UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

"TEORÍA, METODOLOGÍA Y CONSIDERACIONES PARA UN DISEÑO ÓPTIMO DE SISTEMAS DE BASE DE DATOS"

T E S I S

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

ROBERTO ARENAS GASCA

ASESOR:

SILVIA VEGA MUYTOY

MÉXICO **2005**

Índice

Prefacio

Este trabajo como se maneja en su título es acerca de la teoría, metodología y las consideraciones a tomar para un diseño óptimo de sistemas de Base de Datos, y no acerca de la escritura detallada de código. Hay muchos asuntos de programación y técnicas así como el hardware de computadora que están más allá del alcance de este escrito. El propósito de este documento es tratar la teoría, metodologías y consideraciones planteadas y si bien se hace mención a algún tipo de software o hardware la última palabra la tendrá el encargado del proyecto en función a las necesidades específicas.

Este escrito surge pensando en el alumno que cursa una carrera afín a las ciencias informáticas o bien recién egresados que no poseen la experiencia suficiente para desarrollar Bases de Datos sólidas en su implementación; sin embargo desarrolladores, administradores, analistas y programadores encontraran algo útil en este escrito. Ya sea usted un novato o veterano con muchos proyectos hallará en estas páginas los principios fundamentales que forman la base de una ingeniería de software exitosa.

El profesional de la tecnología de la información actual está llegando a ser cada vez más especializado. Hay simplemente demasiado que aprender para poder saber de todo. Usted se puede hacer un nicho de carrera especializado; tomando únicamente una o dos técnicas haciéndose realmente muy bueno en ellas. Otros pueden optar por un enfoque más generalizado, dominando muchas de las técnicas junto con una diversidad de lenguajes de programación y conocimientos técnicos.

Introducción

La información abundante es una ayuda poderosa, pero no una garantía. Con una cantidad mayor de información se pueden cometer errores más grandes y por supuesto más responsablemente cometidos.

La misma super-abundancia de información ha hecho su obligada clasificación y diseminación a través de una incontable multitud de canales que la distribuyen con rapidez para hacerla llegar a quienquiera. Y así, de esta necesidad de orden es que han surgido las bases de datos.

Las bases de datos, depósitos y almacenes inmensos de información de todo tipo están empezando a ser la fuente de información más segura y fiable de información a la que el individuo de hoy puede acudir. Cuando han sido bien diseñadas, pueden prestar un servicio incalculable a todos los hombres y mujeres que desempeñan cualquier tipo de responsabilidad en el ámbito que sea, a los estudiosos e investigadores, a los gobernantes y ejecutivos, a estudiantes y profesores, a periodistas, artistas y profesionales, y en fin, a todo quien sea mínimamente sensible al fenómeno informativo que está en el centro de nuestras vidas y que las va a condicionar y orientar a nuestro futuro inmediato.

Visto desde fuera, la nota más vistosa de los ingenios informáticos es la velocidad. Una velocidad que podría ser a la vez espectacular e inútil. Correr ¿para qué?, si corriendo mucho no se llegase a donde hay que llegar. La racionalidad y la velocidad han de ir juntas, de otro modo correr es un gesto insensato.

Habría que hablar de una incidencia recíproca entre la informática y la documentación siempre que se trata de las bases de datos. Las bases de datos

han surgido y proliferado porque, al ser una conjunción de dos técnicas distintas —informática y documentación—, satisfacen plenamente una exigencia de exactitud y de rapidez que sólo la suma de ambas podría proporcionar.

El éxito en el desarrollo de software de calidad se basa en cuatro pilares fundamentales: "el factor humano (los participantes en el proyecto), la notación y herramientas utilizadas para especificar y construir el sistema, y finalmente el proceso de desarrollo"[1].

Para lograr el objetivo de este escrito que es el análisis técnico en la implementación de un sistema óptimo de Base de Datos el trabajo se ha dividido en cinco capítulos que partiendo de un estudio teórico analiza y planteada una óptima metodología de desarrollo tomando en cuenta consideraciones importantes que logren ir cimentando un camino firme desde el planteamiento de la Base de Datos hasta su puesta en marcha. Así es como los capítulos se dividen en:

Capítulo I Conceptos Básicos. Nos empapa en los términos y conceptos a utilizar en el desarrollo de un sistema de Base de Datos lo que nos beneficio al ubicarnos en una misma sintonía de estudio del tema.

Capítulo II Análisis técnico de la Base de Datos. Este apartado resulta fundamental ya que estudia detenidamente las fallas más recurrentes que suceden durante el análisis para la implementación, así como plantear paso a paso el camino óptimo para el desarrollo de la Base de Datos tomando en cuenta factores como el equipo (software y hardware), tiempo, costos operativos, las personas que se involucran y la toma de decisiones.

[1] Piattini, M., García, F., Calidad en el desarrollo y mantenimiento del software. Página 15. Editorial Alfaomega. México. 2003.

