UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

CONSEJO. Base de datos con el uso de hipertexto para el manejo de las actas de las reuniones celebradas por el Consejo Técnico de la Facultad de Ingeniería.

TESIS PROFESIONAL

Que para obtener el Titulo de

INGENIERO EN COMPUTACIÓN

presenta:

Sergio Hernández Martínez

Director de Tesis: Ing. Víctor D. Pinilla Morán

México, D.F. 1996

RECONOCIMIENTOS

A mis queridos padres, Oscar y Herminia, que gracias a su desinteresado apoyo durante toda mi carrera, hicieron posible que llevara a cabo la presente tesis.

A mi familia que en todo tiempo me apoyo.

A mis compañeros de trabajo por la ayuda que me prestaron durante la realización de esta tesis.

A Marcela por el animo que me infundio para realizar este trabajo.

RESUMEN

La necesidad de constantes accesos a la información de los archivos del Consejo Técnico de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, para el desarrollo de sus funciones; la forma en que ésta es procesada actualmente; por sus características y por no existir ningún sistema computarizado que la maneje, son los motivos de el desarrollo de esta tesis.

CONSEJO es el nombre que se le dio al sistema de base de datos. La información de esta base se obtiene de las actas aprobadas de las sesiones del Consejo, además de los documentos relacionados con éstas. Este sistema de información se realizó usando el hipertexo, en una plataforma Windows[1].

CONSEJO da una *interface*[2] amigable con el usuario final. Logra que la información esté disponible para un mayor número de personas y finalmente que el trabajo de consultas y toma de decisiones sea ágil y práctico.

---

[1] Ver Glosario.

[2] Palabra Inglesa, cuya traducción al Español es superficie de separación entre caras. La connotación que se le da en el ambiente de computación se ve en el Glosario.

TABLA DE CONTENIDOS

CAPITULO I.
INTRODUCCIÓN

I.1 MOTIVACIÓN.

El Consejo Técnico, según el Estatuto General de la Universidad Nacional Autónoma de México en su artículo 45, se define como el órgano de consulta necesaria que tendrán facultades y escuelas de la Universidad. "Los representantes profesores, propietario y suplente, serán designados en elección directa, mediante voto universal, libre y secreto..." (Art. 46). Además el artículo 47 del mismo Estatuto dice "Los alumnos designaran dos representantes propietarios y sus suplentes...". A su vez, el artículo 48, también del Estatuto General, designa al director del plantel como presidente del Consejo.

El Artículo 49 del citado Estatuto, establece como obligaciones del Consejo Técnico:

I. Estudiar y dictaminar los proyectos o iniciativas que les presenten el Rector, el director, los profesores y los alumnos o que surjan de su seno;

II. Formular los proyectos de reglamento de la facultad o escuela...

III. Estudiar los planes y programas de estudio...

IV. Aprobar o impugnar las ternas para director del plantel...

V. Hacer observaciones a las resoluciones del Consejo Universitario o del Rector que tengan carácter técnico o legislativo y afecten a la facultad y escuela...

VI. Dictaminar sobre el nombramiento de profesores extraordinarios, elaborar los reglamentos especiales complementarios del Estatuto del Personal Académico y ejercer las facultades que éste les confiere.

Las anteriores actividades generan una cantidad importante de información, la cual queda oficialmente registrada en el acta de la sesión. Este texto hace referencia a una serie de documentos los cuales la mayoría de veces sólo son nombrados en el acta, pero que existen y son muy importantes. Entre éstos podemos mencionar los siguientes: cartas, oficios, reglamentos, acuerdos, recomendaciones, calendarios, convocatorias, tablas, informes, dictámenes. A su vez toda esta información sirve en ocasiones como base de posteriores tomas de decisión, por lo que es constantemente consultada.

Como se planteó anteriormente los principales factores que motivaron el desarrollo de un sistema de información fueron: la forma en que la información es procesada actualmente; el arduo trabajo que representan constantes accesos a ésta, los que deben ser rápidos y en cualquier

momento; por las características de la información y por no existir ningún sistema computarizado que la maneje.

I.2 PROBLEMAS.

El hecho fundamental que hizo indispensable el desarrollo del sistema de información automatizado que nos ocupa, es la manera en que información de gran relevancia e importancia, como la del Consejo Técnico, es almacenada y recuperada, entre los principales inconvenientes de estos procesos podemos mencionar:

- Tanto el almacenamiento de los datos como su recuperación se hacen en forma manual, con la consabida perdida de tiempo y vulnerabilidad que representa depender de alguien que no siempre esté disponible.

- El uso de la información requiere que sea recuperada en lapsos cortos, ya que decisiones de la misma naturaleza a los datos deben ser tomadas inmediatamente.

- No se tiene ningún tipo de índice de los datos, la forma de localización de éstos se hace con base a la experiencia y memoria de quien los almacena.

- La forma en que se organiza la información es por medio de carpetas, una por cada reunión del pleno del Consejo. Las carpetas al igual que las reuniones no siguen un patrón rígido, por el contrario, los temas a

tratar varían de reunión en reunión; así las carpetas se podrían ver como registros variables de la base de datos que forman.

- Por cuestiones en el manejo de la información, en ocasiones documentos a los cuales se hace referencia en la actas o bien, no llegaron a integrarse a la carpeta, o fueron extraídos y no devueltos, además que no se excluye la posibilidad de error en el momento de integrar dichos documentos a la carpeta.

- El tiempo de integrar la carpeta es largo, ya que se puede decir que el trabajo se ha concluido hasta que todos los documentos que deben integrarse han sido recopilados para su almacenamiento.

- No hay un tipo de validación que defina si la información fue adecuadamente almacenada, si se integraron todos los documentos o si regresaron todos éstos después de una consulta.

- El espacio físico necesario para almacenar una carpeta por acta comienza a ser conflictivo, si se piensa que hay no menos de 10 reuniones por año.

- Una de las desventajas fundamentales es que el medio de almacenamiento es a través de papel, que es rápidamente degradado por el paso del tiempo llegando a ser ilegible, y se corre el riesgo de ser parcial o totalmente perdido.

I.3 SOLUCIÓN.

El medio a través del cual se pretende dar solución a los problemas antes planteados es, como ya se dijo, implementar un sistema de información automatizado el cual realice el trabajo de almacenamiento y recuperación de la información. El sistema deberá realizar un acceso a la información de manera rápida, en cualquier momento, que no dependa de una persona. También se deberá considerar para su ejecución que éste se ajuste a las características de los datos, los cuales no son susceptibles de manejarse eficientemente por medio de archivos, registros y campos ya que su naturaleza está más bien referida a textos como lo son actas y documentos adicionales del Consejo Técnico.

I.4 HERRAMIENTAS DE SOFTWARE.

Como ya se mencionó en los anteriores puntos I.1 y I.2 la característica principal de los datos del Consejo Técnico es que no siguen un patrón fijo, tal es el caso del texto de las actas. Este hecho hace muy complicado organizar la información en registros y campos, quedando el uso de la teoría clásica de base de datos poco apta para este fin, aunque no sería imposible resolver nuestro problema con la teoría clásica, pero se pagaría el alto costo de tener que ajustar nuestros datos a dichos modelos. Adelantándonos un poco al siguiente capitulo, el cual sustentará nuestra propuesta de buscar una nueva teoría de datos, podemos proponer lo siguiente: el sistema de base de datos a desarrollar no usará el modelo de

base de datos relacional, ya que como se planteará posteriormente, éste no presenta las bondades para poder desarrollar un sistema que haga un proceso de la información con las características que perseguimos. Se propone el uso de un sistema de base de datos inteligente usando el hipertexto.

El hecho de implementar un sistema de base de datos inteligente, no es por que sea imposible procesar nuestra información con alguna de las teorías de bases de datos como la jerárquica, la red o la relacional. El hecho más bien, como veremos en el siguiente capitulo, se debe a que cada teoría de bases de datos, en su tiempo, cubrió las necesidades de procesamiento de información que se les encomendaba, pero al crecer o cambiar el tipo de aplicaciones y el manejo de datos hizo necesario emigrar de una teoría a otra que cubriera las necesidades presentadas. Éste es el caso de nuestro problema, su desarrollo con una teoría de bases de datos, como la relacional, no seria eficiente y sólo obtendríamos una fracción de las ventajas que se podrían tener al darse los mecanismos adecuados de almacenamiento y recuperación de información que se ajusten a las características de nuestros datos. Estos mecanismos pueden ser facilitados por las bases de datos inteligentes.

CAPITULO II
MARCO TEÓRICO

II.1 EVOLUCIÓN DE LAS BASES DE DATOS.

La dirección que la computación sigue no sólo se ha determinado por el desarrollo en las capacidades de *hardware*[1] y *software*[1], sino también por las necesidades de las organizaciones que hacen uso de computadoras.

La forma en que han evolucionado las necesidades en el manejo de la información como resultado de el desarrollo de la sociedad, con un soporte de nuevas facilidades de *hardware* y *software*, ha dado como resultado cinco generaciones o formas diferentes en el manejo de la información dentro de la teorías de bases de datos, a partir del inicio de proceso de información en computadora.

Mencionaremos también que la transición de una generación a otra se debió al constante incremento en la complejidad de las aplicaciones que se le dan a las bases de datos y por el costo de implementación, mantenimiento, extensión de estas aplicaciones y recursos disponibles.

[1] Ver Glosario

Definamos como modelo de bases de datos convencionales a las primeras cuatro generaciones y, como teoría de bases de datos inteligentes a la quinta.

II.1.1 MANEJADORES Y MODELOS DE BASES DE DATOS.

Un sistema el cual maneja las tres siguientes tareas se conoce como un Sistema Manejador de Base de Datos (DBMS)[2].

1. Estructura general de los datos. La cual es definida usando un leguaje de definición de datos (LDD).

2. Los datos son modificados y actualizados usando un lenguaje de manipulación de datos (LMD), el cual también facilita la forma de expresar búsquedas.

3. Los métodos de recuperación de datos basados en búsquedas específicas los provee el lenguaje de búsqueda de base de datos (LBD), que también incluye facilidades para la protección de la integridad de la base de datos y para el manejo de los recursos del sistema.

Dada la proliferación de grandes bases de datos, éstas debían ser manejadas y administradas de alguna forma; aunque los DBMS que se

[2] Por sus siglas en Inglés *DataBase Management System.*

han desarrollado manejan el almacenamiento y recuperación de datos, aún no determinan por sí mismos la estructura óptima de la base de datos. Una base de datos debe ser diseñada de acuerdo con la funcionalidad y el uso para la cual fue diseñada. Como el mantenimiento y el uso de base de datos se volvió crítico para la operación de grandes organizaciones, grandes inversiones fueron hechas para proveer soporte a esto.

La funcionalidad del DBMS y de los sistemas de información descansan en una técnica bien establecida. La base de ésta consiste de métodos para estructurar y organizar datos que necesitan estar organizados. Los modelos de datos se especifican como piezas de información separadas, que están relacionadas por medio de una base de datos. La información es dividida en estructuras de datos. Las estructuras más básica de éstas son los registros y los archivos.

II.1.2 PRIMERA GENERACIÓN.

En un principio, para tener acceso a grandes cantidades de información y poder usarla en programas y aplicaciones, comúnmente especializadas en el manejo de sistemas administrativos, la información se almacenó en archivos y se manipuló por medio de programas escritos en lenguajes como FORTRAN, COBOL, PL/1[3].

[3] Ver Glosario.

