llaves Foráneas: columna(s) que se refieren a una llave primaria de alguna tabla. La Llave Primaria en una tabla y Foránea otra tabla en la que se hace referencia deben tener un mismo nombre (cuando sea posible), para hacer más clara la relación. *También es posible que la Llave Primaria puede contener una llave Foránea*

ClaveAplicación	ClaveProyecto	ClaveOficina
INTSAA	SAA	A15
REPMEL	MEL	B45

Las Reglas de Integridad nos dicen que:

- Las llaves primarias deben ser únicas.
- Integridad de *la entidad: todas las columnas* de una llave primaria deben ser *No-Nulas*

- Integridad referencial: una llave foránea debe ser asociada a algún valor del correspondiente en la Llave Primaria.
- Las reglas de las llaves foráneas especifican como mantener la integridad referencial.

Tablas de vistas:

- *Tabla Básica:* físicamente existe como almacén de registros en la Base de Datos.
- *Tabla de vistas:* los registros son derivados de una tabla básica y es almacenada en un catálogo.

```
CREATE VIEW Externos(ClaveUsuario, Nombre)
AS      SELECT (ClaveUsuario, Nombre)
        FROM Usuario
        WHERE Externo = 1
```

Ventajas del Modelo Relacional:

- Fundamento teórico (agrupa teoría, lógica y álgebra)
- Independencia de datos físicos
- Coloca niveles de extracción
- Estructura de datos uniforme.

Veremos algunos ejemplos de la nomenclatura de SQL

Tipos de datos: integer, smallint, tynyint, float, char(n), varchar(n), binary(n), varbinary(n), money datetime, text, image y definidos por el usuario.

Como se usan los tipos de datos:

```
/* Crear un tipo de dato TipoAplicación de tipo varchar(3) que acepte nulos
*/
```

```
SP_ADDTYPE TipoAplicacón, 'VARCHAR(3)',NULL

/* Utilizar el tipo creado anteriormente para la definición de una tabla */

CREATE TABLE Aplicacion(Tipo TipoAplicacion, ... )
```

Como crear Defauls

```
CREATE DEFAULT TipoAplicacion_Defaul AS 'E'
```

Como crear Reglas

```
CREATE RULE TipoAplicacion_Rule AS TipoAplicacion IN ('R','W','U')
```

Como ligar Reglas y Defaults a una columna o un tipo de datos

```
SP_BINDRULE TipoAplicacion_rule, TipoAplicacion_type
SP_BINDEFAULT TipoAplicacion_Default, Aplicacion.Tipo
```

Como documentar una Llave en una tabla

```
SP_PRIMARYKEY    Usuario,ClaveUsuario
SP_FOREINGKEY    Aplicacion,ClaveOficina
```

Los procedimientos del sistema documentan las llaves dentro de las tablas del sistema

Las llaves comunes de Sybase documentan como asociación de columnas para SQL. Ejemplo:

Llaves primarias

Especificar no nulidad para todas las columnas

Refuerzo único con índices

CREATE UNIQUE INDEX ClaveUsuario ON Usuario(ClaveUsuario)

Refuerza las reglas de Llaves Foráneas con triggers, éstos son procesos almacenados que se disparan cuando sucede una acción especifica (existen de tres tipos: insert, delete y update).

En el Diseño relacional básico, cada entidad se transforma en una tabla. Las entidades independientes se transforman en tablas independientes y las entidades dependientes se transforman en tablas dependientes. Las subentidades se transforman en subtablas

Algunas reglas para las Llaves Foráneas son:

- Los nulos y los defaults no están permitidos.
- Eliminar cascadas o restringirlas.
- Clasificar columnas en supertablas es recomendado.

Diseño alternativo

Las relaciones se transforman en Llaves Foráneas.

Nombre de Llave Foránea = nombre de Llave Primaria (añadir calificativo si es necesario).

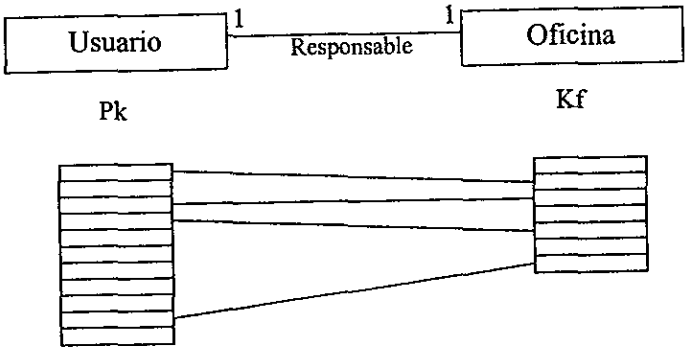
Uso de nombres significativos en vistas (manager, spouse).

En una relación *Muchas a Uno:*

El lugar de la Llave Foránea es del lado de muchas

En una relación *Uno a Uno:*

El lugar de la Llave Foránea va dentro de la tabla con menos registros.



*Reduce o elimina los valores nulos.

En una relación Muchos-muchos se transforma en una nueva tabla de asociación, en donde se encuentran las Llaves Primarias Pk de las tablas que se asocian. En esta nueva tabla pueden encontrarse columnas adicionales. Los

atributos se convierten en columnas; los atributos singulares se convierten en una columna y los datos de tipo compuesto se transforman en muchas columnas (hasta el soporte del vendedor).

Reglas de las Llaves Foráneas

- Los nulos y los defaults no están permitidos
- Eliminar cascadas o restriccciones

En los atributos de tablas dependientes sucede lo siguiente:

- La Llave Foránea se transforma en una tabla padre.
- La Llave Primaria contiene una Llave Foránea y un atributo plural.

Especificar no nulo para los atributos requeridos y las relaciones.

El análisis requerido = mínima cardinalidad 1 = columna no nula (diseño).

Especificar no nulo para:

- La columna a la cual representa requiere atributos.
- Fk a la cual representa requiere relaciones.
- Todas las columnas Pk *siempre*.
- Se desautorizan los defaults cuando signifiquen desconocido o inaplicable.

Los índices pueden insertar lentamente actualizaciones

- *Crear siempre* un índice único en todas las columnas Pk.
- Los atributos de las relaciones van con las Llaves Foráneas.

II.3.2. Normalizar

Redundancia.

- La repetición de un *valor* es necesaria en una Base de Datos Relacional.
- La redundancia es la repetición de un *hecho*
- La redundancia puede mejorar la ejecución, pero es generalmente indeseable.

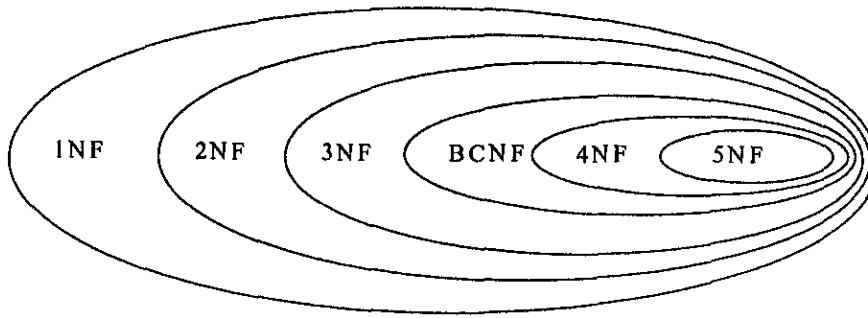
Problemas de Redundancia

- Datos inconsistentes
- Anomalías de actualización
- Ejecución de intercambio (reparación vs. Actualización).
- Incertidumbre en los procedimientos de mantenimiento de datos

El uso de triggers para manejar las anormalidades de actualización e inconsistencia, pero la ejecución intercambiada y la complejidad persisten.

Forma Normal

Las Formas Normales son estructuras de tablas que reducen la redundancia



Las formas normales más altas se logran por descomposición sin perdida - dividiendo una tabla en pequeñas tablas, sin perdida de información.

Primera Forma Normal:

Todos los valores de la tabla dependen de la Llave Primaria Pk. Todas las tablas relacionales están en 1fn por definición (no cumple una tabla que tienen todos los datos iguales).

Las tablas (1fn) pueden ser redundantes, cuando un atributo depende del valor de otro atributo.

Segunda Forma Normal:

La tabla esta en 1FN y cada columna sin llave depende en la Llave Primaria *completa.*

Dado una tabla T en primera forma normal, consistente de una Llave Primaria Pk en múltiples columnas p1, p2, ... y un atributo A, *se dice que T esta en su* segunda forma normal si y solo si el valor del atributo a en cualquier renglón depende de los valores de p1, p2,... para ese renglón. Las tablas con Llaves Primarias en una sola columna están siempre en 2a forma normal

Una causa de redundancia es que un atributo depende únicamente sobre una parte de la llave primaria (una columna depende indirectamente en la Llave Primaria).

Elimina la dependencia parcial *por medio de* la descomposición sin perdida de una tabla.

Tercera Forma Normal:

La tabla está en 2FN y cada columna sin llave depende *directamente* de la Llave Primaria.

Dada una tabla T en segunda forma normal, consiste de una Llave Primaria P y dos atributos A1 y A2, se dice que T esta en tercera forma normal si y solo si el valor del atributo A1 en cualquier renglón no depende del valor en el atributo A2

Cada columna sin llave depende en la llave, la llave completa y nada excepto la llave

Redundancia en las tablas 3NF

Una causa de redundancia es que un atributo depende de otra columna que no sea la Llave.

Elimina redundancia por descomposición sin perdida de información.

Forma Normal Boyce-Codd

Llave candidata: cualquier columna (o grupo de columnas) que es *única* y *mínima*.

Defectos en la definición de 3FN

¿Qué pasa con las Llaves Candidatas?

¿Qué pasa con los problemas *dentro* de una Llave Primaria compuesta?

Definición de BCNF

- La columna es una determinante cuando alguna otra columna depende de ella
- La forma normal boyce-codd: cada determinante es una Llave Candidata.
- BCNF remueve toda la redundancia debido a sus relaciones singulares.

Cuarta Forma Normal:

La tabla esta en 3nf y no contiene relaciones independiente muchas-muchas.

Quinta Forma Normal

¿Que pasa con la redundancia desde las relaciones dependientes muchas-muchas?

Relación dependiente = relaciones de n-sentidos

Generalmente las relaciones en n-sentidos no pueden ser descompuestas sin perdida de información

Dependencia Cíclica

Si el empleado ejecuta el rol r y el empleado e es asignado al proyecto p y el proyecto p requiere un rol r entonces e *debe* estar asignada al proyecto p en el rol r.

La tabla con dependencia cíclica puede ser descompuesta sin perdida.

La dependencia cíclica causa anomalías de actualización. Así la tabla probablemente *podrá* ser descompuesta.

La definición puede ser generalizada a relaciones con n-sentidos.

Quinta forma normal:

La tabla esta en 4FN y no tiene dependencia cíclica (excepto entre llaves candidatas).

La 5FN es la última forma normal sin redundancia adicional puede ser eliminada por proyección. La 5FN es primordialmente de interés académico.

II.4. Construcción

Construcción de software y código de BD (diagrama de Entidad Relación) con SQL

- Pruebas e instalación del sistema
- *Mantenimiento* (preventivo y correctivo) y mejoras

Diseño relacional de reestructuras: adicionar, cambiar índices, llaves Pk Fk

Reorganización física

Especificaciones del Diseño Físico:

- Índices
- Partición: como esta la información físicamente
- Granularidad de los bloques
- Asignación de unidades de almacenamiento

La meta del diseño físico es para optimizar el rendimiento

- Disco I/O
- Almacenamiento
- Tiempo de CPU

En relación al DBMS, el diseño físico nunca afecta el resultado de las consultas.

Categorías de Consultas (Queries)

- Bath vs Interactivo: son instrucciones en paquete.
- Update (insert, delete) vs retriee: muy actualizada o muy consultada
- Ordenar resultado vs 'random'
- Tabla_simple vs join
- Registro-simple vs registro_multiple

II.4.1. Construcción del Modelo.

Se crean las tablas que se diseñaron con base a todo el análisis anterior y que son las siguientes:

```
CREATE TABLE Aplicacion(
        ClaveAplicacion     varchar(15)     NOT NULL,
        Descripcion         varchar(80)     NOT NULL,
        Tipo                char(1)         NOT NULL,
        RutaInterna         varchar(50)     NOT NULL,
        RutaExterna         varchar(50)     NOT NULL,
        Uso                 tinyint         NOT NULL,
        ClaveOficina        char(3)         NOT NULL)

CREATE TABLE Bitacora(
        ClaveUsuario        Varchar(8)      NOT NULL,
        Día                 datetime        NOT NULL,
        ClaveAplicacion     varchar(15)     NOT NULL,
        Operacion           char(1)         NOT NULL)

CREATE TABLE Oficina (
        ClaveOficina        char(3)         NOT NULL,
        Descripcion         varchar(100)    NOT NULL,
        ClaveResponsable    char(6)         NOT NULL)

CREATE TABLE Proyecto(
        ClaveProyecto       varchar(15)     NOT NULL,
        Descripcion         varchar(100)    NOT NULL,
        ClaveTema           tinyint(1)      NOT NULL,
        ClaveOficina        char(3)         NOT NULL)

CREATE TABLE RAplicacion(
        ClaveAplicacion     varchar(15)     NOT NULL,
        Descripcion         varchar(80)     NOT NULL,
        Tipo                char(1)         NOT NULL,
        RutaInterna         varchar(50)     NOT NULL,
        RutaExterna         varchar(50)     NOT NULL,
        Uso                 tinyint(1)      NOT NULL,
        ClaveOficina        char(3)         NOT NULL)

CREATE TABLE RProyecto(
        ClaveProyecto       Varchar(15)     NOT NULL,
        Descripcion         Varchar(100)    NOT NULL,
        ClaveTema           Tinyint(1)      NOT NULL,
        ClaveOficina        char(3)         NOT NULL)

CREATE TABLE RUso(
        ClaveUsuario        Varchar(8)      NOT NULL,
```

```
    ClaveAplicacion     Varchar(15)     NOT NULL,
    ClaveProyecto       Varchar(15)     NOT NULL,
    Privilegio          Varchar(6)      NOT NULL)

CREATE TABLE Tema(
    ClaveTema           tinyint(1)      NOT NULL,
    Descripcion         varchar(100)    NOT NULL)

CREATE TABLE Privilegio(
    ClaveUsuario        varchar(8)      NOT NULL,
    ClaveAplicacion     varchar(15)     NOT NULL,
    ClaveProyecto       varchar(15)     NOT NULL,
    Privilegio          varchar(6)      NOT NULL)

CREATE TABLE Usuario
    ClaveUsuario        varchar(8)      NOT NULL,
    Nombre              varchar(50)     NOT NULL,
    Password            varchar(10)     NOT NULL,
    ConecTotales        tinyint(1)      NOT NULL,
    ConecUso            tinyint(1)      NOT NULL,
    DiasValidos         char(7)         NOT NULL,
    HoraValida          char(4)         NOT NULL,
    Multi               tinyint(1)      NOT NULL,
    Externo             tinyint(1)      NOT NULL)
```

II.4.2. Construcción de índices.

Los índices son secuencias lógicas de valores en columnas numéricas ordenados por el diccionario.

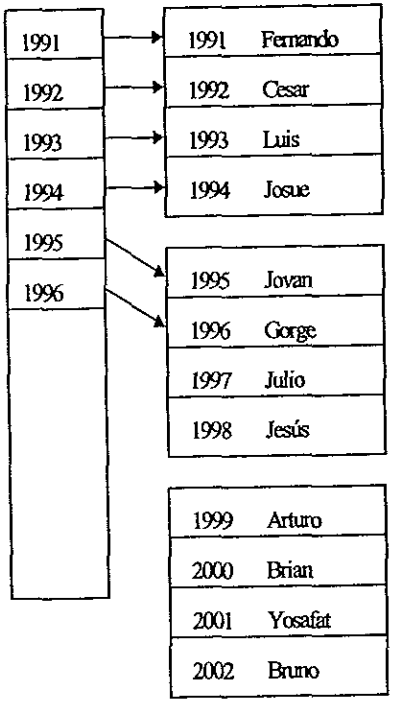
Las secuencias físicas de los renglones en una tabla, se componen de la secuencia lógica de la paginación del disco y secuencia física de registro dentro de la página

Columna Clustered: secuencia lógica de valores que embonan físicamente, en otras palabras, son índices cuyo orden coincide físicamente con el orden de la tabla.

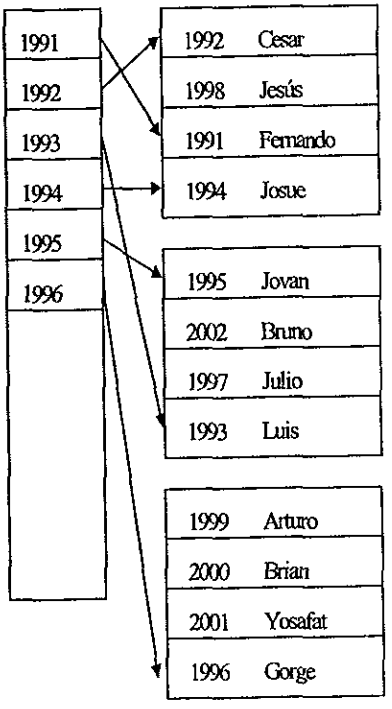
Índice compuesto: es un índice definido en varias columnas

Índice Clustered: en un índice en una columna ordenada (clustered), realiza reordenamiento.