Capítulo III Lenguajes para Base de Datos. Aquí estudiamos las características que poseen los lenguajes de programación con que son escritas las Bases de Datos así como la importancia de una buena selección del mismo.

Capítulo IV Manejador de Base de Datos (DBMS). Otra parte primordial es esta que analiza a los manejadores o gestionadores de una Base de Datos sus modelos y propósitos particulares.

Capítulo V Interfaces (IU). Dentro del desarrollo de casi cualquier sistema resulta ya imprescindible el atento estudio de lo que es una interfaz de usuario ya que en la mayoría de los casos una buena presentación y claridad dan aceptación a un sistema de tal forma que se analizan los errores más recurrentes en el aspecto físico-funcional de una interfaz para una Base de Datos.

Esta introducción va mostrando el armazón a tratar de este trabajo que es ir dando la pauta técnica para el buen desarrollo de una base de datos que no sobrepase a la empresa y cubra las expectativas requeridas.

Capítulo I

Conceptos Básicos

Este capítulo busca mostrar los términos y conceptos básicos para los sistemas de Base de Datos.

En cualquier tema tratado siempre es conveniente hacer mención o reflexión a los aspectos fundamentales que trata dicho tema.

I.1. Conceptos

Al realizar un análisis de las Bases de Datos no se puede dejar de hacer mención a elementos esenciales. Por ello, resulta importante el detenerse a repasar los conceptos básicos de elementos que contienen y cual es su relación con la creación de una Base de Datos ya que ello facilitará la comprensión del funcionamiento de ellas.

El Dato

Para definir este concepto el Diccionario de la Real Academia Española[2], señala tres alternativas:

1) Antecedente necesario para llegar al conocimiento exacto de una cosa o para deducir las consecuencias legítimas de un hecho.

2) Documento, testimonio, fundamento.

3) En informática: es la representación de una información de manera adecuada para su tratamiento por una computadora.

Ante las distintas opciones que se ofrecen, la más acertada es considerar un concepto único que se maneje también dentro del lenguaje común pero sin posibilidad de error y es la de los autores Pérez y Pino quienes señalan que dato es

[2] Real Academia Española. Diccionario de la Lengua Española. Vigésima primera edición. Editorial Espasa Calpe S.A. Madrid, España. 1995.

"la representación de una cierta entidad del mundo real en alguna forma de símbolo". [3]

Frente a este concepto, se debe agregar que las **representaciones** que constituyen los datos se refieren a entidades relativas a personas, objetos, lugares y acontecimientos o hechos. Entendiéndose por entidad aquella unidad básica de información descrita mediante sus atributos (es decir sus características) y mediante sus relaciones con otras entidades.[4]

En definitiva, los datos vienen a constituir la forma más conveniente de representar los atributos esenciales de las entidades para su almacenamiento y comunicación, puesto que ellos son por su naturaleza algo concreto.

La Información

Cuando los datos son organizados (de acuerdo a una forma lógica y consecuente) y procesados de manera significativa que facilite su interpretación y la toma de decisiones, ellos adquieren la cualidad de convertirse en algo útil y pasan a denominarse información. La información es subjetiva y su significado depende de la interpretación del receptor.

Por ejemplo, un informe impreso en Internet que contenga la lista de los sectores productivos del país que se vieron más afectados por los delitos de piratería de obras intelectuales durante el último año, viene a constituir una información bastante importante para quienes se desempeñen como abogados, para quienes detenten la calidad de titulares de derechos de autor y de derechos

[3] Pérez V., Victor L. y Pino U., José A. Curso de Computación e Informática. Volumen II. Estructuras de datos y organizaciones de archivos. Página 17. Editorial Universitaria. 1992.
[4] Villanueva Lara, Julio E. Computadoras y Procesamiento de datos. Página 73. Serie de Matemática. Monografía n° 28.Secretaría General de la O.E.A. Programa Regional de Desarrollo Científico y Tecnológico. Washington, D.C. 1987.

conexos, para los directivos de una sociedad de gestión colectiva de estos derechos, etc.

En virtud de lo anterior, se puede sostener que información son todos aquellos datos que contribuyen a reducir la incertidumbre o riesgo de un evento; o bien, faciliten la comunicación o adquisición de conocimientos que permitan ampliar o precisar los que se posean respecto de una materia determinada.

Si realizamos un breve análisis de la definición anteriormente mencionada, se puede desprender de ella que:

- El valor de la información estará directamente relacionado con el valor de la decisión en que participa o el del riesgo que disminuye, y su costo estará asociado al trabajo que implique recolectar, seleccionar y procesar los datos que conforman la materia prima para producirla. En tal sentido, la información aparece por regla general como algo "temporal", ya que si la decisión fue tomada o el riesgo disminuido, la información pasa a convertirse en un simple dato.