Inicialmente estos programas eran desarrollados para aplicaciones especificas, sin embargo, rápidamente se vio que una gran cantidad de información era común a diferentes aplicaciones y se tuvo la necesidad de compartir datos. Como resultado de esto, archivos y datos fueron integrados en bases de datos centralizadas que proporcionaban los siguientes beneficios:

- Acceso compartido a la información.
- Seguridad.
- Mantenimiento rápido y eficientes actualizaciones.

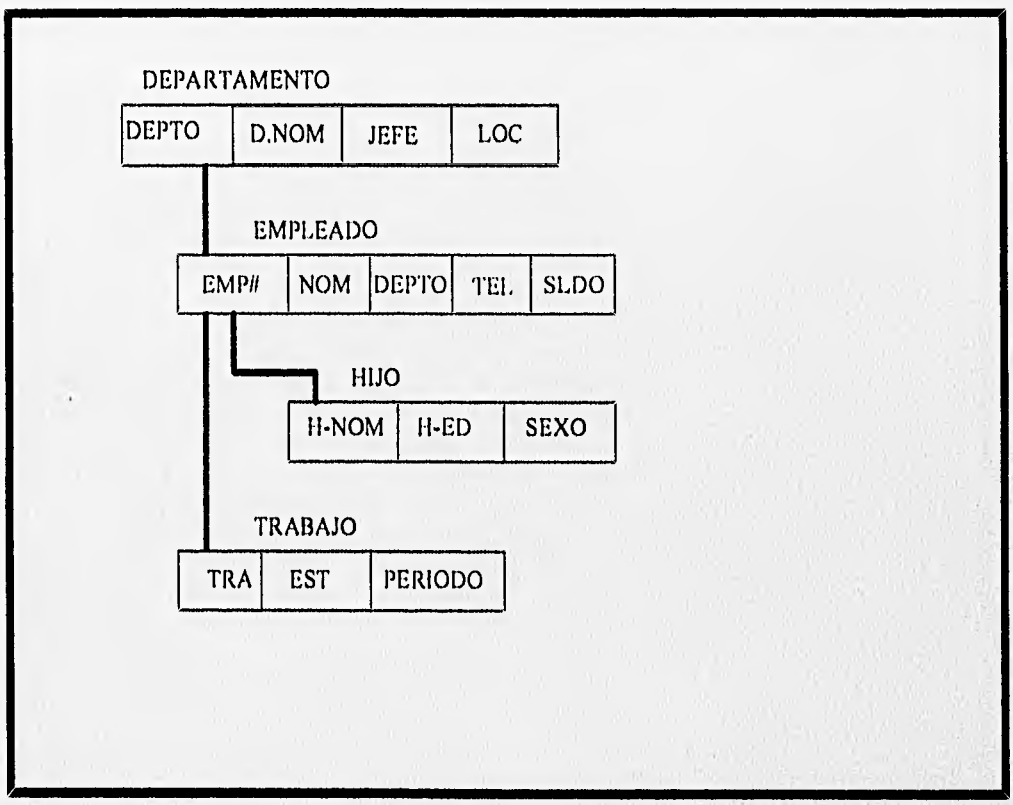
Grandes bases de datos centralizadas requerían métodos de estructuración y manejo de grandes cantidades de información. La información fue generalmente estructurada en términos de archivos, los cuales contienen objetos llamados registros, que a su vez son descritos por campos o atributos.

II.1.3 SEGUNDA GENERACIÓN.

Lo que llamaremos segunda generación se basa en el MODELO DE DATOS JERÁRQUICO.

Un número de modelos de datos han sido propuestos para organizar registros. El primero considerado aquí es el modelo jerárquico.

Una jerarquía o árbol, es una red en la cual los nodos son conectados por ligas. Los nodos en una jerarquía están estrictamente anidados, esto es, cada nodo tiene un padre excepto el nodo raíz de la jerarquía. En las bases de datos jerárquicas, los nodos de la jerarquía se constituyen de registros los cuales están ligados a cualquier otro nodo a través de ligas, como se ve en la siguiente ilustración.



La operación básica en este modelo es la búsqueda de información dentro del árbol. Dada una búsqueda en una base de datos jerárquica, la jerarquía es revisada y los nodos que correspondan con las condiciones de la búsqueda serán regresados como resultado de ésta.

Los datos son estrictamente anidados en jerarquías (un nodo solamente puede tener un padre). Pero el mundo real de los datos no siempre puede ser modelado por medio de una jerarquía. Una pieza de datos puede necesitar ser repetida en diferentes locaciones de la base de datos.

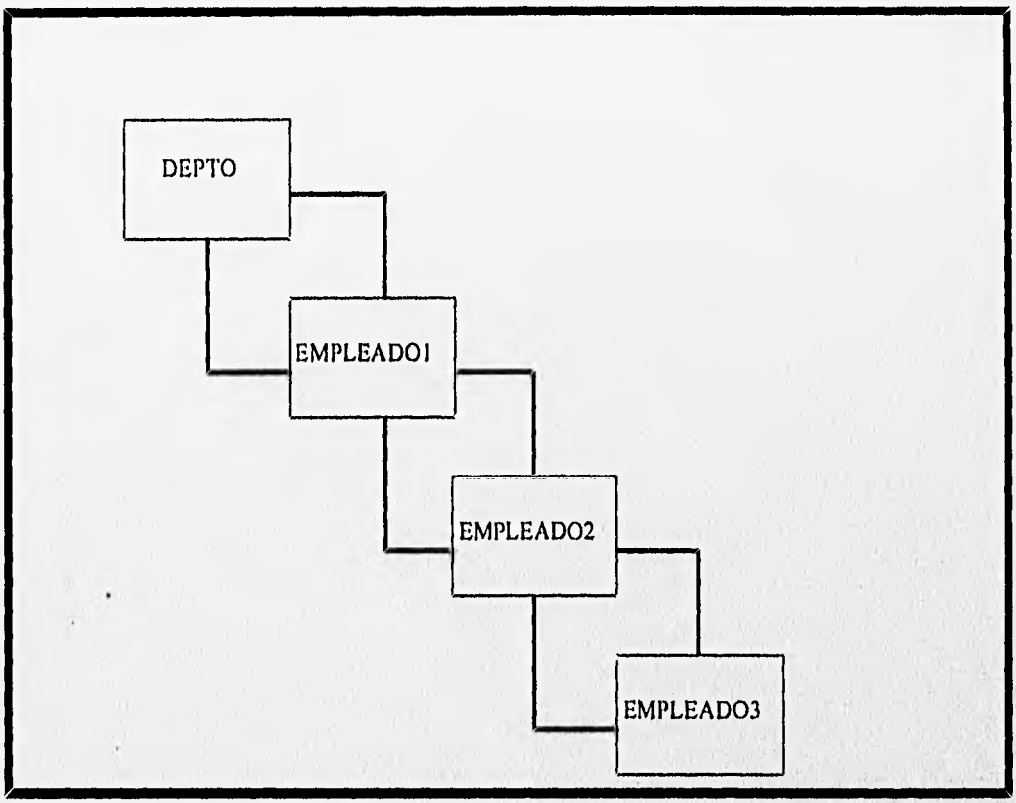
Las bases de datos jerárquicas comúnmente exhibían una pobre flexibilidad, aunque proveen un buen desempeño para aplicaciones preconcebidas. Algunos programas y aplicaciones clásicos (contabilidad, nómina, inventarios) han sido escritos en DBMS jerárquicos. Como consecuencia las aplicaciones realizadas con DBMS jerárquicos emigraron a ambientes relacionales.

II.1.4 TERCERA GENERACIÓN.

Nuestra tercera generación se basa en el MODELO DE DATOS DE RED.

El modelo red usa apuntadores adicionales para sumar flexibilidad al modelo jerárquico. De forma más general, podemos decir que una red es una conexión de nodos con uniones posibles entre cualquiera de éstos. Podemos asignar significados a esas uniones y así la creación de ligas entre nodos puede ser restringida. El modelo jerárquico, por ejemplo, es un caso especial del modelo red donde cada nodo es ligado a su nodo padre. En un modelo puramente jerárquico cada nodo debe tener solamente un padre, aunque por sí mismo puede ser padre de más un nodo de menor nivel. En el modelo jerárquico sólo existe un camino entre

cualquiera de dos nodos, mientras que en el modelo de red puede haber más de un camino, como se ve en la ilustración.



Las redes pueden ser implementadas de diferentes maneras, dependiendo de la forma en la cual los registros son apuntados y organizados. Un método utiliza una estructura de multilista. En la organización multilista, cada registro tiene un puntero por cada liga que éste contenga.

II.1.5 CUARTA GENERACIÓN.

Definiremos a la cuarta generación como aquella que se basa en el MODELO DE DATOS RELACIONAL.

Grandes bases de datos fueron almacenadas en *mainframes*[4] en los años sesenta y setenta. Mientras no existieron las computadoras personales, el mundo de las bases de datos estaba limitado a *mainframes* con aplicaciones escritas en lenguajes tales como COBOL y similares.

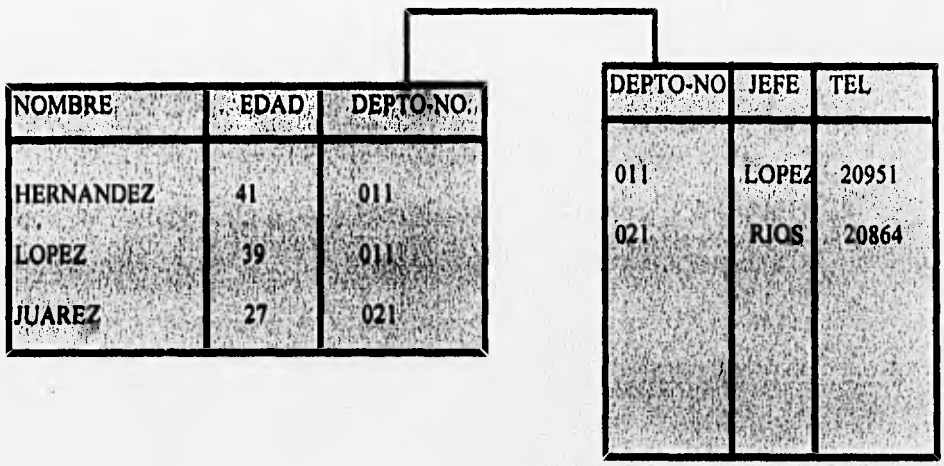
Al principio, las aplicaciones de bases de datos en *mainframes* eran dominadas por los modelos de red y jerárquicos. Pero a mediados de los años setenta, el modelo relacional gradualmente empezó a ganar adeptos.

A mediados de los años ochenta, ya con la proliferación de las computadoras personales, los DBMS relacionales comerciales empiezan a aparecer.

Una base de datos relacional es una base de datos que consiste de un sistema de tablas, cada renglón en una tabla representa una relación entre un conjunto de valores. Una razón para la amplia aceptación que tuvo el modelo relacional es que se basa en un modelo adelantado y conceptualmente fácil de comprender. El modelo puede ser resumido como sigue.

[4] Ver Glosario.

Una base de datos relacional se forma de tablas. Cada tabla lleva el nombre de la relación y se forma de renglones y columnas. Nuevas tablas pueden ser construidas cortando y pegando columnas y renglones de la tabla existente. El proceso de construir nuevas tablas en el modelo relacional es gobernado por las operaciones de el álgebra relacional[4].



La introducción de búsquedas declarativas en las Bases de Datos Relaciónales alivió a los programadores del tedioso quehacer de navegación y recuperación de registros dentro de la base de datos.

II.1.6 QUINTA GENERACIÓN.

Los intentos por hacer uso del modelo relacional en otras aplicaciones como CASE[5], sistemas expertos y multimedia, expuso rápidamente una serie de "limitaciones" de éste y de sus antecesores.