Índices Nonclustered: un índice que no esta en una columna ordenada (clustere), crea apuntadores.



Indices Clustered



Indices Nonclustered

Crea Indices Clustered usualmente en la Llave Primaria Pk, ocasionalmente alguna otra columna es más importante para accesar la tabla.

Los Indices Clustered son únicos y los Indices Nonclustered puede haber de 1 a 4 Indices por tabla incluyendo el Indice clustered

Sintaxis:

CREATE [CLUSTERED | NONCLUSTERED] INDEX nombre_indice

ON nombre_tabla (nombre_columna,...)

[WITH{FILLFACTOR = X, IGNORE_DUP_KEY,[IGNORE_DUP_ROW | ALLOW_DUP_ROW}]]

DROP INDEX nombre_indice

FILLFACTOR: asigna en espacio libre.

El tamaño máximo de un índice es que la suma de la(s) columna(s) sea 255 bytes

Según nuestro análisis podemos observar que las tablas que serían lo suficientemente grandes y/o lo suficientemente solicitadas, serían la tabla de Usuarios que contendrían los registros de cada uno de los empleados y la más pesada sería la de los Atributos ya que por cada UsuarioAplicacion existiría un registro. Y los campos de búsqueda por los que se entraría en la tabla, serían precisamente las Llaves por lo que resulta conveniente usar Indices tipo Clustered.

CREATE CLUSTERED INDEX IndAtributo

ON Atributo (ClaveUsuario, ClaveAplicacion)

CREATE CLUSTERED INDEX IndUsuario

ON Atributo (ClaveUsuario)

Capítulo III

Sistema Administrador de la Base de Datos

Una vez establecidas las características de la Base de Datos propuesta se deben considerar los procesos o tareas implicadas en la administración de la misma. En este capítulo se enfoca al desarrollo de un Sistema Administrador de la Base de Datos que permita al usuario administrador un acceso más eficiente de la información contenida en la Base de Datos.

Con las herramientas de la Aquitectura Cliente/Servidor y siguiendo una Metodología en Espiral, se irá implementando los procesos necesarios tanto de Cliente como de Servidor en cada una de las secciones siguientes, tomando en cuenta las funciones específicas de cada parte.

III.1 Primer Ciclo

III.1.1Requerimientos Iniciales del Negocio

La información contenida en las tablas de la Base de Datos establecerá un control sobre el acceso a diversos sistemas de información de la institución. La supervisión de las Aplicaciones requiere de ajustar o consultar frecuentemente los Datos de control almacenados en las tablas.

Hasta este punto, la Base de Datos construida sólo es manipulada por los individuos encargados del desarrollo del sistema, quienes tienen el conocimiento de herramientas de interacción con la Base de Datos Sybase con un lenguaje SQL. Además de lo laborioso que resulta realizar algunas transacciones (como insertar 20 registros en alguna de las tablas), que aunque no son procesos muy complejo, si quitaran un tiempo muy preciado para un desarrollador .

De este modo la automatización de los procesos más frecuentes se vuelve una necesidad, junto con el desarrollo de un sistema que sirva de interfaz para los usuarios encargados los cuales deberán contar con el conocimiento básico de Windows en este caso en su versión 95.

III.1.2 Diseño Conceptual

Considerando el esquema de Cliente/Servidor propuesto, los procesos involucrados en la administración de la Bases de Datos se distribuyen entre el Cliente y el Servidor. Por un lado el Servidor se encarga de albergar la Base de Datos y dependiendo del proceso, ejecutar ciertas funciones de forma más eficiente que desde el Cliente directamente.

El Cliente manda una solicitud de información al Servidor, el cual ejecuta el proceso correspondiente directamente sobre la Base de Datos. El resultado traído por el Servidor llega al Cliente el cual le da una presentación que le permite al usuario identificar claramente lo que buscaba.

La comunicación entre el Cliente y el Servidor requiere del ODBC (Open DataBase Connectivity) que es un administrador de los drivers necesarios para establecer la conectividad entre dos manejadores de Bases de Datos diferentes, a nivel Windows. Para alguna Aplicación en particular se debe establecer un Origen de Datos (Data Source) basado en el driver correspondiente. El Origen de Datos contiene los Datos que el usuario quiere accesar y la información para obtenerla, por ejemplo:

Para una Base de Datos SQL Server, se debe especificar el Servidor donde reside la clase de Datos, así como la red usada para accesar tal Servidor.

Por el lado del Servidor, Sybase cuenta con los elementos llamados Stored Procedures. Estos procedimientos almacenados conjuntan una serie de instrucciones

que nos permitirán interactuar directamente con las tablas de la Base de Datos. De este modo desde nuestro Cliente VisualFox podemos mandar llamar los Stored Procedures y tan solo enfocarnos a la presentación de los Datos.

Como se ha observado Visual FoxPro proporciona las herramientas necesarias para crear potentes aplicaciones Cliente-Servidor, por combinar la potencia, la velocidad, la interfaz gráfica de usuario y las sofisticadas funciones de consultas, informes y procesos con el acceso multiusuario, y la sintaxis nativa del Servidor del Origen de Datos o sevidor ODBC.

Utilizando, la tecnología de paso a través de SQL que permite enviar instrucciones de SQL directamente a un Servidor. Puesto que las instrucciones se ejecutan en el Servidor de apoyo, son un método potente para mejorar el rendimiento de las aplicaciones Cliente-Servidor. Por ejemplo, puede efectuar definición de Datos en el Servidor remoto, establecer propiedades del Servidor y acceder a procedimientos almacenados en el Servidor.

La tecnología de paso a través de SQL es la mejor herramienta para crear conjuntos de resultados de sólo lectura y para utilizar cualquier otra sintaxis nativa de SQL. La tabla 3.1 enumera las funciones de paso a través de SQL de Visual FoxPro:

La herramienta de Visual FoxPro proporciona además de programación estándar por procedimientos, la potencia y la flexibilidad propias de la programación orientada a objetos.

El diseño orientado a objetos y la programación orientada a objetos representan un cambio de perspectiva con respecto a la programación estándar por procedimientos. En lugar de pensar sobre el flujo del programa desde la primera hasta la última línea de código, se debe pensar en la creación de objetos, componentes autocontenidos de una aplicación que tienen funcionalidad privada además de la funcionalidad que se puede exponer al usuario.

En Visual los controles son objetos que puede incluir en sus aplicaciones. Es posible manipular estos objetos a través de sus propiedades, eventos y métodos.

Las mejoras en el lenguaje orientado a objetos de Visual FoxPro proporcionan un mayor control sobre los objetos de las aplicaciones. Asimismo, facilitan la creación y el mantenimiento de bibliotecas de código reutilizable, proporcionando:

- Código más compacto
- Incorporación más sencilla del código a las aplicaciones sin necesidad de elaborar esquemas de asignación de nombres.
- Menor complejidad a la hora de integrar código de distintos archivos en una aplicación.

Tarea	Función	Objetivo
Administración	SQLCONNECT()	Se conecta a un origen de Datos para operaciones de paso a través de SQL
De	SQLSTRINGCONNECT()	Se conecta aun origen de Datos usando la sintaxis de cadena de conexión ODBC.
Conexiones	SQLDISCONNECT()	Rompe una conexión con un origen de Datos ODBC, haciendo obsoleto el controlador de la conexión especificada.
Ejecución y	SQLCANCEL()	Cancela una consulta SQL que se está ejecutando en una conexión activa.
control de la	SQLEXEC()	Ejecuta una consulta de paso a través de SQL en una conexión activa.
instrucción SQL	SQLMORERESULTS()	Introduce otro conjunto de resultados en el cursor
	SQLCOMMIT()	Solicita una confirmación de transacción.
	SQLROLLBACK()	Solicita una anulación de la transacción.
Información del Origen de Datos	SQLCOLUMNS()	Almacena en un cursor una lista de nombres de columnas e información sobre cada una.
	SQLTABLES()	Almacena en un cursor los nombres de tablas del origen.
Control diversos	SQLGETPROP()	Obtiene una propiedad de conexión desde una conexión activa.
	SQLSETPROP()	Establece una propiedad de una conexión activa.

Tabla 3.1. Funciones de paso a través de SQL de Visual FoxPro.

La programación orientada a objetos es en gran medida un modo de empaquetar código de manera que se pueda volver a utilizar y mantener más fácilmente. Los paquetes principales se llaman clases.

Las clases y los objetos están estrechamente relacionados, pero no son lo mismo. Una clase contiene información sobre cual debe ser la apariencia y el comportamiento de un objeto. Una clase es el plano o el esquema de un objeto. Por ejemplo, el esquema eléctrico y de diseño de un teléfono sería algo similar a una clase. El objeto, o una instancia de la clase, sería el teléfono (La clase determina las características del objeto).

Los objetos tienen ciertas propiedades o atributos. Por ejemplo, un teléfono tiene un color y un tamaño determinados. Cuando se instala un teléfono en la oficina, tienen una determinada posición sobre la mesa. El receptor puede estar colgado o descolgado.

Los objetos que se crean en Visual FoxPro también tienen propiedades que están determinadas por la clase en la que se basa el objeto. Estas propiedades pueden establecerse durante el tiempo de diseño o durante el tiempo de ejecución.

Por ejemplo, en la tabla siguiente se indican algunas propiedades que puede tener una casilla de verificación

Propiedades	Descripción.
Caption	Texto descriptivo que aparece junto a la casilla de verificación.
Enabled	Si un usuario puede elegir o no la casilla de verificación.
ForeColor	Color de texto de título.
Left	Posición del extremo izquierdo de la casilla de verificación.
MousePointer	Apariencia del puntero del mouse cuando está situado sobre la casilla de verificación.
Top	Posición de la parte superior de la casilla de verificación
Visible	Si la casilla de verificación es visible o no.

Cada objeto reconoce y puede responder a determinadas acciones denominadas eventos. Un evento es una actividad específica y predeterminada, iniciada por el usuario o por el sistema. Los eventos, en la mayor parte de los casos, se generan por interacción del usuario. Por ejemplo, con un teléfono, se desencadena un evento cuando un usuario descuelga el receptor. Los eventos también se desencadenan cuando el usuario presiona los botones para efectuar una llamada.

En Visual FoxPro, las acciones del usuario que desencadenan eventos son por ejemplo un Clicks, movimientos del mouse y pulsaciones de teclado. La creación de un objeto y la ejecución de una línea de código que produce un error son eventos iniciados por el sistema.

Los métodos son procedimientos asociados a un objeto. Los métodos se diferencian de los procedimientos normales de Visual FoxPro en que están vinculados intrincadamente a un objeto y tienen nombres distintos que los procedimientos normales de Visual FoxPro

Los eventos pueden tener métodos asociados. Por ejemplo, si escribe código de método para el evento Click, ese código se ejecutará cuando se produzca el evento Click. Los métodos también pueden existir independientemente de los eventos. Se debe llamar a estos métodos de forma explícita en el código.

El conjunto de eventos es limitado, no es posible crear nuevos eventos. Sin embargo, el conjunto de métodos puede ampliarse indefinidamente.

La tabla siguiente muestra algunos de los eventos asociados a una casilla de verificación.

Evento	Descripción.
Click	El usuario hace clic en la casilla de verificación.
GotFocus	El usuario selecciona la casilla de verificación.
LostFocus	El usuario selecciona otro control.

La tabla siguiente muestra algunos métodos asociados a una casilla de verificación.

Método	Descripción
Refresh	El valor de la casilla de verificación se actualiza para reflejar los cambios que se puedan haber producido en el origen de Datos subyacente.
SetFocus	El enfoque se establece en la casilla de verificación como si el usuario hubiera presionado la tecla TAB hasta seleccionar la casilla de verificación.

Todas las propiedades, eventos y métodos de un objeto se especifican en la definición de clase. Además, las clases tienen las siguientes características que las hacen especialmente útiles para crear código reutilizable y fácil de mantener:

- Encapsulamiento
- Subclases
- Herencia

El encapsulamiento, empaqueta el código de métodos y propiedades en un mismo objeto, contribuye a la abstracción. Por ejemplo, las propiedades que determinan los elementos de un cuadro de lista pueden encapsularse en un único control que se agrega a un formulario.

La creación de subclases es un modo de reducir la cantidad de código que hay que escribir. Puede comenzar definiendo un objeto que sea similar al deseado y personalizándolo.

Con la herencia, si realiza un cambio en una clase, ese cambio se reflejará en todas las subclases que se basen en ella. Esta actualización automática ahorra tiempo y trabajo.

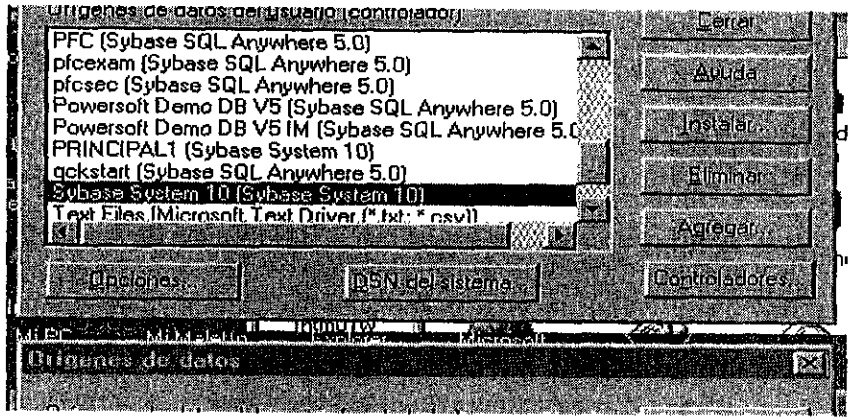
La herencia no funciona con el hardware, pero sí en el software. Si descubre un error en una clase, en lugar de tener que cambiar el código de todas las subclases podrá arreglarlo una sola vez en la clase y el cambio se propagará a todas las subclases pertenecientes a ella.

III.1.3 Construcción Conceptual

Para determinar los elementos necesarios en términos generales para poder llevar a cabo el desarrollo del administrador de la Base de Datos debemos hacer una construcción de prueba.

Para realizar la consulta de la información de la tabla de Tema necesitamos:

Dar de Alta el Origen de Datos en el administrador de ODBC que nos permita conectar a SyBase desde VisualFox.



En este administrador podemos agregar un Origen de Datos, basado en algún controlador en este caso SyBase System 10, y los parámetros correspondientes de identificación básicos:

- Nombre de la fuente de datos
- Descripción
- Nombre del Servidor

Probar en la herramienta de interacción WISQL un Store Procedure, por ejemplo "ExtraeTema" que seleccione la información de la tabla Tema.

Desde Visual Fox probar que:

- Conecte a la Base de Datos.
- Ejecute Store Procedure "ExtraeTema".
- Desconecte de la Base de Datos.

- Regrese los Datos resultantes.

El código a detalle ya como parte de un módulo, se presenta más adelante.

III.1.4 Análisis de Riesgo

No cabe duda que un Sistema Administrador de la Base de Datos es indispensable para el mantenimiento eficiente de la información. La conceptualización inicial del sistema no muestra ningún riesgo, sino por el contrario aprovecha las ventajas que nos proporciona la arquitectura Cliente/Servidor. Por un lado se manejarán los procesos de acceso y actualización directos sobre la Base de Datos en el mismo Servidor. Y los resultados que el Servidor proporcione serán únicamente presentados de la forma más amigable por el Cliente.

La comunicación entre el Cliente y el Servidor requiere del controlador estándar de Windows ODBC, y que tan sólo se requeriría dar de alta el Origen de Datos correspondiente en las computadoras en las que el administrador tuviese acceso. Inclusive pudiera formar parte de algún prototipo de todas las computadoras de la institución para poder tener acceso al sistema desde cualquier punto y así resolver algún problema inmediatamente.

Dentro del mismo prototipo que se necesitaría incluir serían tanto librerías de SyBase como de VisualFox que después podremos detallar.

En general, los costos son mínimos ya que se tienen todas las herramientas para el desarrollo del sistema, tan solo hay que emplear cierto tiempo para incluir los elementos necesarios en cada computadora para poder operar el sistema.

III.2 Segundo Ciclo

El sistema administrador contará con funciones de una complejidad mínima por lo que en este ciclo se realizaran simultáneamente los subsistemas requeridos, dejando para después algunos detalles de presentación más sofisticados, al gusto de los usuarios y que la herramienta nos vayan permitiendo realizar.