- Existirá un tipo de información que es atemporal; o dicho de otro modo, que no perece con el transcurso del tiempo. Este tipo de información es la que constituye el conocimiento o inteligencia acumulada, es la información que hace al hombre, a diferencia, de un animal un ser histórico.[5] Para determinar cómo se produce toda esta transformación de los datos en información será necesario tener en cuenta lo siguiente:

[5] En este sentido se pueden revisar la definición dada por Nuñez Valencia, Oscar en el seminario "Bancos de Datos y Propiedad Intelectual en Informática" (realizado en Santiago en Septiembre de 1987) citado por Mayo, Marie Claude en "Informática Jurídica" y los comentarios de esta autora al respecto. Página 65. Editorial Jurídica de Chile. 1991.

a) Cuando el usuario obtiene la información, ésta previamente ha sido ordenada y procesada a través de un sistema, esto es, un modelo de ordenamiento que se aplica a una determinada organización que opera en un entorno cambiante. Lo que en definitiva producirá una colección de elementos (datos) que al interrelacionarse operarán en conjunto para lograr un objetivo común (transformar estos elementos en información útil para el usuario).

b) Cuando todo lo anterior se realice en un entorno informático, el resultado se obtendrá con la utilización de una serie de dispositivos computacionales que permiten ingresar, almacenar y procesar los datos. Dichas herramientas técnicas serán manejadas por una o más personas conjuntamente con una serie de programas y procedimientos que serán supervisados a través de diversos medios de control con el fin de obtener el adecuado procesamiento de la materia prima (datos) lo que conducirá a la obtención del producto deseado (información).

Dentro de este ámbito, se ha determinado que recurrir a las Bases de Datos constituye la opción más acertada para enfrentar los requerimientos de los distintos tipos de usuarios que cada día necesitan manejar mayores volúmenes de información.

En virtud de lo anterior, se puede sostener que las Bases de Datos constituyen actualmente un eslabón esencial en la cadena que forman los generadores de datos, los procesadores de éstos y los distribuidores de información.

Base de Datos

Las Bases de Datos no son una novedad de la informática sino que muy por el contrario, su origen es anterior a dicha ciencia, ya que pueden equipararse a las recopilaciones de datos, ficheros y archivos que manejados con técnicas manuales permitían lograr resultados similares.

Lo anterior, puede confirmarse al analizar las labores que desempeña un diseñador de una Base de Datos Computacional y el organizador de una compilación tradicional. Se verá que básicamente la forma en que operan ambos sujetos es la misma, realizando el acopio, incorporación y descarte de las piezas de acuerdo a su criterio personal. Dicho procedimiento incluirá la realización de un análisis sobre el material que sea seleccionado con la finalidad de determinar sus significados, relaciones y consecuencias.

Como consecuencia, se puede afirmar que el organizador de la Base de Datos Computacional y el de una compilación tradicional actúan de la misma forma, esto es, seleccionando y reuniendo distintos datos e información en un sólo cuerpo, con una organización sistemática.

Por lo tanto, actualmente "se considera que no tendría por qué conectarse necesariamente la expresión *Base de Datos* con el almacenamiento de datos en una computadora, pues todas las compilaciones de información deberían considerarse *Bases de Datos*, existan o no en forma impresa en unidades de almacenamiento electrónicas o en cualquier otra forma."[6]

[6] Fernández Ballesteros, Carlos. El Derecho de Autor y los Derechos Conexos en los Umbrales del año 2.000. Libro Memoria del Primer Congreso Iberoamericano de Propiedad Intelectual. Tomo I. Página 126.

I.2. Los componentes de una Base de Datos

En los sistemas de Base de Datos se distinguen tres principales componentes que son:

- Una **estructura** diseñada por el organizador para disponer los datos o información seleccionados,

- Un **contenido** (datos o información), y

- Los **elementos necesarios para el adecuado funcionamiento del sistema**.

Los **componentes** de las Bases de Datos Computacionales se vinculan conformando un verdadero "rompecabezas" que se va armando del siguiente modo:

a) La Base de Datos Computacional, desde el punto de vista de su diseño, consta de dos tipos de **estructura**:

   1. **Estructura lógica:** Ella se refiere a la manera en qué el analista y el programador ven **conceptualmente** los datos. Aquí se producirá, para el caso mecanizado, la definición de las funciones computacionales, la cadena lógica de pantallas, los programas, archivos, etc.

   2. **Estructura física:** Se refiere a la descripción de la estructura de los datos y la organización de los archivos, diseño o layout de terminal de presentación visual, interfaz o T.P.V. terminal de pantalla visual (V.D.T. o visual display terminal), de impresoras y otros. Será aquí en donde en definitiva se van a definir el cómo se van a encontrar almacenados los

datos y los medios en que ellos estarán cargados (archivos en discos duros, discos extraíbles, etc.).

b) El **contenido** de la Base de Datos se conformará por datos o información del más diverso tipo (texto, cifras, alfanuméricos).

c) Los elementos necesarios para el adecuado funcionamiento del sistema: conjunto de **dispositivos computacionales** que serán utilizados para ingresar, almacenar y procesar los datos, involucrándose en ello la participación de una o más **personas** que manejarán estas herramientas técnicas, conjuntamente con una serie de **programas y procedimientos** que serán supervisados por diversos medios de control, para conseguir adecuadamente que el modelo de ordenamiento elegido finalmente funcione, esto es, quede operativo.