Los siguientes son algunas limitantes de los modelos de datos convencionales:

- Los modelos de datos convencionales, especialmente el relacional, son demasiado simples para el modelado de complejas entidades anidadas.

- En los modelos de datos convencionales, el uso de tipos de datos compuestos está limitado.

- Los modelos de datos convencionales, no incluyen conceptos semánticos frecuentemente útiles como son generalización[6] y agregación[7].

- Los programas de aplicación son llevados a cabo en lenguajes algorítmicos, (tal como COBOL, FORTRAN o PL/1). Los lenguajes de programación son muy diferentes de los lenguajes manejadores de bases de datos.

---

[5] Computer Aided Software Engineering. Sistemas avanzados de producción de software que pueden construir completamente programas por medio de diagramas.
[6] Es una relación de represión que existe entre una entidad de alto nivel y una de menor nivel.
[7] Es una abstracción a través de la cual relaciones son tratadas como entidades de alto nivel y así manejadas de la misma forma que cualquier otra entidad.

- El modelo de transacción soportado en los sistemas convencionales de bases de datos, es inapropiado para transacciones de larga duración necesarios en procesos interactivos.

Estos aspectos abrieron paso a los profesionales en bases de datos para desarrollar a finales de los años ochenta lo que llamaremos la quinta generación.

Las bases de datos de la quinta generación deberían contener las bondades de un modelo de base de datos relacional e incorporar soluciones para algunos de los anteriores problemas y así poder ser usadas en nuevas aplicaciones.

Existen al menos dos vías para la transición de la cuarta a la quinta generación: la técnica de base de datos relacional extendida y la técnica de bases de datos orientadas a objetos. La diferencia entre estas dos técnicas radica en el modelo y el lenguaje manejador de base de datos usado.

La base de datos relacional extendida inicia con el modelo y el lenguaje de búsquedas relacional y se extiende para permitir el modelado y manipulación de relaciones semánticas adicionales. Como por ejemplo, de este tipo de base de datos tenemos a la base POSTGRES.

Las bases de datos orientados a objetos tales como ORION, ONTOS, GemStone e IRIS, inician con un modelo de datos orientados a objetos y un lenguaje de base de datos que captura a este modelo y le adhiere nuevas capacidades.

Un punto importante que se debe reconocer es que, un modelo de datos orientado a objetos es una forma más natural que un modelo relacional extendido para resolver limitaciones de las bases de datos convencionales, como podrían ser soporte de tipos de datos generales y objetos anidados.

Una diferencia importante entre el modelo relacional extendido y el modelo orientado a objetos es que el último incluye conceptos de encapsulación, herencia y polimorfismo[8], los cuales no forman parte de los modelos convencionales; otra diferencia es que las bases de datos orientadas a objetos pueden directamente soportar las necesidades de aplicaciones las cuales crean y manipulan objetos.

Un sistema de base de datos orientado a objetos, el cual soporta a un lenguaje de programación y de base de datos unificado, puede ser la mejor plataforma para el desarrollo de aplicaciones de base de datos orientadas a objetos que una base de datos relacional extendida.

Si la vía de objetos orientados cumple con las espectativas para el diseño y desarrollo de *software* y representación de conocimiento en futuros sistemas basados precisamente en el conocimiento, la técnica se puede volver crucial como plataforma para el desarrollo de aplicaciones.

[8] Los tres términos se abordarán con mayor detalle al final de éste capitulo.

II.1.7 CARACTERÍSTICAS ADICIONALES.

La siguiente es una lista de las características adicionales en que los modelos de datos convencionales pueden extenderse.

- La habilidad para representar y manipular objetos anidados, para permitir el sucesivo refinamiento de entidades complejas.

- La capacidad para almacenar y recuperar datos de tamaños arbitrarios.

- La habilidad para definir y manipular tipos de datos arbitrarios, además de métodos eficientes para su almacenamiento y recuperación.

- La habilidad para hacer cambios en la base de datos en cualquier momento.

- La habilidad de manipular y representar varios conceptos de modelado semántico.

- La habilidad de especificar reglas y extender restricciones para soportar inferencias las cuales son necesarias como mecanismos básicos para el soporte de aplicaciones basadas en conocimiento.

- La habilidad para manejar transacciones de larga duración.

II.1.8 VÍAS DE ACCESO.

Actualmente hay un número considerable de lenguajes, así como sistemas de representación del conocimiento, *interfaces* y diseño asistido por computadora con técnicas orientadas a objetos. Estas aplicaciones, guardando sus correspondientes características, hacen uso de los conceptos orientados a objetos como son: herencia, clases, métodos, polimorfismo. Por lo tanto estas aplicaciones deben ser la llave para construir un tipo de sistemas de programación inteligente de alto desempeño.

Los conceptos orientados a objetos han evolucionado en tres diferentes disciplinas: en lenguajes de programación, bases de datos e inteligencia artificial.

II.2 BASES DE DATOS ORIENTADAS A OBJETOS.

Un modelo de datos orientado a objetos es una organización del mundo real de las entidades. Una base de datos orientada a objetos es una colección de objetos cuyo medio ambiente, estado y su relación son definidas en armonía con el modelo de datos orientado a objetos. Los conceptos de los sistemas orientados a objetos forman una buena base para un rico modelado de datos, que nos permitirá atender a la siguiente generación de aplicaciones planteadas a las bases de datos tales como: sistemas CASE, sistemas expertos y sistemas de información en

multimedia. El hecho que un modelo de datos orientado a objetos incluya las relaciones de agregación y generalización significa que una base de datos orientada a objetos provee una *interface* para usuarios que define y manipula la relación entre objetos. Esto a su vez significa que los programadores de aplicaciones no necesitan, explícitamente, manejar estas relaciones.

A pesar del actual interés por los conceptos de los sistemas orientados a objetos no existe, hasta ahora, un estándar para éstos. Esto encierra la pregunta de que si se deben construir sistemas de bases de datos orientadas a objetos, especialmente para fines comerciales. Los sistemas de bases de datos relacionales están ahora bien establecidos y los sistemas de bases de datos orientadas a objetos podrán explotar nuevos mercados, sin embargo, algunos creen que la programación orientada a objetos puede ser aún desarrollada y aceptada fuertemente en el futuro y debe entonces emerger un modelo de datos orientado a objetos estándar para que esto sea posible.

II.3 BASES DE DATOS INTELIGENTES.

Éstas representan una nueva técnica para el manejo de información, que ha evolucionado como el resultado de la integración de ventajas de los modelos de datos tradicionales con adelantos en campos tales como: programación orientada a objetos, sistemas expertos, e hipertexto.

Las bases de datos inteligentes en su conjunto ofrecen las siguientes características:

- Proveen herramientas inteligentes de alto nivel, las cuales permiten nuevas capacidades para discernir dentro del contenido de la base de datos; para poder extraer conocimiento de ellas.

- Hacer la información disponible a un mayor número de personas, ya que más gente puede ahora consultar los sistemas debido a su facilidad de uso.

- Lleva a cabo el proceso de toma de decisiones que involucra el uso de la información.

- Interrelaciona información de diferentes fuentes usando diferentes medios (imágenes, textos, números, voz, vídeo); así la información es fácilmente captada y utilizada por el usuario.

- Permite el uso del conocimiento y de la inferencia, haciendo la información fácilmente recuperable, visible, y apta para la toma de decisiones.

II.4 HIPERTEXTO.

Una de las barreras más comunes en el uso de la información, es una falla para identificar la interconectividad que nos permita reconocer ligas y similitudes entre piezas de información, que son normalmente almacenadas en localidades separadas. Para resarcir un poco dicha falla se ha desarrollado el hipertexto, que es una herramienta para construir y usar estructuras asociativas. Un documento normal es lineal y uno suele leerlo de inicio a fin, en contraste en el hipertexto el fin está abierto, es decir, que se puede "saltar" de una idea a otra dependiendo de lo que al usuario le interese. Lo más cercano a un documento en hipertexto es un diccionario de sinónimos, pero en éste la liga es entre palabras y, en el hipertexto se tienen disponibles ligas entre documentos y fracciones de texto.

II.4.1 HIPERTEXTO: DEFINICIÓN.

La mentalidad humana parece ser inherentemente asociativa. Hay fuentes de evidencia de esta asociatividad, partiendo de la estructura interconectada del cerebro para los patrones de memoria asociativa y pensamiento, el cual es frecuentemente observada en el comportamiento humano.

Esta estructura asociativa de la memoria es muy diferente de la forma lineal en la cual generalmente están organizados los libros. Un libro es un

arreglo lineal, representa un sólo camino a través de un tópico. En contraste, hay muchos posibles caminos a través de estructuras asociativas.

El hipertexto puede ser simplemente definido como la creación y representación de ligas entre piezas discretas de datos. Cuando estos datos, que pueden ser gráficos o sonoros, tanto como textos o numéricos, la estructura resultante es conocida como hipermedia. Conceptualmente la noción de hipertexto está cercanamente relacionada a la idea de redes semánticas, es decir, texto y datos son representados en nodos. Tal como en el caso de las redes semánticas, existen características que potencialmente deben ser asignadas a las líneas entre nodos. Debemos enfocarnos en una versión de hipertexto sin rebuscamientos que cuente con la mayoría de los tipos de ligas, las cuales son las más relevantes para el desarrollo de bases de datos activas.

El hipertexto es simultáneamente un método para almacenar y recuperar información. Éste incorpora la noción de piezas de información ligadas, permitiendo a los usuarios navegar a través de una red de trozos de información. La información se compone de dos partes: por como es almacenada en cada nodo y de que manera estos nodos de información están ligados a cada uno de los demás. Para nuestro propósito, el hipertexto debe ser visto como un sistema de base de datos con ligas irrestrictas entre registros y archivos.

El hipertexto, por sí mismo, no es una forma de inteligencia artificial (IA). Esto es, porque la IA y el hipertexto sirven a diferentes propósitos. El

hipertexto utiliza máquinas para aumentar la capacidad cognoscitiva humana a través de un almacenamiento dinámico como medio de información; mientras tanto, la IA tiene como propósito representar el conocimiento humano y la realidad física en una forma que permita un razonamiento sofisticado en máquinas.

Los promotores del hipertexto utilizan términos como *libertad y acceso* en vez de la tradicional ciencia de la información con sus criterios de evaluación que son *precisión* y *recuperación.*

También en la recuperación de textos completos, palabras y frases son usadas como índices *de facto* dados por el usuario. Así la recuperación convencional de texto completo permite al usuario crear sus propios índices sin cambiar la estructura de acceso para la recuperación. El hipertexto por otra parte nos permite hacer un *browse*[9] dentro del espacio del documento. En adición a las ligas entre documentos e índices también se pueden crear ligas dentro del conjunto de documentos. Así con el hipertexto, los usuarios pueden fácilmente moverse de un documento a otro a través de un encadenamiento particular o camino de ideas.

Es interesante hacer notar que los dos estilos de recuperación de información (índices convencionales y texto completo vs. hipertexto) tienen sus analogías en el proceso de recuperación en la mente humana.

[9] La traducción al Español de ésta palabra inglesa es ramonear u hojear, dentro de éste texto tiene la connotación de dejar ver y navegar por una estructura, seguiremos usando la palabra inglesa.

Hay algunos hechos en los cuales la existencia y uso del hipertexto puede ser justificada, las dos principales son:

1. Existe una correspondencia entre la forma asociativa en la cual la gente piensa y el hipertexto.

2. El hipertexto es un principio flexible de estructurar conocimiento que puede ser usado para generar y contener otras estructuras de conocimiento.