III.2.1 Requerimientos del Sistema

Haciendo una revisión con los usuarios y con antecedentes de trabajo con otros sistemas, se han identificado los procesos básicos que se pueden realizar sobre las Bases de Datos:

Dar de *Alta* una nueva aplicación implica no sólo insertar en la tabla de *Aplicaciones* un registro, sino también establecer los usuarios que tendrán acceso a tal *Aplicación*; además darlos de *Alta* en la tabla de Uso para relacionarlos y establecer sus privilegios de trabajo sobre la *Aplicación*.

Un usuario que accesa cierta *Aplicación* sólo puede consultar la información pero ahora debido a cambios de puestos, se necesita que nuestro usuario pueda modificar los Datos. En este caso se debe hacer un *Cambio* en la tabla de Uso para ajustar los privilegios del usuario en cuestión.

Por lo general cuando un sistema deja de operar, los usuarios nos lo comunican y se procede a eliminar la *Aplicación* de la red. Con el nuevo enfoque que se plantea, además se debe dar de *Baja* de las tablas la información correspondiente para anular el acceso.

En ocasiones se pueden solicitar consultas de Búsqueda que requieran ciertos parámetros. Por ejemplo se desea saber a que Aplicaciones tiene derecho de acceso algún usuario para determinar si puede o no trabajar con cierta información.

Se pueden identificar cuatro procesos básicos sobre los que nuestro sistema debe ser desarrollado y que son de las tareas básicas que se pueden realizar sobre una Base de Datos: *Alta, Baja, Cambio, Búsqueda.*

Se debe considerar una clave de acceso única que establezca cierta seguridad sobre el uso del sistema y un mecanismo para la solicitud de ajustes a la información por parte de los usuarios, que permita al administrador realizar alguna aclaración.

III.2.2 Diseño Lógico y Físico

En Base a los requerimientos se considera la implementación de los siguiente elementos:

En el Cliente:

Una ventana de respuesta en la que se solicite la clave y el password de acceso al sistema.

Una ventana principal en la que se pueda seleccionar la tabla con la que se desea trabajar.

Una ventana en la que se pueda especificar la operación a realizar pudiendo visualizar la información de la tabla o el resultado de alguna búsqueda.

Un módulo que permita conectarse y desconectarse a la Base de Datos.

Un módulo que determine que procedimiento almacenado quieren que se ejecute.

Para la elaboración de la interfaz en el Cliente emplearemos los elementos de VisualFox. Actualmente, herramientas tales como VisualFox o PowerBuilder proporcionan un ambiente gráfico orientado al manejo de objetos para el desarrollo de aplicaciones que muestren los elementos estándar de Windows.

De este modo, se pueden elegir los elementos que formarán parte de nuestra interfaz de una forma más eficiente:

- Ventanas
- Menús
- Botones de Comando
- Cuadros de Edición
- Gráficos
- Tablas
- Funciones
- etc.

Estos elementos tienen una función específica la cual es controlada mediante Eventos en los cuales podemos establecer código que convenga para nuestra aplicación, generalmente llamados *Script*. Por ejemplo el evento *Open* que sucede cuando una ventana es abierta. En ese momento si nos es útil que se realice algún proceso lo incluimos en el *Script* del *Evento Open* para que se lleve a cabo.

También cuentan con Propiedades las cuales determinan las características de los elementos. Por ejemplo Título, Color, Visible, Habilitado, Fuentes de Letra, Tamaño, etc. Y que se pueden definir en el momento de la construcción del elemento o al tiempo de ejecución en los *Scripts* de los Eventos.

En el Servidor:

Los procedimientos almacenados para cada una de las tablas de Datos:

Extraer toda la información.

- Dar de Alta.

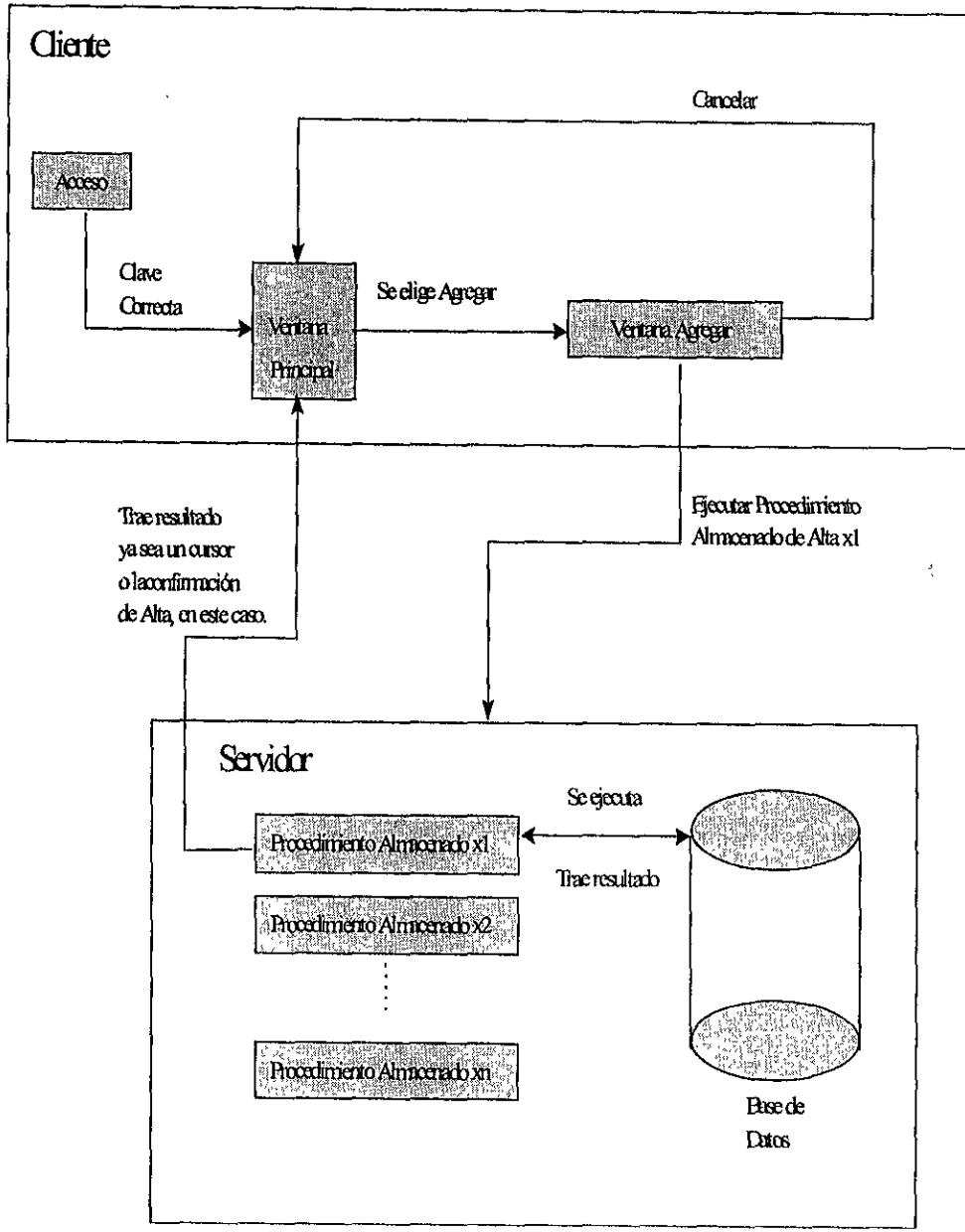
- Dar de Baja.
- Actualizar los Datos de algún registro.
- Los procedimientos almacenados para algunas tablas en las que vale la pena.
- Seleccionar información bajo cierto criterio.

Los procedimientos almacenados en la Base de Datos nos permiten definir basado en estatutos SQL un programa, que pueda ser usado por todas las aplicaciones. Usando procedimientos almacenados las actualizaciones permiten alcanzar un buen nivel de seguridad, integridad, y desempeño. Entonces los procedimientos almacenados para ejecución condicionada, también pueden usarse para fortalecer reglas adicionales del negocio.

Para la creación de los procedimientos almacenados cabe considerar:

Operación/Proceso	Estatutos SQL
Alta	Corresponde a un INSERT.
Baja	Es un DELETE de registros.
Cambio	Se actualiza un registro con UPDATE.
Búsqueda	Es un SELECT de Datos con una condición particular.

Considerando el Procedimiento Almacenado para el Alta en la tabla de Usuarios, la secuencia lógica correspondiente para la construcción del mismo se puede observar en el diagrama 3.2.a.



3.2.a Diagrama de flujo de datos correspondiente al procedimiento de alta.

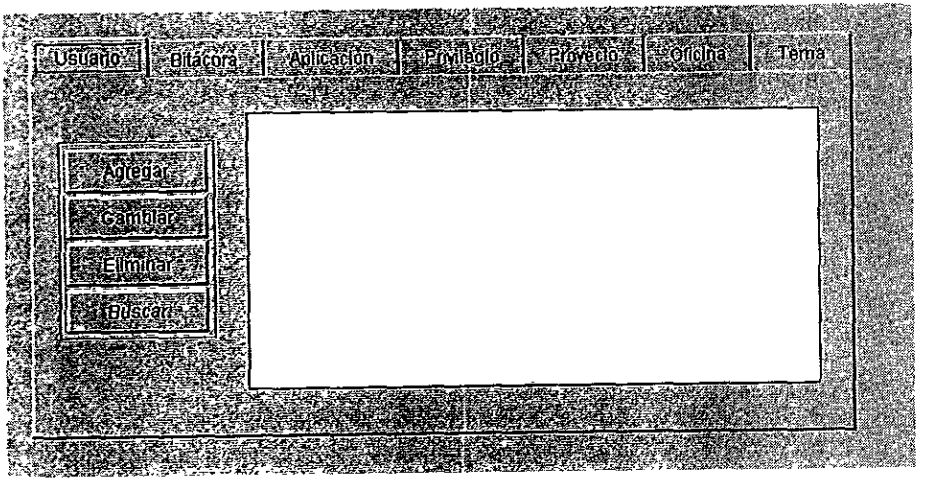
De forma similar se establece el flujo de Datos del resto de los *procedimientos.*

III.2.3 Segunda Construcción

Como prototipo de prueba se inicia el *desarrollo de los siguientes* módulos:

Una ventana principal para elegir la operación de Agregar un registro en la tabla de Usuarios.

Haciendo una revisión de los elementos que proporciona VisualFox, consideramos que la siguiente *ventana permite* la operación adecuada del administrador.



Cada carpeta corresponderá a una tabla de la Base de Datos y los botones especifican la operación a realizar.

El módulo de conexión y desconexión de la Base de Datos correspondientes.

```
/* realiza la conexión */
Store SQLCONNECT('ServidorX','usuarioX,'claveX') TO gnConnHandle
/* si no fue exitosa */
IF gnConnHandle <= 0
        /* mensaje de error */
        = MESSAGEBOX('No se pudo hacer la conexion', 16, 'Conexion SQL
Erronea')
        /* regresa con falso */
        Return .f.
/* si fue exitosa */
ELSE
        = MESSAGEBOX('Conexión realizada', 48, 'Mensaje de conexión SQL'
        /* cambiar a la Base de Datos BaseX */
        xexxec = 'use BaseX'
/* realiza la instrucción */
        xxres=SQLEXEC(gnconnhandle,xexxec,'result')
        /* si no fue exitosa */
        IF xxres < 0
                /* muestra el error */
                Do ErrorSQL
                /* cierra tablas de Visual Fox */
                close data
                /* realiza desconecta */
                = SQLDISCONNECT(gnConnHandle)
                WAIT WINDOW 'Desconectándose de SyBase' TIMEOUT 2
                        Return          /* si fue exitosa */
ELSE
WAIT WINDOW 'Conectándose a  BD  BaseX' TIMEOUT 2
ENDIF
ENDIF
```

Este módulo muestra la forma de conectarse y desconectarse con estatutos de VisualFox. Las funciones básicas empleadas para interactuar con la Base de SyBase son:

SQLCONNECT (nombre del Servidor, clave de usuario, password del usuario)

SQLDISCONNECT(identificador de conexión que genera SQLCONNECT)

/* El módulo que mande llamar un Store Procedure que agregue registros a la *tabla Usuarios* */

```
xexxec = 'exec AltaUsuario  'SI2AAC','Antonio Amezcua Chávez','tono', 1, 0,
12345, 624, 0, 0 '
xxres=SQLEXEC(gnconnhandle,xexxec,'result')
IF xxres < 0
Do ErrorSQL
        close data
        = SQLDISCONNECT(gnConnHandle)
        WAIT WINDOW 'Desconectándose de SyBase' TIMEOUT 2
        Return
ELSE
WAIT WINDOW ' Alta  completada' TIMEOUT 2
ENDIF
```

En esta parte la función más importante es :

SQLEXEC(identificador de conexión, comando SQL a ejecutar, cursor de resultados)

Store Procedure para dar de Alta registros en Usuarios.

```
/*AltaUsuario*/
Create procedure AltaUsuario
        @cve_usu        varchar(8)      = null,
        @nom_usu        varchar(50)     = null,
        @pass_usu       varchar(10)     =null,
        @con_tot                tinyint(1)          =null,
        @con_uso        tinyint(1)      =null,
        @dia_val        char(7)         =null,
        @hor_val        char(4)         =null,
        @mul            tinyint(1)      =null,
        @ext            tinyint(1)      =null
as
if @cve_usu is null or @nom_usu is null or @pass_usu is null or @con_tot is null or
@con_uso is null or @dia_val is null or @hor_val is null or @mul is null or @ext is null
print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
else
if exists (select * from Usuario where ClaveUsuario = upper(@cve_usu))
        print "@@@ Clave de usuario ya existe @@@"
else
        begin
insert into Usuario (ClaveUsuario, Nombre,Password, ConecTotales, ConecUso,
DiasValidos, HoraValida, Multi, Externo)
values (upper(@cve_usu), @nom_usu, @pass_usu, @con_tot , @con_uso ,
@dia_val , @hor_val , @mul , @ext )
print "Alta exitosa"
end
```

III.2.4 Evaluación

En esta etapa se aprecia la automatización de los procesos de administración de la Base de Datos de una forma amigable y eficiente. Las ventanas de la interfaz son lo más comprensibles para llevar a cabo las tareas.

La ejecución de los procesos almacenados no tiene complicación dentro del código de nuestro Cliente. Tan solo hay que parametrizar los datos involucrados en cada operación, para que el procedimiento almacenado se ejecute.

Los procedimientos almacenados nos traen un resultado ya sea un mensaje o una tabla de Datos resultantes que es fácil presentar en la interfaz gráfica del usuario.

III.3 Ciclo Final

3.3.1 Requerimientos Finales

Como requerimientos finales se requiere detallar aspectos tales como:

- Definir los elementos de apoyo para que la Aplicación se ejecute sin problemas.
- Crear las interfaces y mensajes necesarios para cada proceso en la interfaz del Cliente.

Aunque el sistema este formado de elementos no complicados para su uso se debe establecer la Ayuda. De este modo se cubrirá cualquier duda sobre la operación del sistema.

Establecer los procedimientos almacenados para el resto de los procesos.

En coordinación con el usuario se considera que no hay más procesos por el momento que el sistema pueda cubrir. Por lo que sólo resta armar el prototipo final para su puesta en marcha.

III.3.2 Diseño Final

Una vez diseñada la columna vertebral del sistema, podemos revisar el diseño de algunos detalles.

Considerando el diseño de nuestra ventana de selección de la operación y presentación de Datos, apreciamos la necesidad de crear otras ventanas para dar una mayor claridad a la operación de sistema.

En la ventana principal para dar de alta a un Usuario requerimos de la pantalla de Datos, para editar la información, y con el botón Agregar, darlos de alta físicamente en SyBase, si se presento una buena captura de los Datos.

De acuerdo con las operaciones contempladas hay que considerar:

- Agregar: implica la necesidad de una pantalla de captura de los datos y la opción de actualizar o cancelar la operación.
- Eliminar: en este caso es recomendable ubicar el registro que se desea borrar y confirmar si desea realizar la eliminación.
- Cambiar: el cambiar involucra las consideraciones de Borrar y Agregar básicamente la información con los cambios.
- Buscar: para la búsqueda en la pantalla de captura sólo hay que incluir los campos llave y presentar en la pantalla de datos el resultado.

Dependiendo de la tabla seleccionada, las ventanas de captura de datos variarán por la diferencia de la estructura de cada una.

Hay que considerar el caso en el que se intenten realizar dos operaciones similares desde dos Clientes diferentes al mismo tiempo. Por esta situación se puede duplicar información y salir de las reglas de normalización. Entonces debemos implementar en los procedimientos almacenados la verificación de la no existencia de la información en la tabla de las mismas características de la planteada en la operación que se desea realizar, en otro caso restringir la inserción, en otras palabras para Agregar información se requiere que esta no exista en la tabla pero cuando se desea Borrar algún dato este debe existir en la tabla.

La ventana de entrada solicita un password único para dar acceso al sistema sólo a las personas que tengan conocimiento del mismo, este Sistema no será del dominio publico.