Respecto de los dispositivos computacionales si bien puede que no se perciba directamente su importancia a la hora de ingresar, almacenar y procesar los datos pueden tener gran trascendencia al momento de realizar tales labores y posteriormente al hacer funcionar las Bases de Datos en red:

a) **Equipos de comunicación de datos:** Son dispositivos de comunicaciones que se encargan de establecer, mantener y terminar una sesión en una red. Pueden además convertir las señales para las transmisiones y es típicamente un módem.

b) **Servidor de Base de Datos:** Es una computadora habitualmente conectada a una red de área local que está dedicada a realizar labores de almacenamiento y recuperación de contenidos que se manejen en una aplicación cliente.

I.3. Distinción básica de los tipos de Bases de Datos

Las Bases de Datos computacionales parten de sistemas tradicionales, lo que hace se identifiquen en muchos elementos; sin embargo, se expresan en medios de información distintos.

a) **Bases de Datos No Computacionales o Tradicionales:** son aquellas colecciones o compilaciones que se conforman por procedimientos mecánicos que no involucran el uso de herramientas computacionales y que se contienen en un soporte físico que tampoco es operado por esa vía, tales como el archivo contenido en un kardex de una oficina, una guía o directorio telefónico, un fichero bibliográfico, etc.

b) **Bases de Datos Computacionales:** es una colección de datos interrelacionados que puede ser usada simultáneamente por más de un programa o una aplicación.[7]

Debe aclararse que si bien la forma en que operan tanto el organizador de una Base de Datos Computacional como el de una no Computacional son virtualmente idénticas, ello no significa que no se presenten importantes diferencias entre ambos tipos de producciones:

1. El **medio de expresión** elegido por las bases de datos computacionales y las compilaciones tradicionales difiere radicalmente. Los computadores almacenan los BITS de los datos electromagnéticamente en cintas magnéticas, disquetes removibles, discos ópticos u otro tipo de soportes similares o también pueden hacerlo a través de los dispositivos que van al interior del hardware y que se encargan del manejo de los distintos tipos de

[7] Villanueva Lara, Julio. Ob. citada. Página 127.

memoria disponibles (de acceso aleatorio o RAM y de sólo lectura o ROM). Pero, una vez almacenados, los datos no pueden ser aún percibidos por el ojo humano, ya que antes, la computadora debe interpretar y traducir dicha información a un lenguaje comprensible por el ser humano. En contraste, la información que se puede contener en una compilación tradicional puede percibirse, en la generalidad de los casos, sin la ayuda de una máquina.

2. La **ubicación, estructura y forma de los datos en cada uno de los medios de expresión.** En la producción de una compilación tradicional, el organizador de ésta ordena la información de una forma pensada como la más fácil de entender y usar para cualquier persona. Típicamente, el resultado de este esfuerzo se manifiesta en una forma sencilla de presentar el producto en forma de publicación impresa. La ubicación de los datos será determinante de la utilidad y grado de aceptación que tendrá la compilación entre los usuarios, por lo tanto, la facilidad en el manejo de la información, según la disposición que ella tenga en el producto final, es siempre el principal conflicto que debe enfrentar el compilador, ya que la facilidad en torno al acceso a los datos es relevante para quien consulta; marca la diferencia entre una compilación comercialmente rentable para su organizador y una que no tendrá dicho valor. En el caso de las bases de datos computacionales, la situación del organizador no se ve tan sujeta a la simple disposición de los datos, puesto que, quien actúa como tal en estas producciones dejará la determinación de la manera en la que los datos serán vistos a discreción del usuario, dedicándose a establecer los límites mínimos y máximos de operatividad que dicha persona podrá realizar según las características del software que sirva para el manejo de la información.

En definitiva, **los datos en una Base de Datos computacional normalmente se fijan en un medio magnético o en la memoria de la**

computadora, en una disposición distinta a la forma en la que será vista por el usuario.

Sin perjuicio de lo anterior, la diferencia entre la forma en la que la base de datos es vista por el usuario, en cuanto a la disposición de la información y la forma en que esos datos han sido memorizados, no significa que la estructura actual del sistema de base de datos no sea importante.

Al contrario, la ubicación "física" que actualmente presenten los datos y que permitan su recuperación en grandes y complejas bases de datos, es de la mayor importancia. La ubicación de los datos en un disquete, disco compacto grabable (CD-R) o en la memoria interna de la computadora, determinará el tiempo que se va a necesitar para acceder al dato buscado.

Los diseñadores de Bases de Datos y los programadores de software señalan que la disposición de los datos en un sistema de bases de datos se realiza a dos niveles:

Un nivel "lógico"; Los datos conceptualmente según esta disposición reflejarán las relaciones uni, bi o tri-dimensionales existentes entre los diversos campos, registros y archivos de la base de datos. Para entender la disposición lógica debe permitirse al programador la planificación del acceso y recuperación del software que sea utilizado con la base de datos.