¿Por qué el texto o acceso a la información debe de ser lineal, cuando la mente claramente no lo es? Hay dos formas de contestar esta pregunta:

1. Según Bradshaw (Cit. Parsaye-Chignell 1989) la escritura y la generación de texto es dominada por la parte izquierda del hemisferio del cerebro el cual tiende a pensar linealmente.

2. Las pasadas técnicas de escritura y producción de texto no permitían de ningún modo texto no lineal.

La idea de texto no lineal ha sido usada por mucha gente cuando ésta trata de poner sus ideas en fichas y así poder hacer referencias de una ficha a otra. De hecho de esta forma intuitiva es como funciona HiperCard[10].

Los sistemas básicos de hipertexto permiten a los usuarios crear fragmentos de texto como nodos. Estos nodos son objetos en el sentido de

[10] Ver Glosario.

la programación orientada a objetos, la función fundamental(método) de los nodos objeto es, en general, desplegar información.

Los sistemas de hipermedia son más flexibles que los tradicionales métodos de estructuración de información, éstos permiten tener información en una variedad de formas para ser unidos a los nodos, así un nodo en hipermedia puede consistir de sonidos o imágenes, tanto como textos. Los nodos en hipertexto pueden generalmente incluir botones los cuales proveen ligas (envío de mensajes) a otros nodos. Sin embargo, estos botones pueden contener también métodos unidos a ellos, así poderosas capacidades de programación pueden ser contenidas dentro de la representación de hipertexto.

II.4.2 TIPOS BÁSICOS DE NODOS.

1. Nodos Texto. Estos consisten en fragmentos de texto. El texto por sí mismo puede ser un documento o representación de información básica.

2. Nodos Imagen. Las imágenes pueden ser contenidas dentro de los nodos textos o éstas pueden ser nodos por sí mismas.

3. Nodos Sonido. Al igual que las imágenes, los sonidos pueden ser incluidos dentro de un texto o existir como nodos por sí mismos.

4. Nodos Mixtos. Estos nodos contienen algunas combinaciones de texto, imágenes y sonidos. En algunos casos la misma información puede ser representada en una combinación de nodos unidos ó de un sólo nodo mixto.

5. Nodos Botones. Debemos hacer una distinción entre botones y ligas. Una liga conecta un par de nodos, mientras un botón ejecuta un procedimiento. Estos nodos permiten a la hipermedia actuar con *interfaces* de bases de datos o programas de alto nivel.

Los anteriores cinco nodos o clases representan información, más que interpretar conocimiento. Fueron diseñados para ser leídos, oídos o vistos. De forma natural se pueden expresar estos tipos de nodos como clases separadas dentro de una representación orientada a objetos de la red de hipermedia.

6. Nodo Texto-índice. Los nodos de tipo Texto-índice contienen un índice de ligas que apuntan a nodos índice.

7. Nodos Índice. Estos nodos consisten de un término índice. Estos pueden generalmente contener ligas que anotan hacia una definición del concepto representado por el término índice, ligas que anotan a términos relacionados o sinónimos, ligas las cuales corresponden a columnas en bases de datos relaciónales y ligas las cuales anotan hacia el nodo de texto índice que hizo referencia a éste.

8. Nodos Objeto. Estos nodos describen objetos. Estos consisten de ligas de herencia y de procedimientos encadenados. Estos pueden ser usados para representar la organización del conocimiento.

9. Nodos Regla. Estos nodos listan reglas y pueden apuntar hacia la combinación de objetos que satisfagan las reglas, justificación del uso de las reglas y explicación de cuando aplicarlas.

II.4.3 TIPOS DE LIGAS.

La variedad de nodos que pueden ser definidos en hipertexto hacen de esto una herramienta flexible de la representación del conocimiento. Esta flexibilidad es realzada al proveer una variedad de tipos de ligas. Las ligas describen la estructura de la hipermedia y proveen la capacidad de la exploración y *browse* de nodos. Dentro de éstos podemos distinguir los siguientes:

A) De navegación:

1. De movimiento a ligas. Estas ligas simplemente nos mueven a nodos relacionados. Estos nos permiten movernos a través de ó navegar por el hipertexto.

2. Ligas de tipo Zoom. Estas ligas expanden el nodo presente en un conteo más detallado de la información.

3. Ligas de tipo Pan[11]. Estas ligas regresan al nivel más alto de la vista del hipertexto.

4. Ligas de tipo Vista. La disponibilidad o activación de tales ligas está condicionada sobre el estado de interés o propósitos del usuario.

B) Ligas de organización e inferencia. Una segunda clase de tipos de ligas en hipermedia trabaja con la organización de los nodos y el encadenamiento de la hipermedia a un razonamiento de máquina más general y a capacidades de programación.

5. Ligas índice. Estas ligas mueven al usuario a través de un nodo índice al correspondiente índice de entrada para tal nodo. Estos índices pueden ser usados para entrar a la base de datos relacional o para encontrar documentos los cuales comparten un término índice particular. El indexamiento en hipertexto es una buena forma de controlar la proliferación de ligas entre nodos.

6. Liga de pertenencia. Estas ligas son similares a las usadas en redes semánticas para indicar pertenencia o categoría, también permiten a los nodos ser organizados.

7. Liga característica. Estas ligas son usadas para describir las propiedades de un nodo. Estas son generalmente usadas por objetos.

[11] Prefijo griego que significa todo.

8. Ligas de implicación. Estas ligas son usadas para conectar hechos en un árbol de inferencia.

9. Ligas de ejecución (botones). Esta liga permite a la hipermedia tener *interface* con programación de alto nivel. Los botones llevan a cabo acciones, comúnmente de ejecución de un código.

Entre los sistemas más comunes de hipertexto tenemos los siguientes: HiperCard, NoteCards, Guide, Authorware.[12]

II.4.4 DISEÑO DE HIPERTEXTO.

No es necesario decir que la edición de complejas estructuras de hipermedia se vuelve un serio problema en grandes sistemas de hipertexto. La tarea de convertir texto a hipertexto es tomada en mayor medida como un problema de capacidad de almacenamiento masivo.

La conversión de texto es un reto el cual requiere:

- Un cuidadoso análisis del documento original, con el fin de determinar su contenido y estructura. En algunos casos el determinar el contenido y estructura de un documento puede ser muy problemático aun para los expertos en estos textos y más aun cuando el documento fue creado

[12] Ver Glosario.

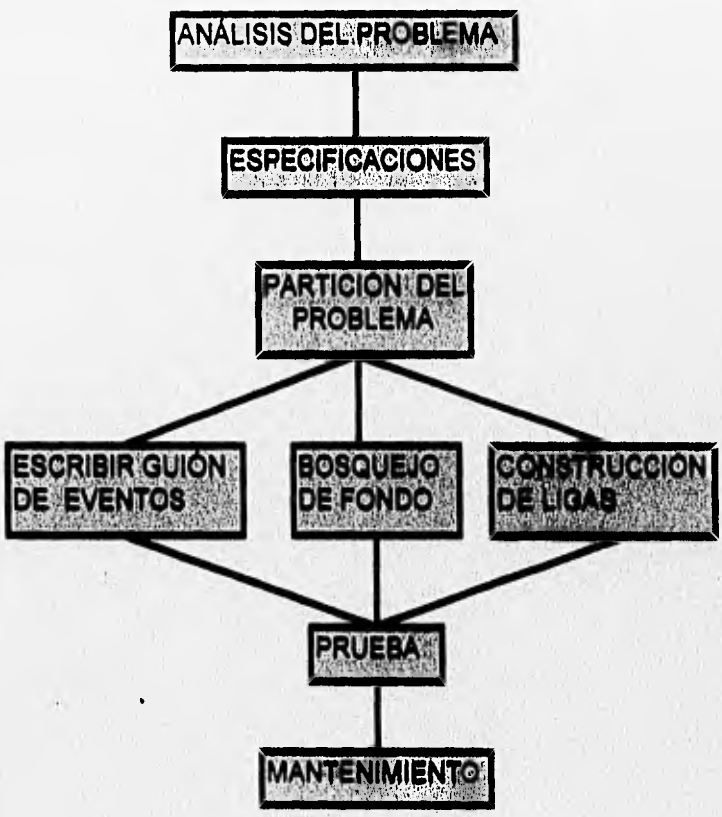
mucho tiempo atrás y no se puede estar seguro de la intención original del autor.

- Seccionar el texto y etiquetar estas secciones como instancias de un concepto particular

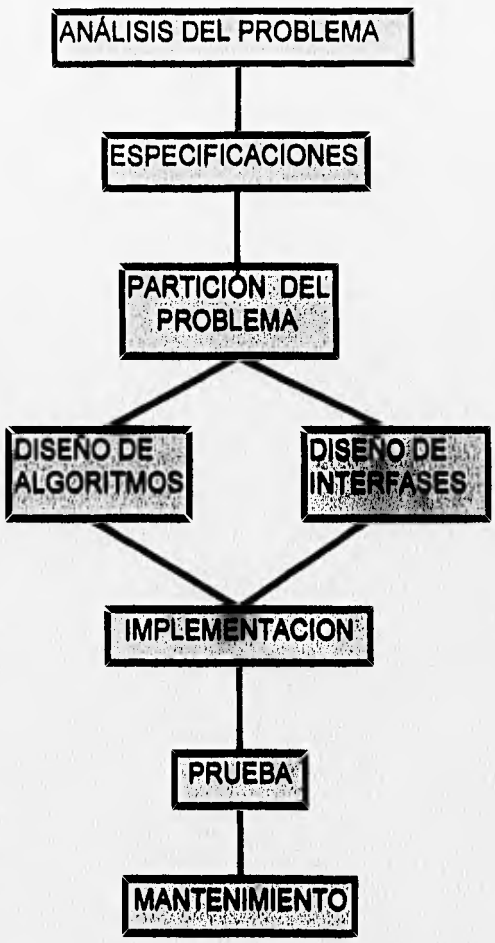
- Diseñar las ligas para que entonces se puedan unir fragmentos de textos los cuales comparten conceptos relacionados.

La expertos en el manejo del hipertexto sugieren que el proceso para construir este tipo de sistemas es, en algo, semejante al diseño convencional dentro de la Ingeniería de Software. En las siguientes figuras se muestran claras analogías entre los pasos de diseño en Hipercard, que es de los sistemas de hipertexto mas conocidos y los comúnmente usados en la Ingeniería de Software.

DISEÑO EN HIPERTEXTO



DISEñO EN INGENIERÍA DE SOFTWARE



Aunque no es un ambiente de programación orientado a objetos totalmente funcional, el diseño en hipertexto contiene algunos de los conceptos de los lenguajes de programación orientada a objetos tales como: objetos, métodos, mensajes, clases y herencia.

II.4.4 HIPERTEXTO Y BASES DE DATOS.

La tendencia en el desarrollo de bases de datos ha seguido generalmente dos caminos.

1. Una capacidad en incremento de la información, de indexamiento y la descripción de relaciones entre diferentes piezas de información.

2. Una capacidad en incremento para expresar búsquedas complejas.

En general, son desarrolladas formas más poderosas de describir información y datos, correspondientemente formas más poderosas de búsqueda de esa información están siendo posibles.

El hipertexto puede tener un impacto en las bases de datos, pero solamente si se encuentran las formas para hacer uso de su inherente flexibilidad.

Las aplicaciones de hipertexto en bases de datos requerirán de la capacidad de llevar a cabo búsquedas estructuradas.

Sin embargo, el problema de ligar bases de datos convencionales a hipertexto no ha recibido mucha atención.