Para el resto de los procedimientos almacenados debemos de considerar el diseño lógico y físico que se plantea a continuación:

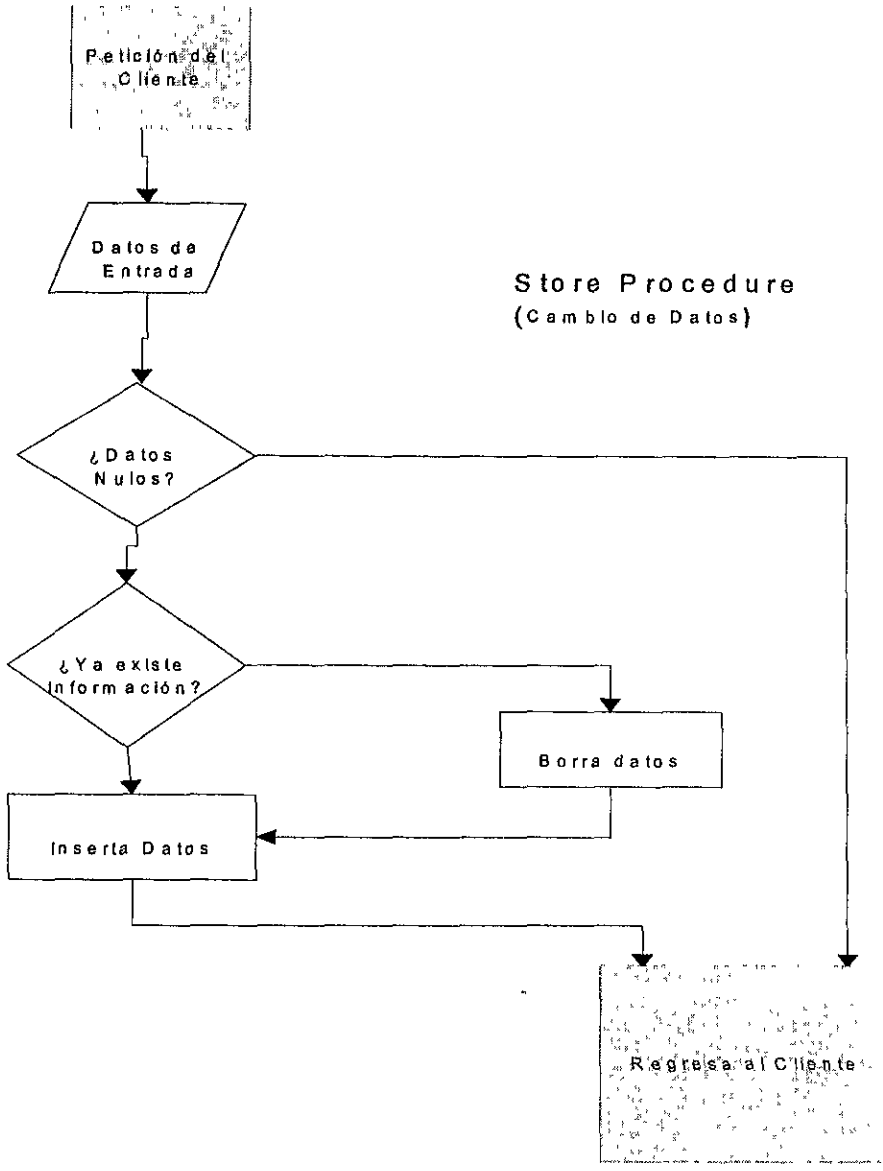


Diagrama de Flujo de Datos para las Modificaciones.

Se debe considerar la información inicial un procedimiento almacenado que tomo la información completa de las tablas, a excepción de la Bitácora que es una tabla que puede crecer *demasiado y sólo tendría caso* extraer la información del día.

III.3.3 Tercera Construcción

La *ventana* de entrada requiere de una clave para permitir acceso al sistema.

Se establece un menú principal para incluir por el momento dos opciones:

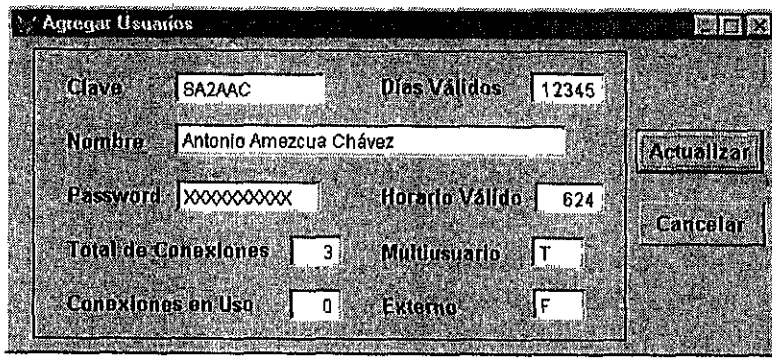
- Ayuda. Para el despliegue de la ayuda.
- Salir. Para terminar la sesión con el sistema.

La ayuda se integra en un documento del paquete Word que cuenta con una herramienta Doc To Help que permite la creación de archivos de ayuda estilo Windows. Este archivo es llamado a partir de la instrucción

SET HELP TO <nombre del Archivo> HLP

Considerando la tabla de Usuarios se muestran las ventanas finales para cada una de las operaciones que así lo requieren.

Para el Alta de Usuarios y Cambio de Usuarios se tiene la siguiente pantalla de captura. Cuando se hace click en el botón de Agregar de la pantalla principal aparece esta ventana pero sin Datos cuando se desea realizar un Alta y con los datos del registro activo en la tabla (que son los que se desean modificar) cuando se desee realizar un Cambios de datos. El botón de Actualizar realiza los cambios directamente en la Base de Datos en SyBase y no en la temporal de Visual que tan sólo es una imagen de la real.

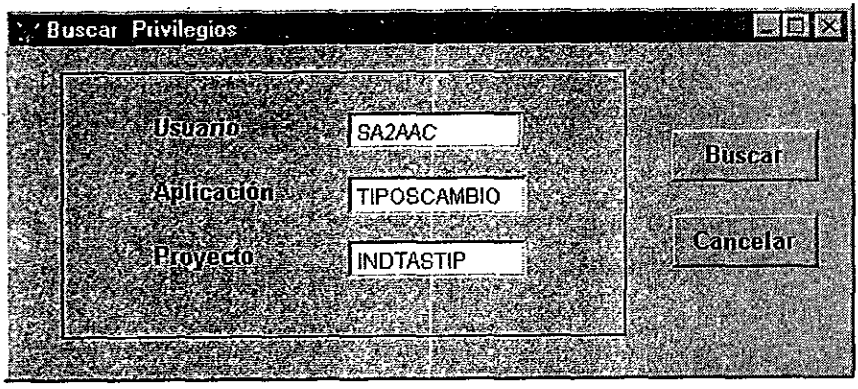


En este caso se realizan validaciones con respecto al tipo de dato de los campos y en particular a campos que tienen valores específicos como:

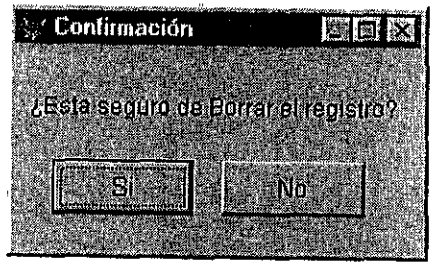
Días Válidos: 1 Lunes, 2 Martes,... 5 Viernes Ej. 1234

Horario Válido: ii Hora Inicial + ff Hora Final Ej. 0612

Los parámetros para realizar una Búsqueda se solicitan en la siguiente pantalla y los resultados se muestran en la carpeta respectiva a la tabla de la pantalla principal. Una búsqueda muy recurrida es la de los Privilegios que tienen cada usuario, a continuación se muestra la ventana de parámetros.



En el caso de eliminación de datos requerimos de una ventana para confirmar tal operación, y evitarle al usuario errores que puedan retrasar su trabajo. Dicha ventana de tipo respuesta cancela o ejecuta el proceso MessageBox() con los respectivos parámetros.



Todos los procedimientos almacenados que desglosamos a continuación se ubican en la propia Base de Datos. Por la diferencia de estructura de las tablas es difícil parametrizar y tener tan solo un procedimiento tanto para Alta, Baja o Cambio.

Para la presentación inicial de datos se mostrarán todos los datos de cada tabla. Se crearon procedimientos almacenados para cada una de las tablas similar al siguiente:

```
/*ExtraeAplicacion*/
Create procedure ExtraeAplicacion
as
select   *
from     Aplicacion
```

Para el resto de las tablas hay que sustituir únicamente el nombre de la tabla por el correspondiente, así como el nombre del procedimiento; estos se podrán ver en el anexo A.

III.3.4 Prueba Final

Inicialmente, se realizaron pruebas de todos los módulos del sistema para verificar su comportamiento en forma individual. La idea es localizar posibles errores a

nivel detalle comparándolo contra las especificaciones de diseño, para evitar un mayor esfuerzo en la detección de errores si la prueba fuera global.

Una vez terminada la construcción del sistema por completo se evalúa su funcionamiento. Revisando los objetivos iniciales en primera instancia cumple con lo siguiente:

Agiliza la Administración de las Bases de Datos.

Permite a un usuario sin conocimiento de SQL, realizar los ajustes sobre las tablas pero con restricción en el acceso.

Se organizó los accesos a las Aplicaciones de la institución en un sistema Cliente/Servidor.

Además se consideraron los factores que determinan la calidad del software:

- Fiabilidad: el sistema puede llevar a cabo funciones esperadas con la precisión requerida.
- Eficiencia: los recursos de hardware y software disponibles son aprovechados de forma óptima.
- Integridad: se tienen requerimientos de software particular para la ejecución del sistema y se establece un nivel de seguridad en el Cliente.
- Facilidad de Uso: el esfuerzo es mínimo para aprender a manejar e interpretar las entradas y resultados del sistema.
- Facilidad de Mantenimiento: el código y los módulos son lo suficientemente claros para identificar errores.
- Flexibilidad: en el caso de requerirse adaptar el sistema a otra plataforma, se deben considerar los alcances tanto de VisulaFox como de SyBase. Si los requerimientos de la plataforma son cubiertos se realizarían los ajustes de configuración.
- Reutilización: muchos de los módulos de conexión principalmente pueden ser empleados en otras aplicaciones que se migren a una plataforma Cliente/Servidor.

La Prueba Global se basó en un enfoque de Caja Negra que se concentra en las funciones que debe cumplir el sistema y si produce las salidas deseadas. La idea es establecer un conjunto de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales del sistema.

Se intentaron encontrar fallas planteando situaciones como:

¿Qué sucede cuando no existe conexión con el Servidor?

Simplemente se le indica al usuario que no hay tal conexión en un cuadro de mensajes.

¿Cuándo dos usuarios dan de Alta una misma clave en una tabla al mismo tiempo, que pasa?

Uno de los dos va a procesar primero su operación, y el otro recibirá un mensaje de que ya existe la información.

En general se probaron para cada una de las tablas todas las operaciones especificadas y damos cuenta de este modo del adecuado funcionamiento integral del sistema.

Capítulo IV

Sistema Administrador de Aplicaciones

Llegamos a la parte esencial de este trabajo, es aquí en donde todo lo construido toma sentido, para lograr darles a los usuarios una herramientas que contribuya a un mejor desarrollo de sus actividades.

Es el momento de la construcción del Sistema Administrador de Aplicaciones, cuyas actividades principales serán: conjuntar todas las Aplicaciones internas de la dirección, logrando mayor agilidad de acceso a las consultas de los usuarios, además de proporcionar mayor conocimientos de la información con la que se cuenta en la institución, y a la cuál pueden tener acceso, en caso de que por cualquier motivo algún usuario ignorara su existencia.

De esta manera también se tendrá un solo acceso, y por lo tanto mayor control a los derechos de los usuarios. La información se encontrará de manera ordenada y clasificada, por lo que resultará comprensible para los usuarios.

El "Sistema Administrador de Aplicaciones" se realizará fundamentado en las herramientas hasta este momento mencionadas, ya que son la base de todas las Aplicaciones de nuestra área. Esta Aplicación es un esfuerzo por tener un sistema de Información accesible a todos nuestros usuarios que contribuya a romper barreras de Información y con ello ampliando la visión del análisis.

IV.1 Primer Ciclo

IV.1.1 Análisis

IV.1.1.1 Requerimientos del Negocio

Actualmente las Aplicaciones que se desarrollan, se realizan a solicitud de alguna oficina en particular y ellos son los usuarios de la información que se genera con esta Aplicación. El acceso normalmente se realiza a través de un icono y el usuario se tiene que identificar (para determinar si el usuario es valido y si no ha rebasado el número de sesiones autorizadas) al inicio de la sesión. Esto mediante un proceso de registro que involucra tanto a los usuarios con las respectivas Aplicaciones autorizadas.

Cuando alguien más requiere de la información se tiene que poner el acceso directo y realizar el registro correspondiente para poder tener acceso y esto se repite para cada usuario que requiera de la información y para cada Aplicación.

Cabe resaltar que por algún motivo existen usuarios que la requiere de cierta información y que desconoce que ya existe, y por lo cual duplican trabajo.

Para evitar estos inconvenientes, se realizará un sistema que con un solo acceso les permita observar catálogos de la información que se encuentra disponible y a la cuál pueden tener acceso si lo requirieran ya sea para consulta o para algún otro proceso más complejo.

Para lo cual se aprovecha la naturaleza de la información para mostrar a los usuarios que no estén familiarizados, de que tipo de información se trata. Los sistemas se esquematizan de la siguiente manera: existen temas generales en los que se manejan Proyectos; dentro de un Proyecto existen varias Aplicaciones a las cuales algunos usuarios tienen acceso para realizar tareas específicas como: realizar integración de información, extracciones de datos, realizar algún reporte, etc.

Se requiere que la información se ponga a disposición de todos los usuarios para su máximo aprovechamiento, evitando tener que realizar tantas instalaciones como Aplicaciones existan para cada usuario, y de igual manera evitar que los usuarios tengan que realizar tantos accesos como aplicaciones se tengan.

Aunque estos requerimientos parecen simples, el aprovechamiento que de esto resulta, será de gran ayuda para el ahorro en tiempo, tanto de accesos como de la difusión de nuevas Aplicaciones. Y así contribuir a un Sistema de Información más completo y eficaz.

Nos resultara más claro este análisis tomando en cuenta, el análisis realizado en la creación de la estructura de la Base de Datos (capítulo II), y con el análisis realizado en la creación del "Sistema Administración de la Base de Datos" (capítulo III).

Así nuestro objetivo se reduce a la explotación de la metainformación que se tiene para darles mejor servicio a los usuarios de nuestra área.

Lo anterior se logrará enterando a los Usuarios de la estructura de la información. Para lo cual se plantea el crear una interfaz particular para cada usuarios, que aunque aparentemente sea igual para todos, estará restringida en derechos de acceso a cada Aplicación; esto quiere decir que todos los usuarios puedan observar la estructura que se tienen de la información, pero sólo los que tengan privilegios puedan accesarla. No es otro el motivo que el proteger a la información de un mal uso.

Cualquier persona que lo solicite podrá tener privilegios, a todas las aplicaciones siempre y cuando la información le sea necesaria para algún análisis, y de esta manera mantener cierto control, se incluirán claves de consulta con las que podrán accesar a las Aplicaciones que no realicen alteraciones a la información, y dichas claves serán del dominio de todos los usuarios.

Se tiene que tomar en cuenta que un simple dato puede no proporcionar ninguna información, por lo que resulta conveniente se muestre una breve descripción de las Aplicaciones, para mayor comprensión.

IV.1.1.2 Requerimientos del Sistema

Debido a los requerimientos del sistema, a la plataforma de los datos y siguiendo con los lineamientos anteriormente expuestos; resulta de gran conveniencia realizar la explotación de la información realizando una aplicación cliente-servidor, que aprovecha las ventajas de nuestra actual plataforma y de las herramientas de desarrollo con que se cuenta.

Como se definió en el capítulo II la información se localiza en un servidor de Sybase. Por lo que un requerimiento importante es tener una adecuada comunicación con el servidor; para lo cual se requiere incluir en el prototipo de las computadoras de los usuarios, la definición de un ODBC (Open Data Base Connectivity) llamado Origen de Datos, el cual tiene las especificaciones de la comunicación con el servidor.

Se aprovechará el ODBC definido en el capítulo anterior del "Sistema Administrador de la Base de Datos".

Anteriormente se vio la conveniencia de utilizar a VisualFox como nuestro cliente, ya que combina la potencia, la velocidad, la interfaz gráfica de usuario y las sofisticadas funciones de consulta, informes y procesos con el acceso multiusuario, robusto proceso de transacciones, inicio de sesiones y la sintaxis nativa del servidor de un origen de datos o servidor ODBC.