Por otra parte, la disposición física de los datos se referirá a la estructura actual de la información o a la ubicación existente dentro del nivel de almacenamiento desarrollado en el medio.

En un sistema de base de datos que cuente con un solo archivo, por su pequeñez, hará que esta última distinción sea poco importante; pero en un

sistema que presente una gran cantidad de archivos la actual disposición de sus registros puede diferir de una forma importante si se revisa la disposición lógica; y la ordenación física de los registros por tanto la localización actual de un registro puede entonces ser determinada por el diseñador de la base de datos para la optimización del sistema.

Aquí será determinante la labor de quien se desempeñe como el diseñador de la base de datos (que puede ser el mismo organizador u otra persona) quien deberá buscar reducir al máximo el tiempo de acceso al sistema de información. Será muy importante para estos efectos, tratar de conocer con anticipación el perfil del grupo de usuarios que requerirán los servicios del sistema, esto es, saber cuales son las áreas que pueden constituir mayor interés para la futura realización de sus consultas, debiendo minimizarse prioritariamente en dichas zonas del sistema el tiempo de acceso a los datos contenidos en ella, "el criterio que deberá ocupar el organizador en estos casos es el de reconocer los distintos *campos* en que deberá concentrar mayor trabajo, ya que definiendo éstos y mejorando su contenido facilitará el posterior procesamiento de la información"[8] así se lograrán satisfacer los tiempos de respuesta que el usuario necesita.

Si el diseñador de la Base de Datos logra minimizar el tiempo de acceso que ocuparán los usuarios para conocer la información contenida en ella, de una forma consistente aumentará el interés comercial sobre el producto final, y a su vez, ganará un mayor prestigio dentro del segmento profesional de los fabricantes de Bases de Datos. Este ahorro en el tiempo de acceso a la información por el usuario, también puede lograrse si se utilizan mecanismos o técnicas adecuadas para conseguir la compresión de los datos que conforman la información utilizada por el sistema.

[8] Freedman, Alan. Diccionario de Computación. Software de The Computer Language Company Inc.Versión 5.2 (1981-1993).

En síntesis, una estructura de almacenamiento de los datos si es bien diseñada en cuanto a las necesidades de acopio de información, generará la reducción en los costos del funcionamiento del sistema de bases de datos y lo hará un producto más atractivo comercialmente.

I.4. Clasificación de los distintos tipos de Bases de Datos Computacionales

Aquí se enumeran los distintos tipos de Bases de Datos de acuerdo a su estructura y contenido.

1. **De acuerdo al diseño de su estructura:** Cuando se realiza el diseño de una Base de Datos computacional se pueden seguir determinados modelos, enfoques o arquitecturas, los cuales reflejarán la forma en que los datos o la información se organiza combinando velocidad de acceso, flexibilidad y facilidad de uso. Fundamentalmente, los modelos de estructuras a seguir son los siguientes:[9]

    a) **Bases de Datos de Modelo Jerárquico:** Este es un modelo del tipo "de arriba hacia abajo", esto es, los registros se enlazan entre sí como un verdadero diagrama de organización, contemplándose que determinados registros serán de nivel superior denominándolos "registros padres", cada uno de ellos tendrá uno o más "registros hijos" en un nivel inferior. Se producirá entonces una estructura en que cada "registro hijo" sólo tiene un "registro padre", pero puede tener otros "hijos" a su vez.

[9] Gillenson, Mark L..Introducción a las Bases de Datos. Páginas 98 y ss., 109 y ss., 155 y ss., 185 y ss., 241 y ss. Ed. McGraw-Hill. México, S.A. de C.V.1988.

De acuerdo a este modelo, para procesar registros de la Base de Datos se suele consultar al "registro padre" de mayor nivel y se continúa hacia abajo a través de los demás componentes como en cualquier jerarquía hasta ubicar el registro deseado. De esta forma los datos quedan organizados en niveles jerárquicos, en el que cada uno constituye una subcategoría.

b) **Bases de Datos de Modelo de Red o Malla:** En este tipo de Bases de Datos varios registros están lógicamente relacionados y cualquiera de ellos puede servir como punto de entrada. Con este enfoque un solo dato puede apuntar a muchos otros y, a su vez, puede ser apuntado por otros datos.