La implementación de la funcionalidad de bases de datos relacionales dentro del hipertexto requiere de el uso de nodos índice y ligas índice.

La fuente principal de información del sistema que se plantea diseñar se tiene en texto (actas y otros documentos) por lo que el sistema de almacenamiento y recuperación de la información se desarrolló en hipertexto. Se hace uso extensivo de la Programación Orientada a Objetos (C++)[13]. Y además como modulo adicional el sistema es capaz de incluir un manejador de base de datos relacional tradicional.

¿Por qué se dice que la nuestra es una Base de Datos Inteligente? Por que al igual que las bases de datos inteligentes integra capacidades de las bases de datos tradicionales con adelantos en campos tales como programación orientada a objetos e hipertexto; además, en alguna proporción, debe contener las cinco ventajas que se definieron como características de las bases de datos inteligentes.

II.5 PROGRAMACIÓN ORIENTADA A OBJETOS.

La programación orientada a objetos es una nueva propuesta de cómo construir software, ésta es capaz de generar tanto grandes como pequeños sistemas de información, confiables, flexibles, mantenibles y capaces de evolucionar.

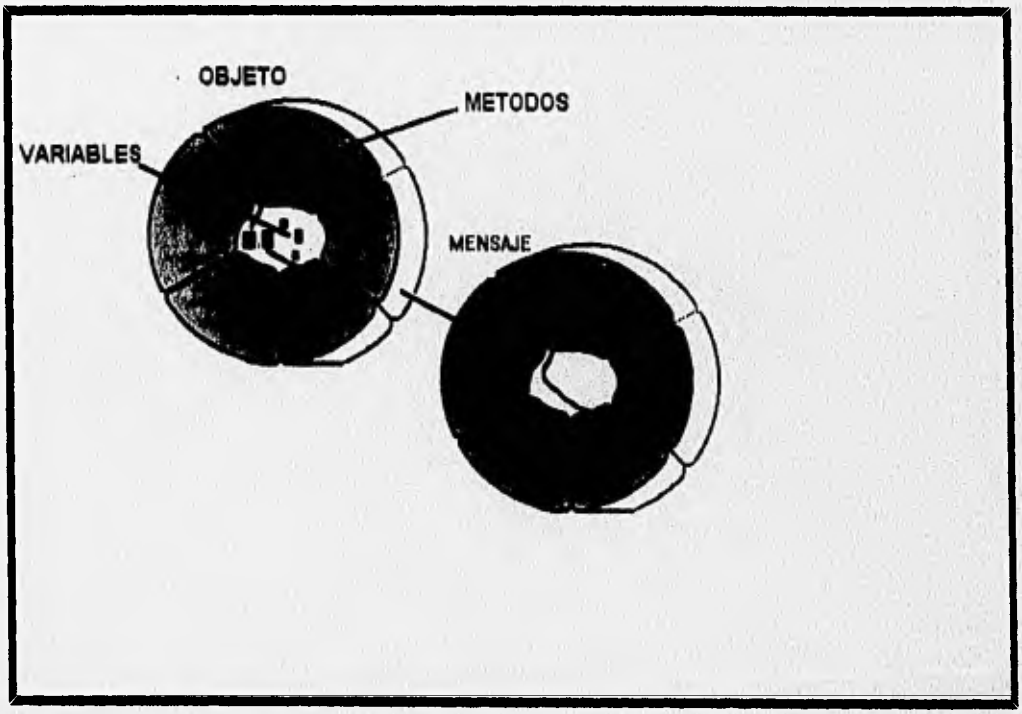
Popularmente se cree que la programación orientada a objetos es una técnica muy reciente, pero tiene más de 20 años de edad. Casi la mayoría de los conceptos de la programación orientada a objetos fueron

[13] Ver Apéndice A.

introducidos en el lenguaje de programación *Simula*, desarrollado en Noruega a finales de los años sesenta.

El concepto de objeto surgió de la necesidad de modelar objetos del mundo real en computadora. Un objeto es un paquete de software el cual contiene una colección de procedimientos y datos relacionados.

Los objetos hacen un módulo real de software, ya que éstos pueden ser definidos y mantenidos independientemente de otros; con cada objeto formando un universo completo. Cualquier cosa que el objeto "conoce" es una variable y cada cosa que un objeto puede "realizar" es un método.



La forma en la cual los objetos interactúan con otros es enviando mensajes, pidiendo que lleven a cabo sus métodos. Un mensaje es simplemente el nombre de un objeto seguido por el nombre de un método.

Si un método necesita información adicional para saber precisamente que hacer, se le envian en forma de parámetros.

Una clase es un patrón o modelo que define los métodos y variables que deben ser incluidos en un tipo particular de objeto. La descripción de métodos y variables los cuales soporten estos modelos deben ser definidos solamente una vez en la definición de la clase. Los objetos que pertenezcan a esta clase, llamados instancias de la clase, contienen sólo los valores particulares de las variables.

Así un objeto es una instancia de un tipo particular de clase, sus métodos y variables se definen en la clase y sus valores están definidos en la instancia.

La herencia es el mecanismo donde una clase de objetos puede ser definida como un caso especial de una clase más general, automáticamente incluye métodos y definición de variables de la clase general.

Casos especiales de una clase son conocidos como subclases de ésta. Además de los métodos y variables que hereda, las subclases pueden definir sus propios métodos y variables y supeditar alguna de las características heredadas.

Las clases pueden estar unidas en cualquier grado; la herencia puede acumularse conforme el nivel va disminuyendo y así se forma un árbol conocido como JERARQUÍA DE CLASES.

El conocimiento humano es estructurado de la misma manera, se basa en conceptos genéricos y su refina en el incremento de especialización de casos. La técnica orientada a objetos maneja el mismo mecanismo conceptual.

El software tradicional inicia con una especificación del problema a resolver, sigue con el diseño de un sistema que produzca el comportamiento requerido. El resultado en un sistema que desempeña el trabajo original, pero no otros trabajos. Un sistema de facturas no puede manejar la correspondencia del departamento de mercado o el sistema de inventarios del departamento de almacén.

La programación orientada a objeto aún mantiene el espíritu de la simulación del mundo real. El diseño de un sistema orientado a objetos se inicia modelando el mundo real para poder desempeñar su función. Una vez que los objetos del mundo real han sido correctamente representados el modelo puede ser usado para resolver una amplia variedad de tareas incluyendo, por supuesto, la original. Si se tiene el modelo correcto y su interacción con él, se puede usar éste adecuadamente tanto para facturas, correspondencia, etc.

La programación orientada a objetos tiene más ventajas además de la flexibilidad, por su estructura; el software puede ser entendido más fácilmente y ser modificado en el futuro, el aprovechamiento de software ya creado, corto tiempo de desarrollo, sistemas robustos y libres de error. La mayoría de la programación actual es escrita por líneas, donde se

recicla muy poco código, ya que al ser muy específico el problema es más fácil escribir nuevos procedimientos que adaptar los ya existentes.

Los objetos por el contrario son de propósito general, esto los hace fáciles de ser otra vez utilizados en subsecuentes proyectos, aun si son algo diferente. Mientras más clases sean acumuladas, el trabajo se transforma en hacer nuevas clases y a ensamblar clases ya existentes.

La programación orientada a objetos es más natural que la programación tradicional en dos diferentes niveles:

- Porque ésta organiza la información en formas que nos son naturales.

- En un nivel más profundo la programación orientada a objetos es más natural ya que ésta refleja las mismas técnicas de la naturaleza para el manejo de la complejidad.

La comunicación basada en mensajes ofrece dos importantes tipos de protección:

- Protege a las variables de un objeto de ser corrompidas por otros objetos.

- Un objeto protege a otros objetos de la complicación de depender de su estructura interna ocultando sus variables.

Con la encapsulación un objeto únicamente necesita conocer como preguntar a otro objeto por información. También los efectos de un cambio son limitados a una sola clase. Una de las ventajas de la programación orientada a objetos es que ésta permite hacernos pensar al nivel de un sistema del mundo real, y no al nivel de un lenguaje de programación.

En la programación tradicional los programadores eran forzados a pensar en un nivel del almacenamiento de datos en vez de pensar al nivel del problema que había que resolver, ya que los tipos de datos estaban ya construidos en cada lenguaje y no se podían modificar.

Los objetos compuestos son importantes ya que éstos pueden representar estructuras más complejas que lo que un sólo objeto puede.

Los objetos contenidos en objetos compuestos pueden tener a su vez otros objetos compuestos y llevar a cabo la composición a varios niveles, así la programación orientada a objetos puede representar la información de manera natural tal como lo hacemos nosotros, siempre una estructura profundamente anidada puede ser tratada como una simple.

La ventaja de un objeto se pierde si se usa aisladamente, su valor nace de su interacción con otros objetos. El vehículo de esa interacción es el mensaje.

Un mensaje consiste de tres partes, el nombre del objeto receptor, el nombre del método que el receptor conoce como ejecutor, y una serie de parámetros que el método requiere para llevar a cabo su función.

La sobrecarga de operadores es el medio por el cual podemos usar el mismo nombre de un método en cualquier clase.

El polimorfismo consiste en ocultar procedimientos alternativos detrás de una *interface* común. Este concepto es tan importante que es considerado una de las características que definen la técnica orientada a objetos. El beneficio del polimorfismo es que éste hace a los objetos más independientes uno del otro y permite que nuevos objetos sean agregados con mínimos cambios a los objetos ya existentes.

De la misma manera que un sistema de clasificación trae orden a un conjunto de plantas y animales en desorden, las clases permiten a los programadores organizar complejos sistemas en una forma racional y ordenada.

La función básica de una clase es definir un tipo particular de objeto. Una vez definida la clase se pueden crear un número de instancias únicas de ésta. La clase define las características compartidas por todas las instancias, considerando que las instancias por sí mismas contienen la información que las hace únicas.

Así una instancia no es más que una serie de entradas para contener valores.

La jerarquía de clases es un mecanismo muy eficiente porque podemos usar definiciones de métodos y variables en más de una subclase sin duplicar su definición y es básica para la forma en que un objeto localiza a un método.

La habilidad para definir métodos en más de un nivel se usa para crear excepciones o reglas generales.

La técnica de anulación[14] es importante ya que permite que casos especiales sean manejados eficiente y fácilmente con un mínimo impacto en otros objetos, también refleja la manera en que la gente aprende y retiene información construyendo reglas generales y anulándolas con casos especiales.

La herencia múltiple entra en juego cuando una clase de objeto debe jugar múltiples roles y cada uno de esos roles es caracterizado por alguna otra clase.

El gran poder de la jerarquía de clases es que ésta aplica reglas generales a un amplio grupo de objetos, mientras se hacen excepciones a esas reglas. Así cuando el sistema orientado a objetos localiza la definición de un método, el sistema aplicará la definición más especializada.

[14] Ver Glosario.

CAPITULO III.
METODOLOGÍA

III.1 ANÁLISIS.

La idea de automatizar la búsqueda de información en las actas de las sesiones del Consejo Técnico, se inició con el análisis de los posibles medios para realizar esta tarea. Fueron dos hechos principales los que se tomaron en cuenta:

1. Localización de información en lapsos cortos.

2. Esta información se encuentra principalmente en textos.

Los datos no son susceptibles de ser organizados eficientemente en renglones o columnas para hacer uso de una base de datos con modelo relacional, ya que se trata de ideas; no existen índices o llaves para hacer la búsqueda, es decir, en ocasiones el único dato llave que se podía tener para reducir el espacio de búsqueda era una fecha, muchas veces ambigua ya que podía ser el periodo de una administración, un año; muchas veces estas fechas eran producto de la memoria de una persona.