IV.1.2 Diseño

IV.1.2.1 Diseño Conceptual

Se desea realizar una interfaz gráfica que permita accesar al almacén de datos de nuestro metadatos (descripción del contenido de nuestras Bases de Datos), con el fin de proporcionar a los usuarios mayor agilidad en el acceso a las

aplicaciones propias de la institución, logrando también mantener enterados a los usuarios de la información que existe y proporcionar los medios para su consulta.[5]

Evidentemente no tendrán derechos de accesar a todas las aplicaciones todos los usuarios en una primera instancia, por cuestiones de seguridad, por lo que se tendrá que llevar a cabo un registro de entrada al sistema, el cual obtenga la clave del usuario y con ella sabremos a que información tienen derecho de accesar. Sin olvidar que uno de nuestro objetivo es dar a conocer toda la información existente, con una breve descripción para que si lo requieren pidan los derechos correspondientes a dicha información.

Se mostrará el catálogo de proyectos que se tiene, sus aplicaciones, y una pequeña definición que haga más clara la información. Y por supuesto permita el acceso a las Aplicaciones desde éste punto, evitando tener que salir, localizar la Aplicación e instalarla para su acceso; para cada computadora de los usuarios.

IV.1.2.2 Diseño Lógico

Ahora veremos como los requerimientos del "Sistema Administrador de Aplicaciones" se representan en entidades lógicas (o modulos) que muestran el mecanismo para garantizar la integridad de la información.

Los requerimientos piden una entidad de acceso, en la que el usuario se identificará, para que con esa información vaya a la Base de Datos en Sybase, verifique que el usuario es valido y traiga la información correspondiente de su sesión particular; esto se refiere a los privilegios que tienen de cada Aplicación, y los catálogos de la totalidad de la información que hay en ese momento, siempre que los usuarios sean internos, en caso que sean externos solo verán lo que pueden accesar.

Podemos definir claramente un estado principal en el que se pueda elegir de la lista de Proyectos el que se requiere, y de ahí pasar a otro estado en el que se pueda elegir de la lista de Aplicaciones del Proyecto seleccionado. En este momento identificamos un Subestado, en el que se realiza la llamada a ejecutar la Aplicación que se ha seleccionado, aquí nos limitaremos a la solo definición de la llamada a la ejecución y no a la ejecución misma, ya que se sale de los límites del sistema que estamos tratando en este momento.

IV.1.2.3 Diseño Físico

La conexión al servidor Sybase es imprescindible en este sistema ya que ahí se encuentra toda la información, tanto de los usuarios, como de los Proyectos y Aplicaciones.

El proceso se iniciará con un modulo de programación estándar que realiza el proceso de inicicialización de ambiente y de variables, la conexión con el servidor antes de cualquier interfaz gráfica.

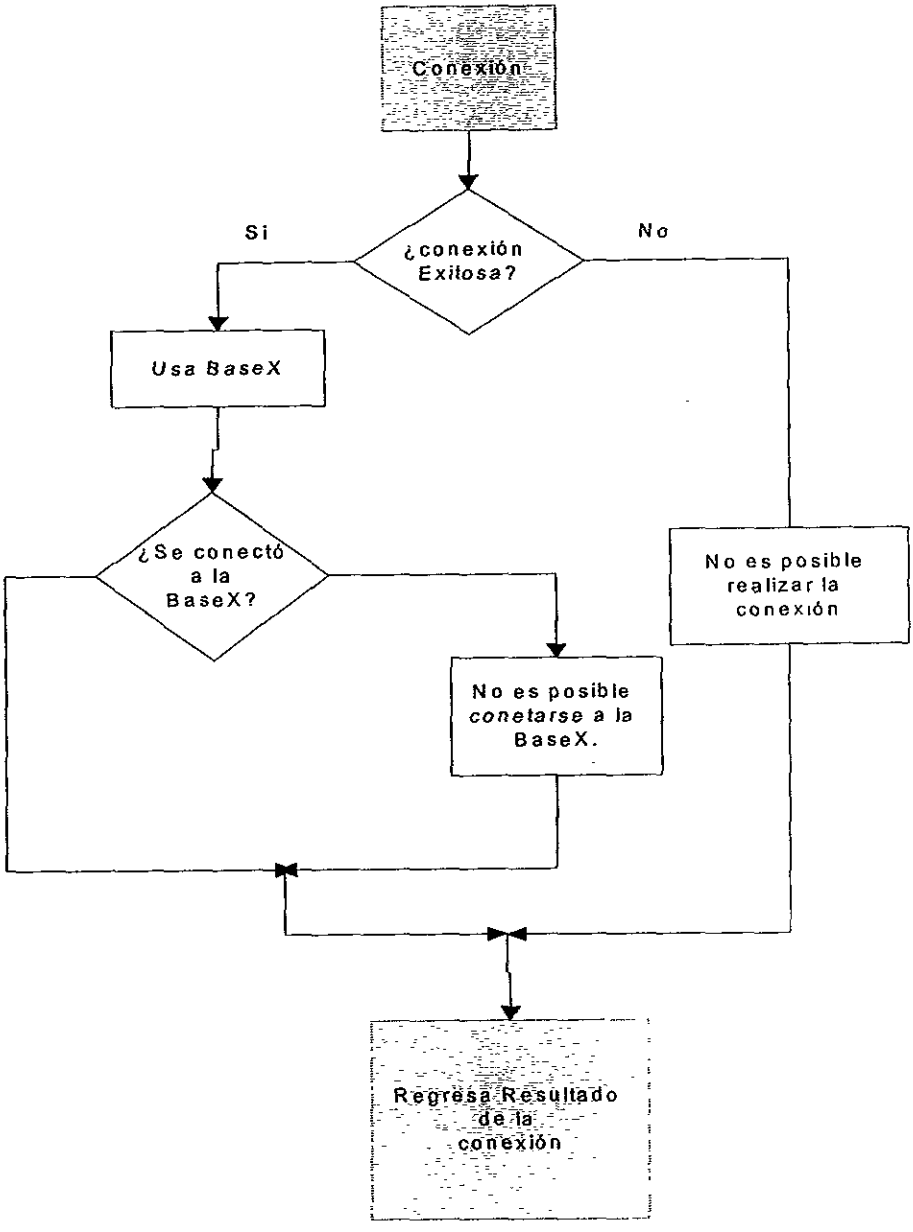
Posteriormente se realizará una interfaz gráfica en la que se pueda introducir la clave del usuario y su password, la cual llevará un proceso en el que verifique que el usuario sea valido y no halla agotado sus conexiones permitidas.

El siguiente paso será realizar la interfaz que muestre al usuario una lista de los Proyectos existentes y al elegir alguno de la lista, le aparezca una ventana que contenga las Aplicaciones a las que tiene acceso, si el usuario no tiene ningún privilegio de acceso para ninguna Aplicación no le aparecerá nada, de esta manera en caso de que no tenga acceso a ninguna Aplicación de dicho Proyecto no quedaran huecos que lo hagan muy evidente, provocando algún disgusto por parte de los usuarios. Aunque cabe resaltar que para los usuarios internos de la institución por lo menos tendrán acceso a las Aplicaciones de consulta para cada proyecto.

El "Sistema Administrador de Aplicaciones" se ejecutarán de manera asíncrona de los procesos de ejecución de las Aplicaciones, de manera que podrán tener ejecutando varias Aplicaciones simultaneas e incluso terminar el "Sistemas Administrador de Aplicaciones" y continuar con la ejecución de sus Aplicaciones.

A continuación se mostrará el diagrama de flujo de datos de la conexión del "Sistema Administrador de Aplicaciones".

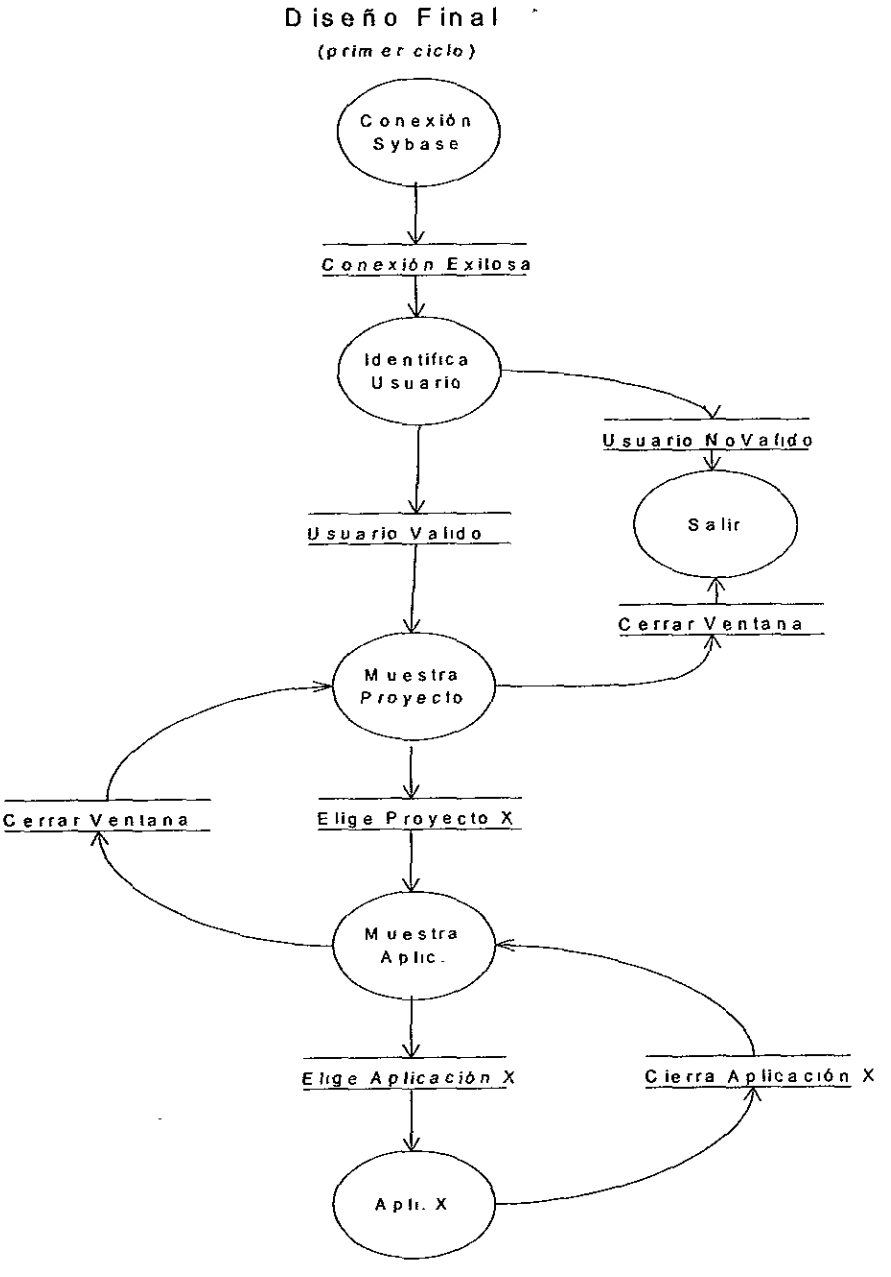
Procedimiento de Conexión (Sybase)



IV.1.2.4 Diseño final

Se mencionaba la importancia de dar más detalle de la información que se mostrará, por lo que en la interfaz gráfica, se desplegara para cada Proyectos, una breve información (nombre completo del proyecto, ya que en general se utiliza el nombre corto, oficina responsable, el tema de origen, etc.) y un contacto para que en caso de que requiera ciertos privilegios sepa con quien puede comunicarse.

A continuación presentaremos un diagrama que ilustra más claramente cual será nuestros estados del "Sistemas Administrador de Aplicaciones".


Diseño Final
(primer ciclo)
Conexión Sybase
Conexión Exitosa
Identifica Usuario
Usuario NoValido
Salir
Usuario Valido
Cerrar Ventana
Muestra Proyecto
Cerrar Ventana
Elige Proyecto X
Muestra Aplic.
Elige Aplicación X
Cierra Aplicación X
Apli. X

IV.1.3 Construcción

Para construir el diseño del "Sistema Administrador de Aplicaciones", se iniciará con una programación estándar por procedimientos en el que se definan las condiciones iniciales del sistema, combinada con la potencia y la flexibilidad propias de la programación orientada a objetos, para la realización de la interfaz gráfica que se definió en el diseño.

Es importante señalar que al iniciar nuestro sistema tenemos que establecer comunicación con el servidor SyBase, en el que se encuentra la Base de Datos; para la cual se utilizará el siguiente código:

```
/* Realiza la conexión al Servidor */
salida = .T.
Store SQLCONNECT('servidorX','usuarioX,'claveX') TO gnConnHandle
/* si no fue exitosa */
IF gnConnHandle <= 0
        /* mensaje de error */
        = MESSAGEBOX('No se pudo hacer la conexion', 16,
'Conexion SQL ')
        /* regresa con falso */
        salida = .F.
/* si fue exitosa */
ELSE
        = MESSAGEBOX('Conexión realizada', 48, 'Conexión SQL')
        /* cambiar a la Base de Datos BaseX */
        xexxec = 'use BaseX'
```

```
/* realiza la instrucción */
xxres=SQLEXEC(gnconnhandle,xexxe c,'result')
/* si no fue exitosa */
IF xxres < 0
                    /* muestra el error */
                    Do ErrorSQL
                    /* cierra tablas de Visual Fox */
                    close data

/* realiza desconecta */
= SQLDISCONNECT(gnConnHandle)
WAIT WINDOW 'Desconectándose de Sybase' TIMEOUT 2
Salida = .F.
/* si fue exitosa */
ELSE
WAIT WINDOW 'Conectándose a  BD  BaseX' TIMEOUT 2

ENDIF
ENDIF
RETURN salida
```

Necesitamos traer información que se encuentra en Sybase, para realizar la identificación del usuario, es decir para saber si nuestros usuarios son o no validos y si aun no han agotado su número autorizado de sesiones en el sistema. - Lo cual se

realizará haciendo uso de los procedimientos almacenados que se pueden definir en el servidor para realizar transacciones sobre las tablas correspondientes, vigilando con ello la integridad de los datos, ya que para realizar ciertas transacciones, directamente sobre las tablas, se tienen que dar privilegios sobre las mismas, y para los procedimientos almacenados sólo se otorgan privilegios de ejecución.

```
/*Busca si el usuario es valido y si aún no agota sus conexiones */


create proc UsuariosDisponibles

@usuario
as

 select * from Usuario where Clave = @Usuario and ConecTotales >
ConecUso
```

También necesitamos traer del servidor la información para llenar nuestras listas de Aplicaciones y Proyectos que se tienen. A continuación se muestra el código de los procedimientos almacenados en el servidor para obtener la lista de Aplicaciones y Proyectos

```
/* Trae información de las Aplicaciones */


create proc TraeAplicacion

@xUsuario char(8) = null

as

select Proyecto=P.Clave, App=A.Descripcion,
ClaveApp=U.ClaveAplicacion,A.Tipo, Rutal=A.RutaInterna,
RutaE=A.RutaExterna, Privilegio = U.Privilegio

 from Proyecto P, Uso U, Aplicacion A

  where U.ClaveUsuario = @xUsuario and
```

```
A.Clave = U.ClaveAplicacion and

P.Clave = U.ClaveProyecto

order by U.ClaveAplicacion

/* Trae información de los proyectos */


create proc ListaProyectosInternos

as

select Clave=P.Clave, Nombre=P.Descripcion,
ClaveT=P.ClaveTema, Tema=T.Descripcion, Referencia="Oficina:
"+P.ClaveOficina+">"+"Responsable: "+ U.Nombre

from P

royecto P, Usuario U, Oficina O, Tema T

where O.Clave = P.ClaveOficina and U.Clave =
O.ClaveResponsable and T.Clave = P.ClaveTema

order by P.Clave
```

Después de tener bien definido el entorno se hará uso de las ventajas que nos proporciona la programación orientada a objetos para la construcción de la interfaz gráfica con el usuario. Para lo cual VisualFox maneja un diseñador de formatos, que es un editor especial que va proporcionando los elementos necesarios para el diseño de interfaces gráficas (las cuales verán los usuarios físicamente como ventanas).

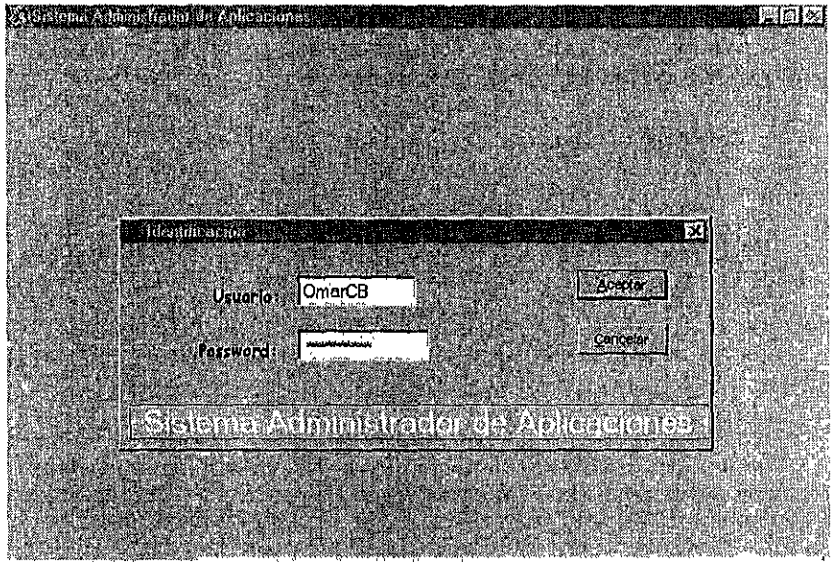
Dentro de los objetos que utilizaremos se encuentran las Page Frame, que son un número de paginas con una pequeña pestaña que sirve para activarla, solo una se encuentra activa a la vez y es la que se muestra en la pantalla. En cada página, se diseñara una ventana con objetos que nos ayuden a realizar una o varias tareas especificas, de esta manera podemos organizar mejor nuestras tareas, para mayor comprensión del usuario y evitando pantallas muy densas.