Las Bases de Datos ya sea que sigan el enfoque jerárquico o el de red, a menudo se denominan "sistemas de navegación", ello porque en ambos casos los datos serán almacenados como registros de distintos tipos que estarán interconectados por medio de apuntadores de dirección. Los programadores que optan por utilizar estas arquitecturas, pueden "navegar" a través de sus estructuras; esto significa, por ejemplo que una llamada de programa a la Base de Datos puede crearse para que encuentre una **Sociedad de Gestión de Derechos de Autor** específica, encomendarle que ubique una determinada **obra intelectual** que se encuentre dentro del **repertorio** de obras por las cuales esa sociedad recaude derechos de autor o conexos y arroje una lista con los nombres, direcciones y datos de contactos de las personas autorizadas por contratos de **licencia** para utilizar dicha obra.

c) **Bases de Datos de Modelo por Relación:** Según este enfoque se utilizan todos los datos almacenados en la forma de filas y columnas

(las filas son los registros y las columnas los campos, bits asignados a cada dato). Físicamente, esto es lo mismo que el tradicional archivo simple; sin embargo existen reglas específicas para organizarlo, que conforman el concepto por relación. El objetivo de este modelo, es facilitar la petición de los datos, de acuerdo con las necesidades específicas, respecto de las estructuras tradicionales de Bases de Datos Jerárquicas o las de Red. Una de las principales características de este tipo de arquitectura, es su capacidad de generar un nuevo archivo con los datos provenientes de dos archivos por relación.

d) **Bases de Datos de Modelo Seudorrelacional:** En "este tipo de Base de Datos se combina la estructura de archivo plano de la Base de Datos Relacional con el concepto de lograr la integración de los datos con algún tipo de construcción física prealmacenada, como sucede en los sistemas navegacionales"[10].

2. **De acuerdo a su contenido:** se atiende aquí a lo que puede conformar el cuerpo del documento de una Base de Datos computacional, lográndose distinguir las siguientes:

a) **Bases de Datos Fuente:** son aquellas que recogen el dato o el texto completo de la información original. Suelen subdividirse en:

- **Numéricas o factuales:** recogen y proporcionan datos directamente utilizables tales como índices relativos a mercados cambiarios, bolsas de comercio, series estadísticas, archivos de parámetros, etc.

[10] De toda la bibliografía consultada, este modelo pudo encontrarse sólo en la obra de Gillenson, Mark.Ob citada. Página 98.

- **Textuales:** recogen, seleccionan, describen y analizan documentos primarios de todo tipo. Los datos catalográficos se complementan con descriptores y generalmente resúmenes más o menos amplios.

b) **Bases de Datos Referenciales:** son aquellas que se remiten a otra fuente para obtener los datos o la información. Entre estas a su vez se pueden distinguir:

- **Bibliográficas:** contienen sólo información referencial de literatura impresa (libros, artículos de revistas, patentes, etc.) o de otro material no impreso (discos, cassettes, medios audiovisuales, etc.), pero sin realizar ninguna trascripción, adaptación o resumen de las obras que se citan (por ejemplo en el caso de obras literarias suelen indicar: autor, título, editorial, lugar de publicación, etc.), y tienen por finalidad que el usuario conozca de la existencia y ubicación de tales obras.

- **De Directorio:** contienen referencias respecto de datos o información que no ha sido publicada (por ejemplo: clasificación o ubicación de empresas, nóminas de profesionales por especialidad, etc.).

c) **Bases de Datos Mixtas:** son aquellas que presentan simultáneamente las características de las Bases de Datos Fuente y Referenciales.

I.5. Sistema de Manejo o de Gestión de Bases de Datos (S.G.B.D. o D.B.M.S.)

Encontramos dentro de estos elementos uno que podría considerarse como la pieza más fundamental de todo este conjunto (sin la cual no podríamos determinar cómo se arma nuestro rompecabezas...), este se denomina **Sistema de Manejo o de Gestión de Bases de Datos (S.M.B.D. o S.G.B.D.).** Se conoce bajo este concepto al software que especifica la forma en que los datos pueden estructurarse (organización, almacenamiento), controla todos los accesos a éstos (recuperación, seguridad e integridad de la información) y proporciona algunos otros servicios esenciales de datos. La estructura de datos, en algunos casos de acuerdo con la forma en que esa información se manipula, debe proporcionarse para obtener integración en los datos, reducir su redundancia y para expresar sus relaciones múltiples.

Cuando se usa un S.M.B.D. o S.G.B.D., los sistemas de información pueden transformarse o actualizarse más fácilmente a medida que cambien los requerimientos de la organización. Nuevas categorías de datos pueden agregarse a la Base de Datos sin dañar el sistema existente.

En síntesis, cabe señalar que la Base de Datos es creada, consultada y actualizada por medio de un software especializado, el cual se denomina Sistema de Manejo o de Gestión de Bases de Datos (S.M.B.D. o S.G.B.D.). Dicho elemento presenta la particularidad de ser un "intermediario", ya que actúa como una interfaz, en el medio ambiente de procesamiento de datos, entre los archivos de datos y las personas que buscan los datos de esos archivos. Esto significa que todo acceso a los archivos para la realización de una búsqueda o de una consulta, ya sea que ellas se hagan desde el teclado de una terminal de exhibición o por

medio de informes impresos, deberán pasar (tanto las solicitudes como las respuestas a estas operaciones) a través del S.M.B.D.