En concreto la única pista que se tiene es la de una idea. Ejemplo: "Localizar bajo que términos se aprobó el Reglamento de Prácticas Escolares de la Facultad de Ingeniería".

Lo que si se tiene es la presión de localizar esta información ya que se requiere como antecedente para una toma de decisiones, en el caso de nuestro ejemplo, para una actualización.

Cuando se inició el análisis de las herramientas de software posibles para el desarrollo del sistema se estudiaron las siguientes:

- Un manejador de bases de datos relacional como ACCESS de Microsoft, pero quedó descartado desde un principio por todo lo anteriormente expuesto en el capitulo II.

- Una base de datos orientada a objetos, llamada GemStone, pero no se disponía de este software en la Facultad; su precio era muy alto para adquirirlo; además el equipo donde debía ser instalado como mínimo, debía ser una estación de trabajo[15].

- Hacer uso de un paquete de desarrollo en multimedia llamado *Authorware* que nos permite el uso del hipertexto y aún más de la hipermedia, de una manera semejante al CASE, pero desgraciadamente, por cuestiones de presupuesto no fue posible su uso en esta tesis.

- Finalmente se encontró que la Unidad de Servicios de Cómputo Académico de la Facultad de Ingeniería, dependiente de la Secretaria General, cuenta con el paquete Visual C++ 2.0, que es un compilador de C++, el cual a su vez, es un lenguaje orientado a objetos y una

[15] Es un tipo especial de computadoras caracterizado por su alta capacidad de procesamiento, memoria y un gran despliegue gráfico, además de hacer uso de la tecnología RISC.

Lo que si se tiene es la presión de localizar esta información ya que se requiere como antecedente para una toma de decisiones, en el caso de nuestro ejemplo, para una actualización.

Cuando se inició el análisis de las herramientas de software posibles para el desarrollo del sistema se estudiaron las siguientes:

- Un manejador de bases de datos relacional como ACCESS de Microsoft, pero quedó descartado desde un principio por todo lo anteriormente expuesto en el capitulo II.

- Una base de datos orientada a objetos, llamada GemStone, pero no se disponía de este software en la Facultad; su precio era muy alto para adquirirlo; además el equipo donde debía ser instalado como mínimo, debía ser una estación de trabajo[15].

- Hacer uso de un paquete de desarrollo en multimedia llamado *Authorware* que nos permite el uso del hipertexto y aún más de la hipermedia, de una manera semejante al CASE, pero desgraciadamente, por cuestiones de presupuesto no fue posible su uso en esta tesis.

- Finalmente se encontró que la Unidad de Servicios de Cómputo Académico de la Facultad de Ingeniería, dependiente de la Secretaria General, cuenta con el paquete Visual C++ 2.0, que es un compilador de C++, el cual a su vez, es un lenguaje orientado a objetos y una

[15] Es un tipo especial de computadoras caracterizado por su alta capacidad de procesamiento, memoria y un gran despliegue gráfico, además de hacer uso de la tecnología RISC.

herramienta de desarrollo en ambiente Windows. Casi desde el inicio del presente trabajo se pensó en Windows, ya que la *interface* de ventanas es muy adecuada para el usuario. Además Visual C++ provee una serie de herramientas de producción de software tal como: MFC (Microsoft Fundation Class Library, que es un conjunto organizado de clases prefabricadas) las cuales permiten el manejo del ambiente de ventanas, entre otras ventajas; también permite el uso de elementos avanzados como son OLE[16] y ODBC[17]; otra ventaja fundamental para seleccionar a Visual C++ es que por medio de éste se puede hacer uso de el compilador de ayuda de Windows, el cual se incluye en el paquete y que es la herramienta a usar para convertir texto lineal en hipertexto.

III.2 ESPECIFICACIONES.

Para el desarrollo del sistema se recomienda un equipo con las siguientes características:

Software. Windows 3.1 o superior
- Word 6.0 para Windows
- Compilador de Ayuda de Windows.
- Visual C++ 2.0

[16] OLE. Microsoft Fundation Object Linking and Embedding conjunto de clases C++ que encapsulan la funcionalidad de aplicaciones escritas para Windows. Estas clases forman parte de Microsoft Foundation Class Library (MFCL)

[17] ODBC Object Microsoft Open Data Base Connectivity. Es una interface escrita en lenguaje C, para el enlace con bases de datos. Permite a las aplicaciones accesar datos en un DBMS usando SQL como estándar.

Hardware. Sistema PC con las siguientes características:
Procesador 80386 o superior.
8 MB en RAM (recomendable)
20 MB de espacio libre en disco.

III.3 PLANTEAMIENTO.

Después de seleccionar la herramienta de desarrollo adecuada, la realización de nuestro sistema se puede resumir en dos tareas:

1. Realización de un sistema de software en ambiente Windows, que permita el manejo del hipertexto, lo llamaremos Sistema Hipertexto.

2. Convertir el texto lineal del acta de las sesiones en hipertexto, siguiendo una forma de diseño adecuada, a esta tarea la llamaremos Generación de Hipertexto.

III.4 DISEÑO.

En el diseño del Sistema Hipertexto se debe contemplar las dos tareas principales que éste debe manejar:

1. Que permita la inclusión de la opción de ayuda ya que, como se especificó, la herramienta para convertir texto lineal a hipertexto es la del compilador de ayuda que tiene Windows.

2. También se debe incluir en el sistema la opción de que éste, usando las facilidades de Visual C++ pueda contener una base de datos relacional para poder hacer consultas de datos referentes al Consejo Técnico que no necesiten estar en hipertexto.

La tarea Generación de hipertexto, como ya se comentó en el capitulo II.4.4, no es una tarea fácil, de hecho, será la parte más complicada del sistema ya que es necesario: analizar las actas, fragmentarlas en tópicos, definir las características de éstos, la forma en que serán desplegados, además, discernir los tipos de ligas a usar. En conclusión un trabajo de diseño arduo, basado en la experiencia de quienes manejan la información del Consejo.

III.5 IMPLEMENTACIÓN.

III.5.1 SISTEMA HIPERTEXTO.

Esta implementación consistió de los pasos que a continuación se definen y se ilustran en el diagrama.

- Uso de AppWizard, la cual es una herramienta de desarrollo integrada en Visual C++. Ésta selecciona dentro de MFC las clases necesarias para el manejo de ventanas y la opción de ayuda, en este caso particular, que se adecuaran al diseño de requerimientos. AppWizard genera código en una serie de archivos, los principales son:

  CONSEJO.H, es aquí donde se hace la definición de clases.

  CONSEJO.CPP es el archivo principal donde se implementan las funciones de las clases y se utilizan.

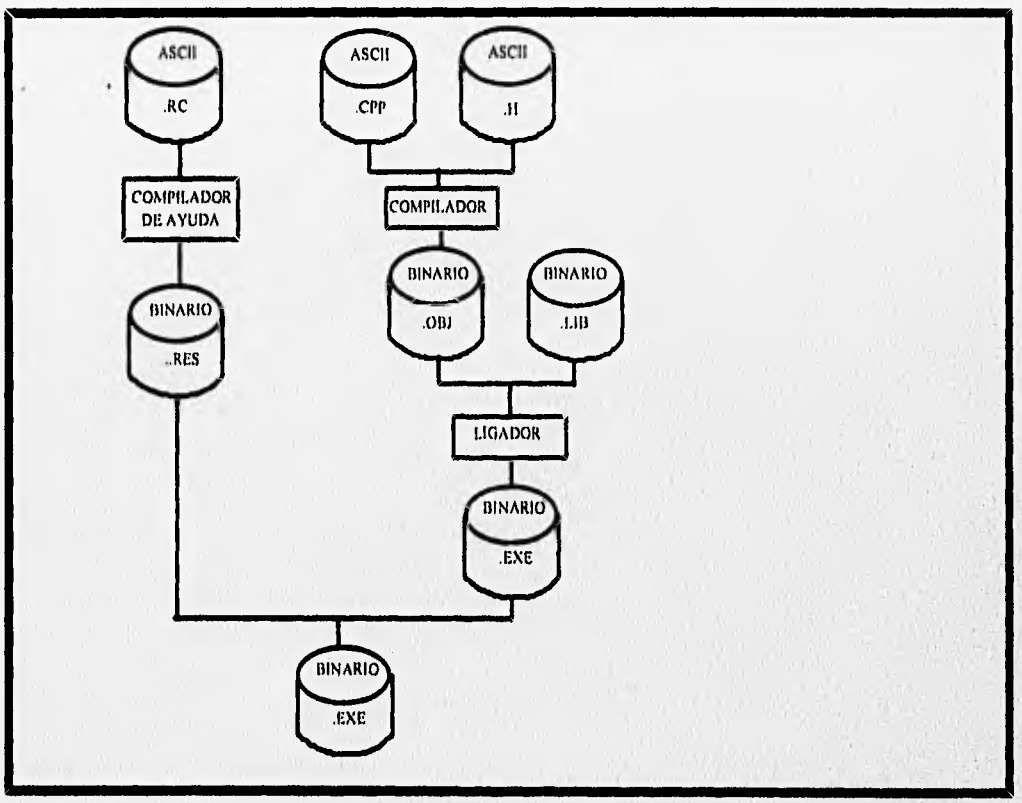
  CONSEJO.RC, el cual es un archivo ASCII[18] con la definición de los recursos a usar por el sistema tal como: iconos, cajas de dialogo, imágenes.

- Una vez que se tienen los anteriores archivos estos se deben compilar. Esta compilación se hace por separado. Por un lado el archivo CONSEJO.RC es procesado por un compilador especial llamado *RESOURSE COMPILER*, el cual genera un archivo en código binario llamado CONSEJO.RES. Por otro lado, los archivos CONSEJO.H y CONSEJO.CPP son procesados por el compilador de C++ y se genera un archivo en código binario llamado CONSEJO.OBJ.

- El proceso de ligado también se lleva a cabo por separado. El archivo CONSEJO.OBJ junto con las librerías pasan por el ligador para generar

[18] Siglas de American Standard Code for Information. Es un código estándar para representar caracteres como números binarios, este código es usado en la mayoría de las computadoras, impresoras y terminales.

así un archivo binario llamado CONSEJO.EXE. Ya una vez que se tienen los archivos CONSEJO.RES y CONSEJO.EXE son unidos para obtener como resultado un archivo ejecutable que tendrá como una de sus características la opción de ayuda.



III.5.2 GENERACIÓN DE HIPERTEXTO.

El proceso de generación de hipertexto sigue los pasos, que se comentan e ilustran a continuación:

El texto lineal se tiene en archivos con extensión .DOC (generados por Word 6.0). Dentro de estos textos se hace el diseño de los diferentes tópicos, es decir, el texto se fracciona en fichas según la experiencia, ya que no todo es fragmentado, sólo aquel que es susceptible de ser consultado. Aquí también se hace el diseño de las diferentes ligas y sus características.

Ejemplo.

El Ing. Solís da lectura a un documento enviado por el maestro Gilberto Silva Romo, quien realiza actividades dentro de un convenio de colaboración entre la Facultad de Ingeniería y PEMEX, en el que explica que su inasistencia a esta reunión se debe a la proximidad de la visita de supervisión de campo por parte de técnicos de Petróleos Mexicanos.

Por otra parte, el Secretario informa que el consejero técnico alumno Vladimir Meléndez Flores, le notificó la imposibilidad de asistir a la reunión de hoy debido a problemas de salud.