Existen otros controles como los Label (etiquetas), Text Box (cuadros de texto), Edit Box (cuadros de edición), Command Button (botones de comando), Opcion Group (grupo de opciones), Check Box (botón de checado), Combo Box (cuadro de edición ligado a una lista), List Box (Lista), Grid (estructura de tabla), Image (Imagen), Line (línea) y otros pero éstos son los que utilizaremos con mayor frecuencia; y que se manipulan a través de sus propiedades, eventos y métodos dependiendo de la clase a la que pertenezca cada objeto. Las propiedades pueden establecerse durante el tiempo de diseño o durante el tiempo de ejecución.

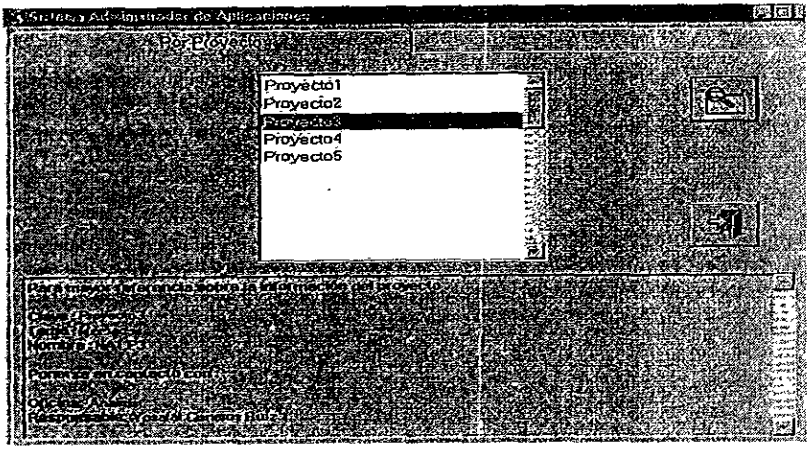
Cada objeto reconoce y puede responder a determinadas acciones denominadas eventos. Un evento es una actividad específica y predeterminada, iniciada por el usuario o por el sistema. Los eventos, en la mayor parte de los casos, se generan por interacción del usuario. Los eventos también se desencadenan cuando el usuario presiona los botones para efectuar una llamada.

Finalmente nuestro Prototipo ha quedado de la siguiente manera:

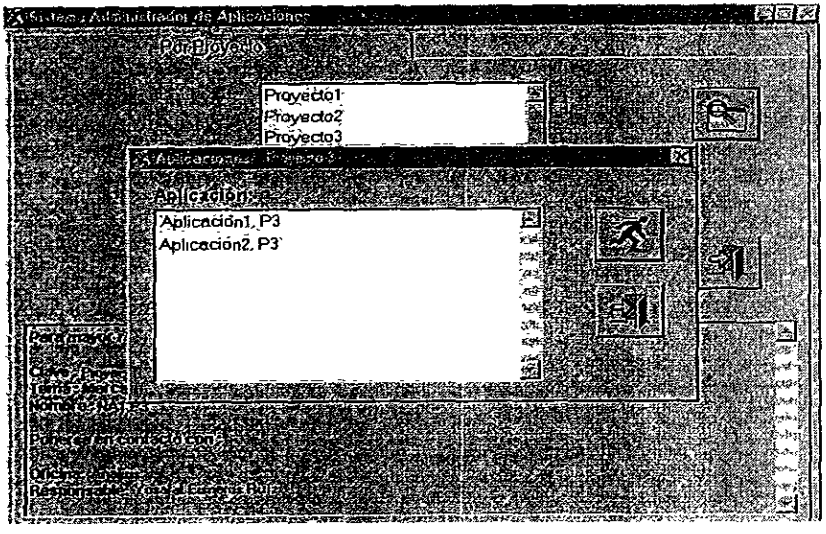
Una interfaz gráfica para la identificación del usuario.



La siguiente ventana es para mostrar la lista de Proyectos que se tienen y su definición. Al elegir uno de los Proyectos en la lista mostrará su respectiva información y al hacer clic en el botón que tiene una lupa mostrará la lista de Aplicaciones. El botón de la puerta es para indicar la salida del sistema.



Como se explicaba anteriormente al elegir el botón de la lupa se sobrepone una ventana de la lista de *Aplicaciones* que contenga cada *Proyecto* y que el usuario tenga derechos para su acceso como se muestra enseguida.



IV.1.4 Evaluación

IV.1.4.1 Análisis de Riesgo

Para la entrega de este primer ciclo se realizaron las pruebas pertinentes de conexión y de buen funcionamiento de todos los procesos utilizados; por lo que podemos apreciar que el desarrollo del sistema planteado en este ciclo fue adecuado.

Alguna recomendación para futuras construcciones es el agregar paginas en el Page Frame, como módulos distintos, de esta manera resultara menos embarazoso tener que corregir la programación ya existente, iniciando nueva programación.

IV.1.4.2 Evaluación del Cliente

Al evaluar el progreso del desarrollo del sistema, se encontró satisfactorio ya que cumplía con los requerimientos del software, del diseño lógico y de los objetivos para esta primera construcción.

Se mostró el resultado de este primer ciclo a los usuarios para su evaluación, y sus observaciones fueron que si cumplía con los requerimientos, con la pequeña observación de que querían también la opción de poder ver los proyectos clasificados por Temas para poder ver la información de manera mas organizada, pensando en las personas que no están familiarizadas con los proyectos, lo cual se desarrollara en el siguiente ciclo.

I.2 Segundo Ciclo

Dadas las recomendaciones de los usuarios en la evaluación del ciclo anterior, se realizará un *módulo que muestre* la organización completa de la información para mayor comprensión de los usuarios.

IV.2.1 Análisis

IV.2.1.1 Requerimientos del Negocio

Se requiere mostrar a los usuarios la organización completa de la información que se tienen *en la institución*, con el fin de que las personas que no tienen conocimiento del tipo de información que se les está mostrando, la puedan comprender *mejor y logren sacar* mayor provecho de ello.

IV.2.1.2 Requerimientos del Sistema

El sistema en este momento tiene todo lo que requiere para su ejecución sólo es cuestión de anexar programación a nuestra aplicación.

IV.2.2 Diseño

IV.2.2.1 Diseño Conceptual

Se necesitará mostrar al usuario la estructura de la información de manera más completa, para su mayor comprensión, esto se refiere a que la información se clasifica en Temas que contienen Proyectos a los cuales se les desarrollan Aplicaciones.

De esta manera los usuarios que estén menos familiarizados con la información puedan enterarse del origen de la información, comprendiendo de manera más clara de que se trata.

IV.2.2.2 Diseño Lógico

Se necesitará un nuevo módulo que proporcione los elementos para mostrar la lista de Temas que se tienen a la cual responda otra lista de Proyectos que se define para cada Tema en particular y de igual manera que en el ciclo anterior de cada Proyecto se puede seleccionar algún Proyectos y mostrar las Aplicaciones a las que tienen acceso el usuario en particular.

IV.2.2.3 Diseño Físico

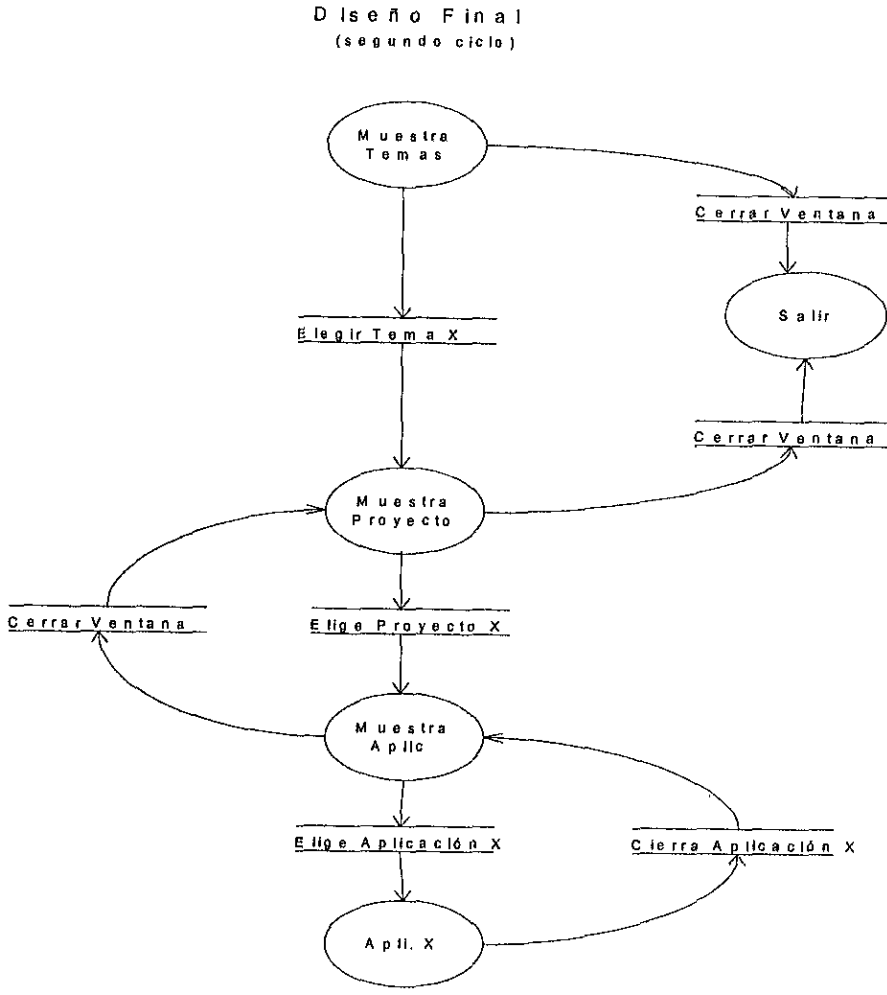
En este momento ya tenemos una interfaz gráfica en la que se representó un esquema básico de la información, ahora tenemos que crear otro módulo en el se represente de manera más completa la estructura de la información contribuyendo con ello a una mejor comprensión.

Se dan a elegir uno de los posibles Temas que en un principio se obtuvieron de nuestras consultas al servidor y del resultado de la elección se construye otra lista en la que aparezcan, sólo los Proyectos que se derivan del Tema seleccionado, esto puede estar en la misma interfaz gráfica ya que de hecho todos los usuarios podrán ver hasta este nivel de la información, y lo siguiente se realizará de manera semejante

que en el ciclo anterior; al elegir un Proyecto se muestre una ventana diferente en la que se muestren las Aplicaciones que tiene dicho proyecto para que puedan ser ejecutadas.

IV.2.2.4 Diseño Final

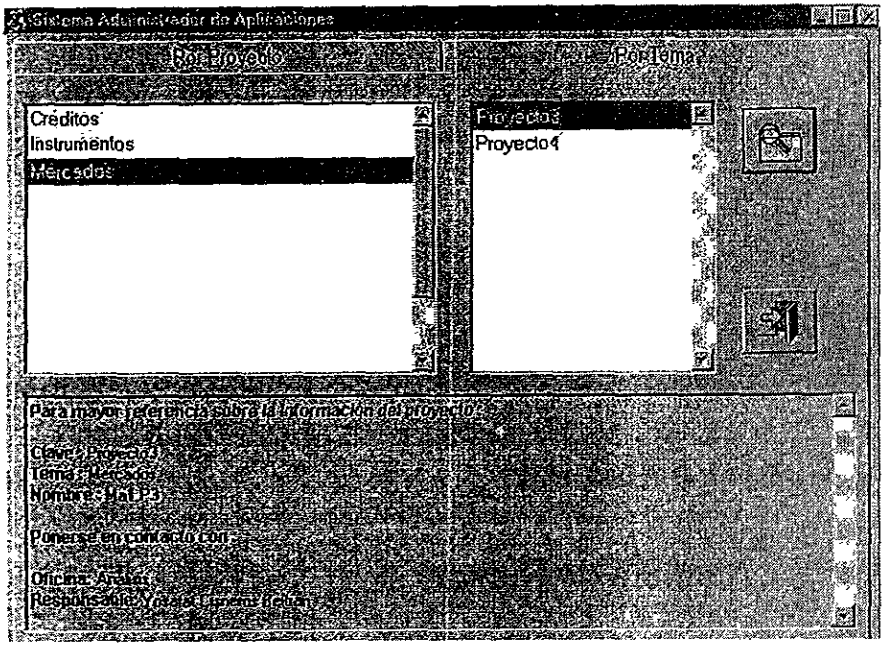
También en este momento falta hacer énfasis en que a la nueva presentación de los datos también conviene agregarle la información complementaria que se tiene de las Aplicaciones.



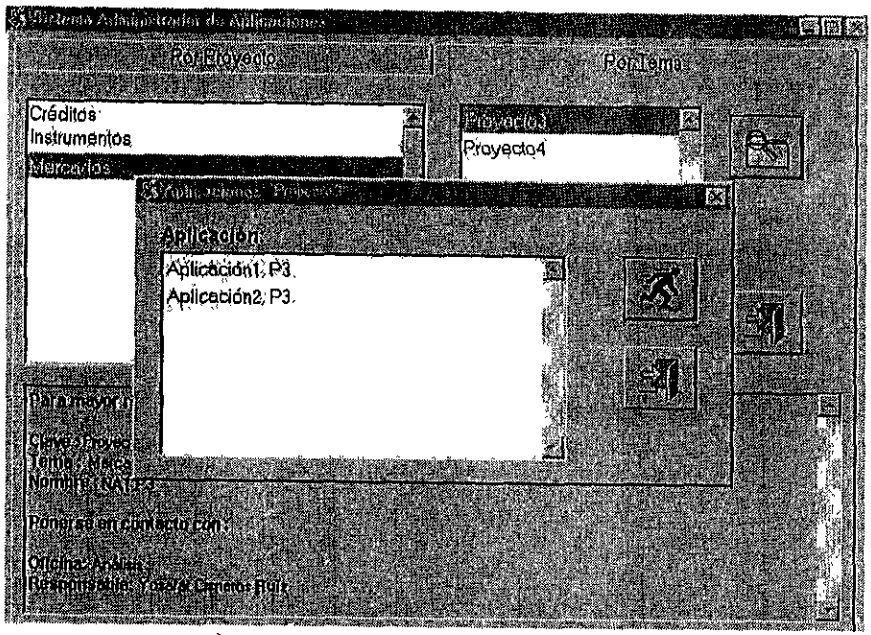
IV.2.3 Construcción

Para la construcción del nuevo ciclo sólo bastará con agregar una página al Page Frame y definir un Combo Box que abra una tabla que se ha bajado

previamente del servidor, la cual contenga la lista de los Temas que se tienen y al *ocurrir este evento refresque* otro Combo Box que abra una tabla que contenga los Proyectos que estén definidos para cada Tema que se elija, como se muestra en la pantalla siguiente:



Y al igual que en el ciclo anterior al elegir un Proyecto se abra una nueva ventana que en el *caso especial de VisualFox* se trata de un *Form* (forma), que muestre otra lista llamada Combo Box para mostrar las Aplicaciones que se refieren al Proyecto que se ha seleccionada y que tiene disponible para su acceso.



IV.2.4 Evaluación

IV.2.4.1 Análisis de Riesgo

Para la entrega del segundo ciclo se realizaron las pruebas pertinentes del buen funcionamiento de todos los procesos utilizados principalmente el de refresco de la lista de Proyectos al cambiar de Tema; por lo que podemos apreciar que el desarrollo del sistema planteado en este ciclo fue adecuado.

IV.1.4.2 Evaluación del Cliente

Se mostró el resultado del segundo ciclo a los Usuarios para su evaluación, y su contestación fue que si cumplía con los requerimientos, con la pequeña observación de que querían también la opción de poder ver los formatos de la información con los que se encontraban más familiarizados y les resultaría de gran

ayuda, y lo cual se realizará en el siguiente ciclo y quedando de acuerdo con los usuarios que posteriormente se liberaría el sistema para producción. Aunque en cualquier momento que surgieran nuevos requerimientos podrían ser realizados siguiendo el método de desarrollo de sistemas del modelo en espiral.