Entre las **principales características que presenta un S.M.B.D. o S.G.B.D.** se encuentran:

- **Seguridad e integridad de los datos**: la seguridad de los datos evita que usuarios no autorizados vean o modifiquen la Base de Datos. Por ejemplo, una Base de Datos puede contener todos los datos sobre un determinado autor, sin embargo, un grupo de usuarios puede estar autorizado para ver sólo los datos relativos a sus obras musicales, mientras que a otros les estará permitido acceder solamente a información relativa a aquellas obras musicales que han sido incluidas en una obra audiovisual. El S.M.B.D. puede mantener la integridad de la Base de Datos no permitiéndole a más de un usuario actualizar el mismo registro al mismo tiempo. El S.M.B.D. puede impedir registros duplicados en la Base de Datos; por ejemplo, no pueden ser introducidas dos obras distintas con el mismo número o código identificador internacional.

- **Consulta, interrogación interactiva**: casi todos los S.M.B.D. proveen lenguajes de consulta (interrogación) y los escritores de informes que permiten a los usuarios interrogar interactivamente a la Base de Datos y analizar los datos. Esta es una de las más importantes características de un S.M.B.D., ya que permite a los usuarios obtener información gerencial inmediatamente.

- **Entrada de datos y actualización interactiva**: muchos S.M.B.D. proveen una manera de introducir y editar datos interactivamente, permitiendo a los usuarios manejar Bases de Datos Personales. Sin embargo, la operación interactiva no deja un rastro de revisión de acciones

y no brinda los tipos de controles necesarios en una organización de multiusuarios. Estos controles sólo están disponibles cuando los programas de aplicación están especialmente diseñados para cada función de entrada de datos y actualización; es decir guardan un registro de quien y cuando realizó cierto tipo de cambios en la Base de Datos.

- **Independencia de los datos**: con S.M.B.D., los detalles de la organización de los datos no necesitan incorporarse a cada programa de aplicación. El programa de aplicación le pide al S.M.B.D. los datos mediante el nombre de campo; por ejemplo, un equivalente codificado de "deme nombre de autor y saldo a favor" sería enviado al S.M.B.D. Sin un S.M.B.D., el programador deberá reservar espacio para la estructura completa del registro en el programa. Cualquier cambio en la estructura de los datos que quisiera llevarse a cabo, en tales casos, haría necesario tener que realizar cambios en todos los programas de aplicación.

Para concluir esta breve revisión de los elementos necesarios para el adecuado funcionamiento del sistema, no podríamos dejar de hacer mención a un dispositivo que podría tener una determinada trascendencia al ingresar, almacenar y procesar los datos y su cada vez menos eventual manejo por una Base de Datos conectada en una red. Este se conoce como **máquina de Base de Datos o Servidor** una computadora especialmente diseñada para contener una Base de Datos y ejecutar solamente un S.M.B.D. y software relacionado. Este equipamiento tiene un gran número de funciones de S.M.B.D. construidas dentro del hardware y también proporciona técnicas especiales para el acceso a discos que contienen las Bases de Datos, tales como el uso simultáneo de múltiples procesadores para búsquedas de alta velocidad.

Su importancia es evidente, ya que si consideramos que ellas al ser conectadas a una o más microcomputadoras o servidores por medio de un canal

de alta velocidad, permiten la utilización del contenido de las Bases de Datos en ambientes donde se procesan grandes volúmenes de transacciones, a su vez hacen posible concretar el sueño de interconectar grandes cúmulos de información tanto a nivel nacional como internacional.

I.6. Una nueva forma de aproximarse a la información

La industria de la informática prosigue con regularidad su penetración en los sectores de servicios, industrial y agrícola y confirma su papel como motor en la competitividad de las empresas. Prescindir hoy en día de la informática puede suponer volver la espalda al crecimiento económico e incluso retroceder.

Asistimos a un macrofenómeno cultural, que es la consecuencia de una mejor percepción de las posibilidades de la informática.

La informática no se percibe solamente por su parte visible y material *–el hardware–*, sino también por su parte oculta e inmaterial *–el software–*.

Esta toma de conciencia se está desarrollando con rapidez en las grandes empresas, en las que los presupuestos destinados a ampliar el software crecen vertiginosamente, mientras los costos de infraestructura material siguen el camino contrario con la misma velocidad. La tendencia irreversible de las cifras no se percibe aún en toda su importancia en el mundo de la pequeña y mediana empresa.

Hoy en día, tanto para las empresas que ya han alcanzado un cierto grado de experiencia en la informática como para las que carecen de ella, se presenta una *nueva forma de aproximación a ella.* En lugar de crear el software propio, se orientan hacia la compra de software estandarizado o software comercial.

Gracias a esta aproximación pueden las empresas conseguir considerables ahorros y asegurarse un mayor dominio de su informática (no necesitan hacer programas ni aprender ningún lenguaje informático).

El principio esencial en el que se fundamenta esta aproximación está contenido en la idea de que es preferible elegir el software antes de elegir el hardware. Dicho en otras palabras, dar preferencia a la *función* (los programas) sobre el *órgano material* (computadora). De esta manera es como surge una nueva idea de la informática como se muestra en la Fig. I.1.