Aquí tenemos una sección de texto de algún acta, se puede obtener de ésta un tópico llamado Inasistencias con dos fichas, la primera relativa a la falta justificada del consejero profesor y la segunda a la de un consejero alumno. Se diseña que en estas fichas las palabras claves **inasistencia** e **imposibilidad de asistir** se vean en color azul y que además al inicio de la ficha se vea la fecha de la sesión y el número de página donde aparece, en color rojo.

Se diseña una liga de Inasistencias con las dos fichas y que además se pueda hacer un *browse* de éstas.

- El archivo .DOC es transformado en un archivo .RTF (Word hace ésta transformación) .

- Se hace la definición de las fichas y de las ligas dentro del archivo .RTF usando los directivos del compilador de ayuda. Ejemplo:

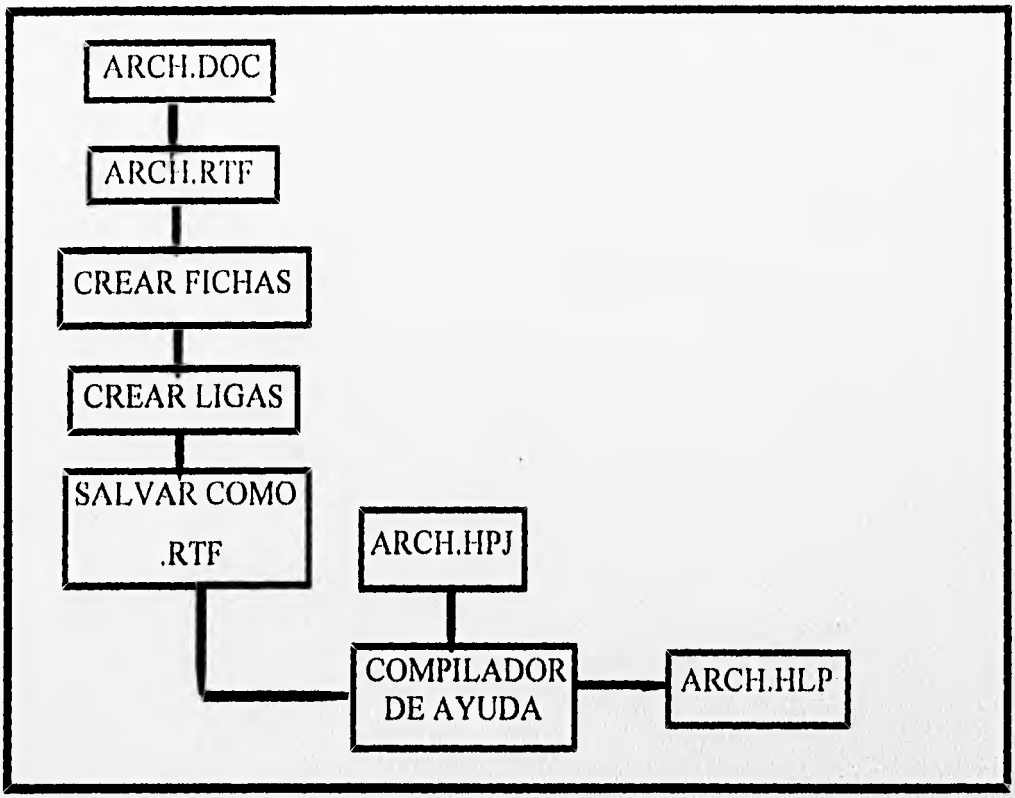
{\page} #{\footnote inasistencia1} k{\footnote inasistencias de consejeros} ${\footnote inasistenicias} +{\footnote inasistencia1}{\f5\fs25\cf6\keepn \pard}
{\f5\fs25\cf6\keepn ACTA DE LA REUNION\line DEL 31 DE ENERO DE 1996\line PAGINA.1\line\pard}{\f5\fs21\expnd0\expndtw-2 El Ing. Sol\'eds da lectura a un documento enviado por el maestroGilberto Silva Romo, quien realiza actividades dentro de un convenio de colaboraci\'f3n entre la Facultad de Ingenier\'eda y PEMEX, en el que explica que su {\cf2\fs25 inasistencia} a esta reuni\'f3n se debe a la proximidad de la visita de supervisi\'f3n de campo por parte de t\'e9cnicos de Petr\'f3leos Mexicanos.\par }\pard \qj\tx-1440\tx-720\tx0\tx360\tx720\hyphpar0 {\f5\fs21\expnd0\expndtw-2\par}\pard\qj\tx-1440\tx-20\tx0\tx360\tx720\hyphpar0{\page}

#{\footnote inasistencia2}k{\footnote inasistencias de alumnos}${\footnote inasistenicias}+{\footnote inasistencia2}{\f5\fs25\cf6\keepn ACTA DE LA REUNION\lineDEL 31 DE ENERO DE 1996\line PAGINA.1\line\pard} {\f5\fs21\expnd0\expndtw-2 Por otra parte, el Secretario informa que el consejero t\'e9cnicoalumno Vladimir Mel\'e9ndez Flores, le notific\'f3 la {\cf2\fs25 imposibilidad de asistir }a la reuni\'f3n de hoy debido a problemas de salud.\par }\pard \qj\tx-1440\tx-720\tx0\tx360\tx720\hyphpar0{\f5\fs21\expnd0\expndtw-2 \par }\pard \qj\tx-1440\tx-720\tx0\tx360\tx720\hyphpar0{\page}

- Por separado usando un editor de código ASCII, tal como Write de Windows, se edita un archivo .HPJ y es aquí donde se hace la definición de las características que contendrá el archivo de ayuda, que en este caso será nuestro archivo en hipertexto. Ejemplo:

```
[OPTIONS]
TITLE=CONSEJO TECNICO
COMPRESS=true
WARNING=2
 [FILES]
hlp\960131.rtf
hlp\960229.rtf
hlp\960306.rtf
hlp\960418.rtf
 [CONFIG]
BrowseButtons( )
 [BITMAPS]
[ALIAS]
 [MAP]
#include <c:\msvc\mfc\include\afxhelp.hm>
#include <hlp\tesis.hm>
```

- El proceso de compilación se lleva a cabo cuando ya que se tienen las ligas y fichas definidas en el archivo .RTF y el archivo .HPJ editado, éstos pasan por el compilador *HELP COMPILER* el cual genera un archivo con terminación .HLP, éste es el archivo ya en hipertexto que puede ser usado en nuestro Sistema de Hipertexto.



III.6 PRUEBAS.

Después de las primeras pruebas del sistema, aún sin hipertexto, se localizaron funciones que en un principio o bien, se habían contemplado aparecieran y otras que no debían ser incluidas. Estas "fallas" se originaron ya que Visual C++ tiene la posibilidad de hacer uso de clases las cuales ya están creadas y contenidas en MFC. Al seleccionar la clase principal de generación de ventanas ésta contenía como se dijo antes condiciones no previstas o bien faltaban algunas, y fue necesario adecuarlas al planteamiento del sistema y probarlas una vez más, éstas,

conforme aumentaba la cantidad de código fuente, se hicieron más lentas ya que el compilador tarda más en programas extensos. Así se continuó hasta llegar al sistema final.

Ya con el sistema de manejo de hipertexto funcionando, se hicieron las pruebas del sistema trabajando con datos, para revisar su desempeño. Además otro de los fines de estas pruebas fue comprobar si la liga entre las diferentes partes del texto era la adecuada y verificar su presentación (posición en la pantalla, congelamiento de cierta parte de la pantalla, barras de corrimiento, color, efectos).

III.7 RESULTADOS.

El producto de la presente tesis es un sistema de información funcional y su principal característica es la de facilitar la consulta a la información del Consejo.

Este sistema requiere de la captura de datos de manera organizada y bien detallada, no tiene la capacidad de directamente llegar a ser hipertexto por el sólo hecho de convertir texto de cualquier formato a RTF, sino que, requiere de un gran trabajo en el análisis del documento, su fragmentación y el diseño de ligas entre estos fragmentos. Si bien el trabajo seguirá realizándose de una forma manual será de una manera organizada.

La ventana principal muestra una serie de opciones. Una de éstas, la más importante, es la que nos une con la parte del sistema encargada de la consulta y presentación por medio del hipertexto de los datos contenidos en las actas.

Entre otras opciones el usuario puede hacer consultas de bases de datos relaciónales con información que es susceptible de ser almacenada en una de éstas, gracias a que una de las bondades de usar Visual C++ está la de poder embeber otros productos de software (OLE 2.0). Estos datos pueden ser referentes al Consejo Técnico tales como, número de reuniones al año y su carácter (ordinarias o extraordinarias), asistencia a dichas reuniones. Esta información, será capturada en la Unidad de Apoyo del mismo.

También en un futuro se puede incluir información referente al Personal Académico de la Facultad. Esta ya se encuentra en una base de datos, realizada por la Unidad de Servicios de Cómputo Académico.

CAPITULO IV.

CONCLUSIONES

La necesidad de desarrollar formas rápidas de conversión de texto en hipertexto puede proveer un gran estímulo al desarrollo de métodos de procesamiento de texto.

La hipermedia en CD-ROM.[19] Otro concepto del diseño; se relaciona con las características de dispositivos de almacenamiento masivos usados en almacenamiento de hipermedia.

De primera instancia el CD-ROM pudiera parecer un medio de almacenamiento inapropiado a la hipermedia por su naturaleza estática, sin embargo, la hipermedia y el CD-ROM pueden trabajar juntos, si una adecuada arquitectura en hipermedia es adoptada.

Una vez que la hipermedia es almacenada en el CD-ROM, ésta no puede ser removida o alterada.

La implementación de hipermedia en el CD-ROM, elimina el problema de actualizar y previene la posibilidad de modificaciones.

[19] Ver Glosario.

Una línea en la cual nuestro sistema pudiera continuar, es precisamente en el almacenamiento de hipertexto en CD-ROM, ya que este texto no debe ser alterado y nos da la posibilidad de almacenar años de información en un solo disco, lo que actualmente se hace en varias carpetas.

Otra línea es la de hacer a nuestro sistema de hipertexto en hipermedia, al incluir secciones de grabación, de película o imágenes extraídas de las reuniones del Consejo Técnico, o de otro tipo de fuente pero relacionada con el mismo.

Nuestro sistema es capaz de contener un DBMS, pero actualmente no lo contiene, sería conveniente incluirlo en un futuro junto con la información que se tiene en otros departamentos de la Secretaría General relativa al Consejo Técnico.

GLOSARIO DE TÉRMINOS.

Álgebra relacional.
El álgebra relacional es un método para operar relaciones, éstas son consideradas comúnmente como conjuntos de registros, donde cada registro se describe en términos de campos de llaves o atributos.

Cuando dos relaciones comparten un atributo, se dice que existe dependencia funcional, o están relacionadas. La dependencia funcional se puede expresar como una relación de 1-a-N.

Existen cinco operaciones fundamentales en el álgebra relacional: selección, proyección, producto cartesiano, unión y diferencia de conjuntos.

Anulación.
Es un caso especial de sobrecarga en el cual el mismo nombre es asignado a un método o variable en dos o más niveles de la misma rama de una jerarquía de clases. Cuando esto sucede, el nombre que está en la parte más baja de la jerarquía toma precedencia, anulando las definiciones más generales dentro de la jerarquía.

Authorware.
Funciona mediante una representación visual, organiza iconos y muestra como están relacionados y se afectan entre si, dentro de una línea de flujo. Con Authorware podemos presentar y combinar el contenido de la

información en una variedad de formas, usando la línea de flujo, podemos orquestar cuándo y cómo aparecerá el contenido. El texto es un elemento clave de muchas piezas, puede ser usado como prompts, etiquetas, direcciones o éste puede ser el principal elemento de una pieza.