IV.3 Ciclo Final

IV.3.1 Análisis

IV.3.1.1 Requerimientos del Negocio

De acuerdo a las sugerencias de los usuarios hechas en la evaluación del ciclo anterior, se implementara la opción de mostrar los formatos de la información (esto se refiere a la plantilla con la que los usuarios están familiarizados, la cual contiene los títulos de las columnas de los datos con los nombres largos y anotaciones de la información allí contenida, para las columnas que lo requieren y que son los mismos usuarios los que las elaboran), para que no haya lugar a duda de la información de la que se trata, al estar navegando en la estructura de la información.

IV.3.1.2 Requerimientos del Sistema

Para el desarrollo del siguiente ciclo, resulta necesario crear una serie de formatos, que no es más que un diseño de cómo los usuarios colocan la información en hojas para su mejor comprensión, lo cual se puede realizar en Word mandándolo a desplegar desde VisualFox.

IV.3.2 Diseño

IV.3.2.1 Diseño Conceptual

Se desea poner a disposición de los usuarios los formatos de la información con los que interpretan con mayor facilidad a dicha información de manera que no les quede duda de lo que se les esta mostrando.

IV.3.2.2 Diseño Lógico

En este momento identificamos claramente dos procesos separados; por una parte se tienen que generar todos los formatos de la información que se tiene de una manera en que se pueda mostrar en VisualFox y el otro proceso es crear los medios en VisualFox para que el usuario pueda realizar dicha ejecución cuando lo desee.

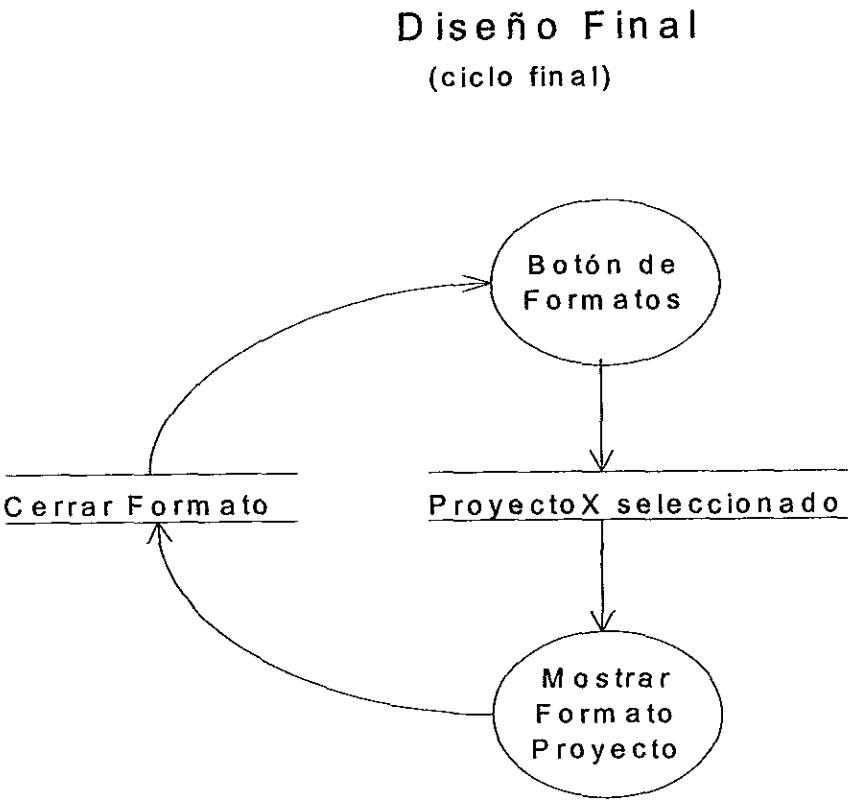
IV.3.2.3 Diseño Físico

Por el lado de crear los formatos contamos con la herramienta que nos proporciona las plantillas y el generador de formatos Help de Word que es el construir formatos que pueden ser desplegados en VisualFox sin la mayor complicación.

Por el lado de VisualFox se agregaría un objeto de tipo Botón, utilizando su evento de la acción del click ejecute el desplegado de los formatos del Proyecto que en el momento se encuentre activado.

IV.3.2.4 Diseño Final

El diseño Final de este tercer ciclo se ilustrará a continuación.



IV.3.3 Construcción

Los archivos de Word se tendrán que construir con una plantilla d2h.wll y posteriormente generar un archivo de tipo Help y este se asignará como un archivo default para cuando se haga la llamada a la ayuda de VisualFox.

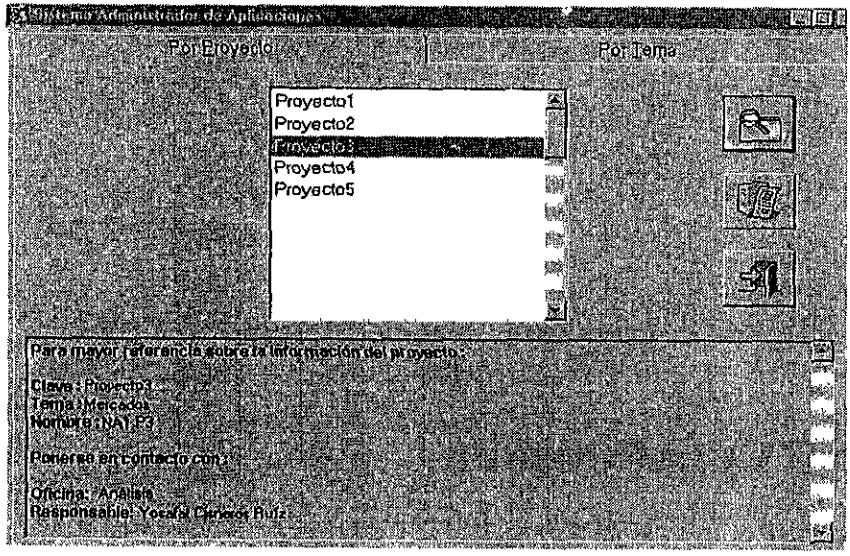
Entonces en las pantallas se agregará el botón que realizará la llamada al despliege del formato.

La instrucción que se usa en VisualFox para realizar la asignación del archivo de ayuda y el despliege que se pondrá en el botón son las siguientes instrucciones:

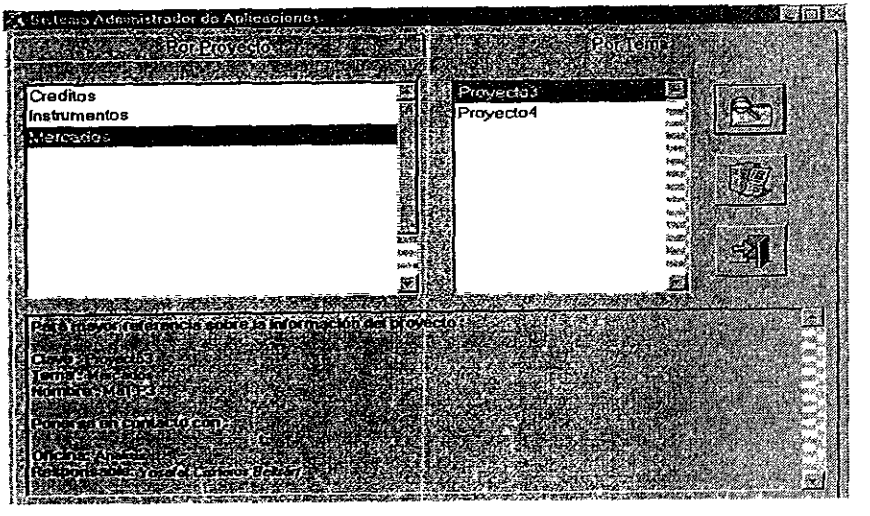
```
SET HELP TO (arch_ayuda)
 HELP
```

Entonces las ventanas anteriormente creadas ahora se verán de la siguiente manera:

La ventana de Por Proyectos:



La ventana de Por Tema



IV.3.4 Evaluación

IV.3.4.1 Análisis de Riesgo

Para la entrega del sistema se realizaron pruebas exhaustivas de todos y cada uno de los módulos y procedimientos del sistema, para reducir al mínimo la *posibilidad de que* fallara una vez entregado al usuario.

El sistema se desarrolló conforme a los planes iniciales, dando como resultado un sistema que estaba listo para ser liberado a producción.

IV.1.4.2 Evaluación del Cliente

En una demostración que se les hizo a los usuarios bajo las pruebas sugeridas por los mismos, se liberó el sistema con su puesta en marcha, además se les entregó documentación necesaria para mayor comprensión del funcionamiento del "Sistema Administrador de Aplicaciones".

Los resultados del impacto del "Sistema Administrador de Aplicaciones" en los usuarios fueron los esperados. Y esto se midió en el número de accesos que los usuarios realizan al sistema y a la gran cantidad de solicitudes de privilegios para accesos a las diversas Aplicaciones.

Capítulo V

Conclusiones

Conclusiones

Al llegar al término de este trabajo, contamos con los elementos necesarios para valorar la importancia y el impacto que la información, tiene en la estructura organizacional de una empresa.

Al trabajar con el "Sistema Administrador de Aplicaciones", se atendió a las necesidades de información (que son de vital importancia) que los usuarios tenían. Proporcionando a los usuarios de una herramienta útil, para un pronto, completo y confiable análisis de los datos.

Evitar la duplicidad de la información, resultó una tarea sumamente productiva para la empresa, ya que esto representó un gran ahorro de recursos, por la gran dimensión de información con la que se cuenta. Dicho ahorro se logró gracias al "Sistema Administrador de Aplicaciones", ya que al reunir un grupo de Aplicaciones que proporcionaban información a los usuarios, se evito el tener información repetida para cada oficina, además se logró tener fuentes de datos únicas para evitar discrepancia.

Aunque el objetivo fundamental de inicio, era el establecer un mecanismo de acceso controlado a los diversos sistemas, con el fin de resguardar la integridad de la información; también se pretendía de tras

fondo el desarrollo de un sistema cuya presentación realizara la difusión de la información a todas las áreas, haciendo ciertas consideraciones. Se consiguió con gran éxito los objetivos planteados, más aún también se logró una mayor transparencia de muchos de los procesos que en la institución se realizan.

La aplicación del concepto de *metainformación*, que se basa en la descripción de la propia información (del cómo esta organizada la información, de manera estructural). Esto proporcionó en gran media, mayor orden de la información que se tiene, para poder darla a conocer a los usuarios y permitirles el acceso.

A pesar de contar con Aplicaciones desarrolladas en plataformas diferentes (situación que es común encontrar en las organizaciones debido al paso del tiempo, evolución de las tecnologías de información, y nuevos requerimientos de la propia empresa), se logró compaginar a través de una sola puerta el acceso a las mismas, así como su catalogación. Esto da a los usuarios una idea más clara de la información disponible en cada una de las Aplicaciones y en caso de haber algún interés por alguna aplicación en particular, se tienen a la mano los datos necesarios para ponerse en contacto con el responsable, aclarar sus dudas con respecto a la información presentada y solicitar su acceso o modificaciones a sus privilegios actuales.

Podemos decir que el éxito del "Sistema Administrador de Aplicaciones" se debe en gran medida a la metodología en espiral, que condujo al buen desarrollo del sistema de manera muy natural al origen del problema.

Se administraron exitosamente los recursos tanto técnicos y humanos, por lo que se resalta la gran importancia que tiene una relación en armonía entre las personas, las computadoras y la información.

Toda la carga de trabajo de la administración (control sobre quién puede o no accesar alguna información, incluir o excluir usuarios, aplicaciones, temas etc.) de la Base de Datos del "Sistema Administrador de Aplicaciones", se ve disminuida con el "Sistema Administrador de la Base de Datos", que brinda todas las facilidades para realizar de manera eficiente cualquier ajuste solicitado por los usuarios, o por motivos de mantenimiento, y que puede realizar cualquier usuario que conozca la institución.

De manera integral, el planteamiento hecho en los capítulos anteriores permitirá sentar las bases sobre la conveniencia de ocupar un diseño del tipo Cliente/Servidor, para el traslado de la mayoría de las Aplicaciones o el desarrollo de nuevos sistemas. Las ventajas de utilizar este enfoque ya se han tratado, sin embargo, cabe recordar que la importancia de trabajar con una plataforma de este tipo, radica en compartir información, situación que cada vez es más frecuente en las empresas.

La inclusión de una Aplicación en el "Sistema Administrador de Aplicaciones" se irá dando de manera muy natural. La posibilidad de incluir Aplicaciones muy dedicadas que no cumplan al 100% con el esquema planteado, se tendrían que evaluar para establecer su factiblilidad de inclusión

Aún, cuando este sistema trata de dar a conocer gran cantidad de información y evitar su duplicidad en varias áreas, existen diversas necesidades por parte de algunos usuarios, que le dieron un cierto grado de complejidad a la Aplicación, que pudo ser mucho más abierta. Ejemplo de tal situación, fue el hecho de manejar consideraciones especiales sobre privilegios de acceso a Aplicaciones que en su mayoría, tenían reportes restringidos o funciones muy particulares.

De consideraciones como la anterior, se retroalimentaron los requerimientos y el diseño original del sistema, lo cual en realidad no impacto de manera significativa debido al uso de la metodología del Modelo Espiral, la cuál permitió llevar a cabo una revisión detallada de las necesidades de los usuarios involucrados y cubrir tales aspectos en el diseño. Aunque la idea principal fue globalizar y tratar de prever muchos conceptos y procesos, se debieron hacer algunas consideraciones específicas para el

correcto funcionamiento de las Aplicaciones. Todo esto con miras a transformarse con una estructura más general.

Los elementos teóricos y prácticos adquiridos en la licenciatura fueron indispensables para analizar, diseñar, programar, y probar el "Sistema Administrador de Aplicaciones", cabe destacar algunas áreas de estudio relacionadas con el desarrollo de sistemas, conceptos de los campos de: Ingeniería de Sistemas, Estructura de Datos, Programación Estructurada, Organización de Centros de Cómputo, Organización de Computadoras, y Administración de Centros de Computo.

El conjunto de herramientas para el desarrollo del proyecto fue establecido por los lineamientos establecidos de la empresa. Por lo cual, su planeación se apegó a los recursos disponibles y a los que la empresa considera como los más adecuados para el trabajo informático.

Para la consumación de un producto final de calidad, también se contempló la capacitación e investigación en las herramientas usadas en el desarrollo del proyecto, como los lenguajes de programación en VisualFox y sentencias SQL empleadas en la interacción con Bases de datos en servidores Sybase y utilerías.

Prácticamente es una herramienta que utilizan usuarios de todos los niveles, para accesar en forma centralizada la información que no necesariamente esta concentrada en un solo lugar. De este modo se cumple con los objetivos de difundir la información en forma global y además de centralizar el control y mantenimiento de acceso a las Aplicaciones.

La satisfacción del usuario se logró alcanzar debido fundamentalmente a la metodología de desarrollo empleada, que permite una constante interacción con el usuario, ya que participan prácticamente en todas las etapas, dando oportunamente sus comentarios, ayudando a reajustar a tiempo aspectos que en un inicio no se aprecian. Otro factor de gran influencia, para la mejor aceptación del usuario es el manejo del aspecto más visual y amigable del Cliente que permite al usuario interactuar con el sistema muy cómodamente. También se aprovecharon las ventajas

que el servidor ofrecía, como son: el esquema *multiusuario*, y su velocidad de procesamiento, entre otras.

La realización de un trabajo como el presentado, aporta un gran cúmulo de experiencia. El enfrentamiento con la realidad de conceptualizar e implementar un sistema computacional permite apreciar un conocimiento que la simple teoría no siempre nos puede mostrar. Uno se da cuenta que pueden existir muchos factores de influencia en una problemática y no hay que descartar ninguno por más controlado que pueda parecer. Las soluciones planteadas pueden ser triviales o muy complicadas pero su valía radica en su efectividad.

Finalmente se logró de manera exitosa los objetivos iniciales que fueron:

- La Administración de Aplicaciones ya existentes.
- La difusión de la información de manera rápida y eficiente.
- Mantener la integridad de la información.
- Reducir redundancia en la información.

El sistema cumple con los requerimientos básicos requeridos, lo que quiere decir que es factible a mejoras, como podría ser que en el "Sistema Administrador de la Base de Datos" presentara más facilidades al usuario tales como que al editar un campo de la Base de Datos pudiera desplegar una lista de posibles valores validos, como son los casos las transacciones para los privilegios, en donde sabes que debe ir un usuario existente, una aplicación existente, un proyecto existente y días, horas, y privilegios definidos.

El sistema nos proporcionó gran satisfacción, ya que el hecho de realizar una herramienta de gran utilidad para los usuarios, es gratificante. Actualmente el sistema esta en producción con gran aceptación entre los usuarios y dejando una gran influencia en la institución de un nuevo concepto en el manejo de la información.