Bases de datos activas.
Almacenan objetos con métodos "vivos" que se pueden activar directamente en la base de datos.

Bases de datos pasivas.
Almacenan un objeto separando los métodos y los datos de la misma, usualmente los métodos en un archivo externo separado de la base de datos. Estos métodos solamente pueden ser usados una vez que han sido retomados nuevamente.

C.
Lenguaje de programación desarrollado en los laboratorios Bell durante los años setenta. C es un lenguaje de propósito general, pero a diferencia de otros lenguajes de este tipo, da al programador completo acceso a la parte interna de la máquina, que lo hace apto para actividades que debiera realizar un lenguaje ensamblador. Además hace un desarrollo de programación de una manera muy eficiente.

CD-ROM.
Compact Disc Read Only Memory, es un medio de almacenamiento masivo de sólo escritura de una gran capacidad, no menos de 500 megabytes.

información en una variedad de formas, usando la línea de flujo, podemos orquestar cuándo y cómo aparecerá el contenido. El texto es un elemento clave de muchas piezas, puede ser usado como prompts, etiquetas, direcciones o éste puede ser el principal elemento de una pieza.

Bases de datos activas.
Almacenan objetos con métodos "vivos" que se pueden activar directamente en la base de datos.

Bases de datos pasivas.
Almacenan un objeto separando los métodos y los datos de la misma, usualmente los métodos en un archivo externo separado de la base de datos. Estos métodos solamente pueden ser usados una vez que han sido retomados nuevamente.

C.
Lenguaje de programación desarrollado en los laboratorios Bell durante los años setenta. C es un lenguaje de propósito general, pero a diferencia de otros lenguajes de este tipo, da al programador completo acceso a la parte interna de la máquina, que lo hace apto para actividades que debiera realizar un lenguaje ensamblador. Además hace un desarrollo de programación de una manera muy eficiente.

CD-ROM.
Compact Disc Read Only Memory, es un medio de almacenamiento masivo de sólo escritura de una gran capacidad, no menos de 500 megabytes.

COBOL.

COmputer Bussines Oriented Language. Fue desarrollado a principios de los años sesenta por varias empresas de computación y el Departamento de Defensa de los Estados Unidos. Su aplicación se da en sistemas de procesos administrativos y de negocios.

FORTRAN.

FORmula TRANslator. Fue creado a fines de los años cincuenta y fue el primer lenguaje estructurado que permitió a los programadores describir cálculos a través de fórmulas matemáticas.

Guide.

Fue el primer sistema de hipertexto disponible para equipo Macintosh, fue inicialmente concebido para ser una herramienta en la construcción de documentos electrónicos. Aunque por si mismo no es un procesador de texto totalmente funcional, provee formas flexibles de construir documentos electrónicos con una variedad de referencias cruzadas y anotaciones.

Hardware.

El hardware en un sistema de computación consiste de todos los elementos físicos dentro de la computadora tal como: circuitos integrados, cables, teclados, monitores, manejadores de disco.

Hotspot.

Área que representa un comando en particular, para ejecutar éste se debe apuntar con el puntero del mouse y presionar el botón.

HyperCard.

Desde su introducción en 1987 por Bill Atkinson se ha convertido en el sistema de Hipertexto más ampliamente usado. Existe una analogía entre la forma en que se diseña en HyperCard y la generalmente usada por la Ingeniería de Software. Aunque HyperCard no es un sistema totalmente funcional en un ambiente orientado a objetos, tiene incluidas algunas de las características de los lenguajes de programación orientados a objetos, tales como: objetos, métodos, mensajes, clases y herencia.

Interface.

Es el "envoltorio" de software de una computadora que hace el manejo de ésta fácil de aprender, simple de utilizar, directo y no muy estricto.

Mainframe.

Computadora de gran capacidad de procesamiento, dispuestas en salas especiales con sistema de aire acondicionado. Estas máquinas soportan de 100 a 500 usuarios a la vez. Como ejemplo tenemos a las IBM 370 y 3081.

NoteCards.

A pesar de ser referenciado comúnmente como un sistema estructurado de información, es también un sistema de hipertexto desarrollado en Xerox PARC, los diseñadores ven a NoteCards como un ambiente de procesamiento de ideas, se construye básicamente de una red semántica hecha a partir de una serie de NoteCards interconectadas por algún tipo de unión. NoteCards incorpora un número de ideas de manejo de

información las cuales pueden ser útiles en el desarrollo de una base de datos inteligente.

PL/1.

Iniciales de Programming Language. Poderoso lenguaje de programación de computación desarrollado por IBM a principio de los años sesenta para trabajar en los sistemas IBM/360.

Pop up.

Es una clase especial de tópico, el cual se ilustra como una pequeña ventana que emerge y es momentánea.

Software.

Es el conjunto de programas que dicen a la computadora que hacer, este término contrasta con el de hardware, entre algunos de los productos de software tenemos: compiladores, lenguajes de programación, interpretes, procesadores de palabras, hojas de cálculo.

Windows.

Producto de Microsoft, el cual proporciona una *interface* gráfica entre el usuario de la computadora y el sistema operativo MS-DOS, trabaja por medio de la selección de iconos, botones, menús y cajas de dialogo.

APÉNDICES

APÉNDICE A.

LENGUAJE DE PROGRAMACIÓN C++

El lenguaje de programación C++ fue desarrollado por Bjarne Stroustrup quien es miembro del centro de investigaciones en ciencias de la computación de los laboratorios Bell AT & T. El lenguaje fue originalmente desarrollado debido a que el autor buscaba algo para realizar simulaciones, para lo cual *Simula67* pudiese ser ideal, excepto por consideraciones de eficiencia, "C con clases", que fue la primera forma en cómo se conoció al C++, fue usado para proyectos más extensos de simulación, en los cuales la facilidad para escribir programas que usasen un mínimo de tiempo y espacio fueron ampliamente probadas. "C con clases" carecía de sobrecarga de operadores, referencias, funciones virtuales. C++ fue por primera vez instalado fuera del grupo de investigación del autor en julio de 1983.

El nombre C++ significa la evolución natural de los cambios de C. "++" es el operador de incremento en C. El nombre C+ es un error de sintaxis; conocedores de la semántica de C encuentran C++ inferior a ++C. El lenguaje no es llamado D, ya que éste es una extensión de C.

EFICIENCIA Y ESTRUCTURA.

C++ fue desarrollado tomando como base el lenguaje de programación C y con muy pocas excepciones contiene a C como un subconjunto. El lenguaje base, el subconjunto C de C++, es diseñado de tal forma que hay una correspondencia entre sus tipos, operadores, enunciados, números, caracteres y direcciones.

C++ fue diseñado para que grandes programas pudiesen ser estructurados de una forma racional, de tal forma que éstos no fueran incomprensibles para los programadores.

Un lenguaje de programación sirve para dos propósitos que se relacionan, provee un vehículo para que el programador pueda especificar acciones que deban ser ejecutadas y un conjunto de conceptos para el programador los use cuando piense acerca de que puede hacer. El primer aspecto idealmente requiere un lenguaje el cual es " cercano al de la máquina", así aspectos importantes de la máquina han sido simple y eficientemente manejados de una forma que sea razonablemente obvia para el programador. El segundo aspecto idealmente requiere un lenguaje el cual es "cercano al problema a resolver", así los conceptos de una solución pueden ser expresados directa y concienzudamente. Las facilidades agregadas a C para crear C++ fueron principalmente diseñadas pensando en lo anterior.

APÉNDICE B.

PROGRAMAS FUENTE

```
//consejo.cpp : Define el comportamieto de las clases dentro de la aplicación.
#include "stdafx.h"
#include "consejo.h"
#include "mainfrm.h"
#include "consejodoc.h"
#include "consejovw.h"
#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CConsejoApp
BEGIN_MESSAGE_MAP(CConsejoApp, CWinApp)
        //{{AFX_MSG_MAP(CConsejoApp)
        ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
                // NOTE - ClassWizard agregará y quitará macros de mapeo
                //   EL SIGUIENTE CODIGO LO ESRCIBE AppWizard, NO
DEBE //ALTERARSE
        //}}AFX_MSG_MAP
        // // Comandos estándar para el manejo de documentos.
        ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
        ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
        // Comandos estándar para la impresión de archivos
        ON_COMMAND(ID_FILE_PRINT_SETUP,
CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// Construcción de CConsejoApp
CConsejoApp::CConsejoApp()
{}
```

```
CConsejoApp NEAR theApp;
// inicio de CConsejoApp initialization
BOOL CConsejoApp::InitInstance()
{
	SetDialogBkColor();	// Fija el color de dialogo estándar
	LoadStdProfileSettings();  // Carga los archivos de opción INI,
//incluyendo //(including MRU)
	// Registra los templates de aplicaciones.
	//· sirve como conexión entre documentos, ventanas y vistas.
	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CConsejoDoc),
		RUNTIME_CLASS(CMainFrame),    // ventana principal SDI
		RUNTIME_CLASS(CConsejoView));
	AddDocTemplate(pDocTemplate);
	// crea un documento nuevo (vacio)
	OnFileNew();
	if (m_lpCmdLine[0] != '\0')
	{
	}
	return TRUE;
}
/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialogo usado para App About
class CAboutDlg : public CDialog
{
public:
	CAboutDlg();
// Datos del dialogo
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA
// Implementación
protected:
```

```
        virtual void DoDataExchange(CDataExchange* pDX);    // soporte
DDX/DDV
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// App command para correr el dialogo
void CConsejoApp::OnAppAbout()
{
        CAboutDlg aboutDlg;
        aboutDlg.DoModal();
}
/////////////////////////////////////////////////////////////////////////////
// comandos CConsejoApp
```

APÉNDICE C.

LLAMADAS AL COMPILADOR DE AYUDA.

\ansi	Especifica el conjunto de caracteres ANSI
\b	Inicia texto de tipo bold
\bmc	Despliega archivos bitmap o metafile en el texto
\bml	Despliega archivos bitmap en el margen izq.
\bmr	Semejante al anterior pero en el margen derecho
\cf	Fija el color base
\colortbl	Crea la tabla de colores
\deff	Fija el font por default
\f	Fija el font
\fonttbl	Crea la tabla de fonts
\footnote	Define información especifica del tópico
\fs	Fija el tamaño del font
\i	inicia texto en itálicas
\keepn	Crea una región non-scrolling
\line	Rompe la línea en uso
\page	Finaliza el tópico en uso
\pard	Restablece las propiedades de default del párrafo
\plain	Restablece propiedades de default de los caracteres
\rtf	Especifica la versión RTF
\strike	Crea a un hotspot
\tab	Inserta el carácter tab
\ul	Crea una liga a un tópico pop-up
\uldb	Crea a un hot spot
\v	Crea una liga a un tópico

BIBLIOGRAFÍA

- Kim, Won

  Introduction to Object-Oriented Databases

  MIT Press Third Printing, 1992

- Parsaye, Kamran

  Intelligent Databases

  Jhon Wiley & Sons, Inc. 1989

- Taylor, David A.

  Object-Oriented Technology

  Addison-Wesley Publishing Company 1993

- Roetzheim, William

  Programming Windows with Borland C++

  Ziff-David Press California, 1992

- Perry, Paul J.

  Using Visual C++ 2

  Que Corporation 1994

- Kruglinski, David

  Inside Visual C++

- Schmitz, Joe

  Using Authorware 3

  Macromedia Incorporation 1995

- Stroustrup, Bjarne

  The C++ Programming Language

  Addison-Wesley Publishing Company 1986

- Universidad Nacional Autónoma de México.

  Compilación de Legislación Universitaria.

  Primera Edición 1992.