ANEXO A

AltaAplicacion

```
Create procedure AltaAplicacion
        @cve_apl        varchar(15)     = null,
        @nom_apl        varchar(80)     = null,
        @tip_apl        char(1)         = null,
        @rut_int        varchar(50)     = null,
        @rut_ext        varchar(50)     = null,
        @u               tinyint        =null,
        @cve_ofc        cha r(3)        = null

as
 if @cve_apl is null or @nom_apl is null or @tip_apl is null or @rut_int is null or
@rut_ext is null or @cve_ofc is null
  print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select *       from    Aplicacion    where    ClaveAplicacion    =
upper(@cve_apl))
    print "@@@ Clave de aplicación ya existe @@@"
  else
   if not exists (select * from Oficina where ClaveOficina = upper(@cve_ofc))
    print "@@@ Clave de oficina debe existir en tabla Oficina @@@"

   else
        begin
          insert into Aplicacion (ClaveAplicacion, Descripcion, Tipo, RutaInterna,
RutaExterna, Uso, ClaveOficina)
                values (upper(@cve_apl), @nom_apl, @tip_apl, @rut_int,
        @rut_ext, @u, @cve_ofc)
         print "Alta exitosa"
        end

(return status = 0)
```

AltaOficina

```
Create procedure AltaOficina
        @cve_ofc        char(3) = null,
```

```
        @nom_ofc        varchar(100)     = null,
        @cve_rsp        char(6)  = null
as
 if  @cve_ofc is null or @nom_ofc is null or @cve_rsp is null
  print "@@@ Alguno de  los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Oficina where ClaveOficina = upper(@cve_ofc))
   print "@@@ Clave de oficina ya existe @@@"
  else
   if exists (select * from Usuario where ClaveUsuario = upper(@cve_rs p))
    begin
         insert into Oficina (ClaveOficina, Descripcion, ClaveResponsable)
                  values  (upper(@cve_ofc), @nom_ofc, @cve_rsp)
     print "Alta exitosa"
    end
   else
    print "@@@ El responsable no existe en la tabla de usuarios @@@"


(return status = 0)
```

AltaProyecto

```
Create procedure AltaProyecto
        @cve_pro        varchar(15)      = null,
        @nom_pro        varchar(100)     = null,
        @cve_tem        tinyint   = null,
        @cve_ofc        char(3)  = null
as
 if @cve_pro is null or @nom_pro is null or @cve_ ofc is null or @cve_tem is null
  print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Proyecto where ClaveProyecto = upper(@cve_pro))
   print "@@@ Clave de proyecto ya existe @@@"
  else

   if not exists (select * from Oficina where ClaveOficina = upper(@cve_ofc))
    print "@@@ Clave de oficina debe existir en tabla Oficina @@@"
   else
    if not exists (select * from Tema where ClaveTema = @cve_tem)
```

```
      print "@@@ Clave de Tema  debe existir en tabla Tema @@@"
        else
         begin
              insert into Proyecto (ClaveProyecto, Descripcion, ClaveTema,
ClaveOficina)
                          values    (upper(@cve_pro), @nom_pro, @cve_tem,
@cve_ofc)
          print "Alta exitosa"
          end

(return status = 0)
```

AltaTema

```
Create procedure AltaTema
        @cve_tem        tinyint(1)        = null,
        @nom_tem        varchar(100)      = null

as
 if @cve_tem is null or @nom_tem is null
  print "@@@ Alguno de  los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Tema where ClaveTema = upper(@cve_tem))
   print "@@@ Clave de tema ya existe @@@"
  else
     begin
         insert into Tema (ClaveTema, Descripcion)
                    values  (upper(@cve_tem), @nom_tem)
       print "Alta exitosa"
     end


(return status = 0)
```

AltaPrivilegio

```
Create procedure AltaPrivilegio
        @cve_usu        varchar(8)        = null,
        @cve_apl        varchar(15)       = null,
```

```
        @cve_pro        varchar(15)     = null,
        @privi          varchar(6)      = null
as
 if @cve_usu is null or @cve_apl is null or @cve_pro is null or @privi is null
  print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
 else
    if exists(select * from Privilegio where ClaveUsuario=upper(@cve_usu) and
ClaveAplicacion=upper(@cve_apl) and ClaveProyecto=upper(@cve_pro))
     print "@@@ Ya existe Privilegio para el Usuario en la Aplicación del
Proyecto @@@"
    else
       if not exists (select * from Proyecto where ClaveProyecto =
upper(@cve_pro))
          print "@@@ Clave de proyecto debe  existir en la tabla Proyecto
@@@"
       else
          if not exists (select * from Aplicacion where ClaveAplicacion =
upper(@cve_apl))
             print "@@@ Clave de oficina debe existir en tabla Oficina
       @@@"
          else
             if not exists (select * from Usuario where ClaveUsuario=
@cve_pro)
                print "@@@ Clave de Usuario  debe existir en tabla
       Usuario @@@"
       else
                begin
                   insert into Privilegio (ClaveUsuario, ClaveAplicacion,
ClaveProyecto, Privilegio)
                values  (upper(@cve_usu), @cve_apl, @cve_pro, @privi)
         print "Alta exitosa"
         end

(return status = 0)
```

BajaAplicacion

Create procedure BajaAplicacion

```
        @cve_apl        varchar(15)     = null,
        @nom_apl        varchar(80)     = null,
        @tip_apl        char(1)         = null,
        @rut_int        varchar(50)     = null,
        @rut_ext        varchar(50)     = null,
        @u               tinyint        =null,
        @cve_ofc        cha r(3)        = null

as
 if @cve_apl is null or @nom_apl is null or @tip_apl is null or @rut_int is null or
@rut_ext is null or @cve_ofc is null
   print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
 else
   if exists (select * from Aplicacion where ClaveAplicacion = upper(@cve_apl))
      begin
            delete from  Aplicacion where ClaveAplicacion = upper(@cve_apl)
            print "Baja exitosa"
      end
   else
     print "@@@ Clave de aplicación no existe @@@"

 (return status = 0)
```

BajaOficina

```
Create procedure BajaOficina
        @cve_ofc        char(3)  = null,
        @nom_ofc        varchar(100)     = null,
        @cve_rsp        char(6)  = null
as
 if @cve_ofc is null or @nom_ofc is null or @cve_rsp is null
   print "@@@ Alguno de  los parámetros requeridos no fue suministrado @@@"
 else
   if exists (select * from Oficina where ClaveOficina = upper(@cve_ofc))
      begin
          delete from Oficina where ClaveOficina = upper(@cve_ofc)
          print "Baja exitosa"
      end
     else
          print "@@@ Clave de oficina no existe @@@"
```

```
(return status = 0)
```

BajaProyecto

```
Create procedure BajaProyecto
        @cve_pro        varchar(15)     = null,
        @nom_pro        varchar(100)    = null,
        @cve_tem        tinyint   = null,
        @cve_ofc        char(3)  = null
as
 if @cve_pro is null or @nom_pro is null or @cve_ ofc is null or @cve_tem is null
  print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Proyecto where ClaveProyecto = upper(@cve_pro))
   begin
        delete from Proyecto where ClaveProyecto = upper(@cve_pro)
        print "Baja Exitosa "
    end
  else
        print "@@@ Clave de proyecto no existe @@@"

(return status = 0)
```

BajaTema

```
Create procedure BajaTema
        @cve_tem        tinyint(1)      = null,
        @nom_tem        varchar(100)    = null

as
 if @cve_tem is null or @nom_tem is null
  print "@@@ Alguno de  los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Tema where ClaveTema = upper(@cve_tem))
      begin
        delete from Tema where ClaveTema = upper(@cve_tem)
        print  "Baja Exitosa"
      end
  else
```

```
        print "@@@ Clave de tema no existe @@@"

(return status = 0)
```

BajaPrivilegio

```
Create procedure BajaPrivilegio
        @cve_usu        char(8)         = null,
        @cve_apl        varchar(15)     = null,
        @cve_pro        varchar(15)     = null,
        @privi          varchar(6)      = null
as
if @cve_usu is null or @cve_apl is null or @cve_pro is null or  @privi is null
        print "@@@ Alguno de  los parámetros requeridos no fue suministrado
@@@"
 else
 if exists (select * from Privilegio where     ClaveUsuario = upper(@cve_usu)
        and
                                               ClaveAplicacion = upper(@cve_apl)
        and
                                               ClaveProyecto = upper(@cve_pro)
                )
   begin
         delete from Privilegio  where         ClaveUsuario = upper(@cve_usu)
        and
                                               Cl aveAplicacion = upper(@cve_apl)
        and
                                               ClaveProyecto = upper(@cve_pro)

         print "Baja exitosa"
   end
 else
        print "@@@ Clave de Uusuario en la Aplicación del Proyecto no existe
@@@"


(return status = 0)
```

BajaUsuario

```
Create procedure BajaUsuario
        @cve_usu        varchar(8)      = null,
        @nom_usu        varchar(50)     = null,
        @pass_usu       varchar(10)     =null,
        @con_tot        tinyint(1)      =null,
        @con_uso        tinyint(1)      =null,
        @dia_val        char(7)         =null,
        @hor_val        char(4)         =null,
        @mul            tinyint(1)      =null,
        @ext            tinyint(1)      =null

as
  if @cve_usu is null or @nom_usu is null or @pass_usu is null or @con_tot is null
or @con_uso is null or @dia_val is null or @hor_val is null or @mul  is null or
@ext is null
    print "@@@ Alguno de  los parámetros requeridos no fue suministrado @@@"
  else
    if exists (select * from Usuario where ClaveUsuario = upper(@cve_usu))
     begin
          delete from Usuario where ClaveUsuario = upper(@cve_usu)
          print "Baja exitosa"
    end
else
          print "@@@ Clave de usuario no existe @@@"

(return status = 0)
```

CambioAplicacion

```
Create procedure CambioAplicacion
        @cve_apl        varchar(15)     = null,
        @nom_apl        varchar(80)     = null,
        @tip_apl        char(1)         = null,
        @rut_int        varchar(50)     = null,
        @rut_ext        varchar(50)     = null,
```

```
        @u              tinyint         =null,
        @cve_ofc        cha r(3)        = null

as
 if @cve_apl is null or @nom_apl is null or @tip_apl is null or @rut_int is null or
@rut_ext is null or @cve_ofc is null
  print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Aplicacion where ClaveAplicacion = upper(@cve_apl))
        begin
                update Aplicacion set Descripcion=@nom_apl,
                Tipo=@tip_apl, RutaInterna=@rut_int, RutaExterna=
                @rut_ext, Uso=@u, ClaveOficina=@cve_ofc where
                ClaveAplicacion= upper(@cve_apl)
                print "Cambio exitoso"
        end
  else
   print "@@@ Clave de aplicación no existe @@@"

 (return status = 0)
```

CambioOficina

```
Create procedure CambioOficina
        @cve_ofc        char(3)  = null,
        @nom_ofc        varchar(100)     = null,
        @cve_rsp        char(6)  = null
as
 if  @cve_ofc is null or @nom_ofc is null or @cve_rsp is null
  print "@@@ Alguno de  los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Oficina where ClaveOficina = upper(@cve_ofc))
    begin
        update Oficina set Descripcion= @nom_ofc,
        ClaveResponsable=@cve_rsp  where ClaveOficina= upper(@cve_ofc
        print "Cambio exitoso"
    end
   else
        print "@@@ Clave de oficina no existe @@@"
```

```
(return status = 0)
```

CambioProyecto

```
Create procedure CambioProyecto
        @cve_pro        varchar(15)     = null,
        @nom_pro        varchar(100)    = null,
        @cve_tem        tinyint   = null,
        @cve_ofc        char(3)  = null
as
 if @cve_pro is null or @nom_pro is null or @cve_ ofc is null or @cve_tem is null
  print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Proyecto where ClaveProyecto = upper(@cve_pro))
        update  Proyecto set  Descripcion=@nom_pro, ClaveTema=@cve_tem,
ClaveOficina=@cve_ofc
          where ClaveProyecto= upper(@cve_pro)
        print "Cambio exitoso "
  else
         print "@@@ Clave de proyecto no existe @@@"

(return status = 0)
```

CambioTema

```
Create procedure CambioTema
        @cve_tem        tinyint(1)      = null,
        @nom_tem        varchar(100)    = null

as
 if @cve_tem is null or @nom_tem is null
  print "@@@ Alguno de los parámetros requeridos no fue suministrado @@@"
 else
  if exists (select * from Tema where ClaveTema = upper(@cve_tem))
        update  Tema  set Descripcion= @nom_tem
            where ClaveTema= upper(@cve_tem)
        print  "Cambio exitoso"
  else
        print "@@@ Clave de tema no existe @@@"
```

(return status = 0)

CambioPrivilegio

```
Create procedure CambioPrivilegio
        @cve_usu        char(8)         = null,
        @cve_apl        varchar(15)     = null,
        @cve_pro        varchar(15)     = null,
        @privi          varchar(6)      = null
as
if @cve_usu is null or @cve_apl is null or @cve_pro is null or  @privi is null
        print "@@@ Alguno de  los parámetros requeridos no fue suministrado
@@@"
 else
 if exists (select * from Privilegio where     ClaveUsuario = upper(@cve_usu)
        and
                                               ClaveAplicacion = upper(@cve_apl)
        and
                                               ClaveProyecto = upper(@cve_pro)
                )
   begin
         update  Privilegio  set  Privilegio= @privi   where   ClaveUsuario =
upper(@cve_usu) and
                                               ClaveAplicacion = upper(@cve_apl)
        and
                                               ClaveProyecto = upper(@cve_pro)

        print "Cambio exitoso"
   end
 else
        print "@@@ Clave de Uusuario en la Aplicación del Proyecto no existe
@@@"


(return status = 0)
```

CambioUsuario

```
Create procedure CambioUsuario
        @cve_usu        varchar(8)      = null,
        @nom_usu        varchar(50)     = null,
        @pass_usu       varchar(10)     =null,
        @con_tot        tinyint(1)      =null,
        @con_uso        tinyint(1)      =null,
        @dia_val        char(7)         =null,
        @hor_val        char(4)         =null,
        @mul            tinyint(1)      =null,
        @ext            tinyint(1)      =null

as
  if @cve_usu is null or @nom_usu is null or @pass_usu is null or @con_tot is null
or @con_uso is null or @dia_val is null or @hor_val is null or @mul  is null or
@ext is null
    print "@@@ Alguno de  los parámetros requeridos no fue suministrado @@@"
  else
    if exists (select * from Usuario where ClaveUsuario = upper(@cve_usu))
          update  Usuario  set     Nombre=  @nom_usu,Password=@paas_usu,
        ConecTotales=@con_tot, ConecUso=@con_uso, DiasValidos=@dia_val,
        HoraValida=@hor_val,  Multi=  @mul,  Externo=@ext          where
        ClaveUsuario=upper(@cve_usu)
        print "Cambio exitoso"
else
        print "@@@ Clave de usuario ya existe @@@"

(return status = 0)
```

ConsulPrivilegio

```
Create procedure  ConsulPrrivilegio
        @cve_usu        char(8)         = null,
        @cve_apl        varchar(15)     = null,
        @cve_pro        varchar(15)     = null,
as
```

```
select * from Privilegio where ClaveUsuario=@cve_usu and
ClaveAplicacion=@cve_apl and ClaveProyecto=@cve_pro

(return status=0)
```

ConsulBitacora

```
Create procedure ConsulBitacora
        @cve_usu        char(8)         = null,
        @cve_apl        varchar(15)     = null,
        @dia_ini        datetime        = null,
        @dia_fin        datetime        =null
as

select * from Bitacora where ClaveUsuario=@cve_usu and
ClaveAplicacion=@cve_apl and
Dia >= @dia_ini and Dia<=@dia_fin

(return status=0)
```

ConsulProyecto

```
Create procedure ConsulProyecto
        @cve_pro        varchar(15)     = null,
        @cve_tem        tinyint         = null,
        @cve_ofc        char(3)         = null
as

select * from Proyecto where ClaveProyecto=@cve_pro and
ClaveTema=@cve_tem and ClaveOficina=@cve_ofc

(return status=0)
```

Bibliografía

BIBLIOGRAFÍA

- Pressman, Roger
  Ingeniería de Software. Un enfoque práctico.
  3a. ed., México, McGraw Hill/Interamericana de España, 1993.

- Szymansky, Robert A.; Szymansky, Donald P.; Morris, Norma A.;Plushen, Donna M.
  Introduction to Computers and Information Systems.
  2a. ed., U.S.A., McMillan Publishing, 1991

- Kendall, Kenneth E.; Kendall, Julie E.
  Análisis y Diseño de Sistemas.
  1a. ed., España, Prentice Hall Internacional, 1977.

- Microsoft
  Manual del Programador, Visual FoxPro
  Versión 3.0

- SYBASE
  Relational Databse Design
  Triple O Computación, México, D.F., 1990.

- SYBASE
  Introduction to SYBASE Client/Server architecture,
  Version 1.0
  Triple O Computación, México, D.F., 1995.

- SYBASE
  System 10 Performance & Tuning for Developers.
  Version 2.2
  Códice, México, D.F., 1995.