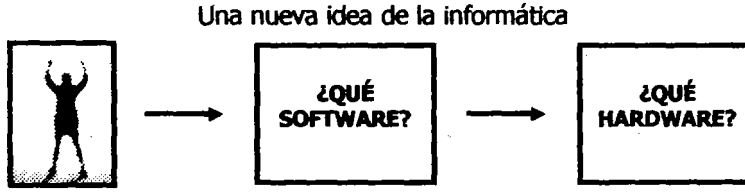

Fig. I.1. El mercado de paquetes de software experimenta un crecimiento explosivo.

Esta aproximación, puesta en práctica ya con éxito por empresas de todas las dimensiones, se opone radicalmente al enfoque técnico tradicional del proceso de informatización. Insiste en la necesidad de un método funcional (abogando claramente por la solución que ofrece el software y, más concretamente, el software comercial).

Las consideraciones técnicas (la elección del equipo) no por eso quedan olvidadas, pero sí relegadas a un segundo plano, y no se consideran ya como elemento primordial del proceso de informatización.

Sin embargo, ciertas aplicaciones de la informática muy específicas en el campo industrial (control de procesos industriales, etc.) se apartan de esta idea e imponen ciertas características materiales (fiabilidad, rapidez, etc.). ¿Se debe

comenzar obligatoriamente la introducción de la informática en una empresa por la compra del equipo? Cualquiera que sea el nivel técnico alcanzado hoy por las computadoras, no será perceptible más que en función de los programas de aplicación bien concebidos y fiables, que le sean asignados.

Dejarse llevar por las características materiales del equipo conduce a introducir nuevas limitaciones en la elección del software: compatibilidad del sistema operativo, capacidad de memoria exigida, capacidad de los equipos periféricos a instalar, etc. El usuario corre el riesgo de no encontrar en el mercado paquetes de software que realicen las funciones que exige el material elegido.

La experiencia muestra lo precario de los resultados obtenidos en la práctica normal, que convierte frecuentemente al software en el pariente pobre del proceso de informatización.

Muchas empresas han sufrido y sufren aún las consecuencias de un enfoque excesivamente técnico, realizado en detrimento de una madura reflexión sobre los *campos de gestión* que se deben informatizar.

Son las responsables quienes deben definir sus necesidades y, en consecuencia, elegir el paquete de software que mejor se adapte a tales necesidades. No se deben dejar atar previamente por las consideraciones técnicas relativas al hardware (algunas excepciones, evocadas ya en parte, confirman esta regla general).

En la elección del software estandarizado se aplica en su conjunto a la informática, aunque los aspectos materiales (hardware) pueden pesar mucho más en los procesos de elección informáticos en razón de problemas de:

- Compatibilidad entre equipos
- Rendimiento
- Redes
- Seguridad
- Aplicaciones complejas ya existentes

Toda esta reflexión hace recordar un antiguo refrán "lo barato sale caro" y nuestro caso no es la excepción.

En muchas ocasiones surge que al iniciar un proyecto se cuenta ya con un software en el que se esta trabajando, en éste caso habrán de estudiarse detenidamente las capacidades de los mismos antes de iniciar cualquier carga de software, sin embargo dentro de casi todo proyecto debe tomarse en cuenta la adquisición o mejora de uno o varios equipos ya que nuevos sistemas requieren en su mayoría computadoras de superiores capacidades y todo gasto hecho debe considerarse como parte de un avance y que los gastos efectuados están bien justificados para un progreso en los procesos de la empresa.

I.7. Naturaleza del software

El software en nuestros tiempos se divide en dos campos: el software comercial y el hecho a la medida, el cual depende en muchas ocasiones de software libre de tal modo que para la implementación de un sistema es importante conocer los aciertos y deficiencias que estos representan.

Software particular

«Software particular» es aquel que la empresa requiere para el mejor desarrollo de alguna actividad propia de ella. Este tipo de software suele ser

diseñado por el propio personal de la empresa o bien mediante la contratación de personal especialmente capacitado y encargado únicamente del desarrollo de la solución.

Actualmente existen dos grandes caminos para el desarrollo de este tipo de aplicaciones: es mediante el uso de herramientas de diseño y programación de uso comercial, que es como acostumbramos trabajar; y una nueva tendencia de desarrollo que surge del trabajo en equipo y la libertad de mejorar aplicaciones que es el software libre.

Software libre

El «software libre» es una cuestión de libertad, no de precio. Para comprender este concepto, debemos pensar en la acepción de libre como en «libertad de expresión».

Cuando se hace mención a software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software, es decir:

- Libertad para ejecutar el programa sea cual sea nuestro propósito.

- Libertad para estudiar el funcionamiento del programa y adaptarlo a tus necesidades —el acceso al código fuente es condición indispensable para esto—.

- Libertad para redistribuir copias y ayudar así a tu vecino.

- Libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad —el acceso al código fuente es condición indispensable para esto—.

Software libre es cualquier programa cuyos usuarios gocen de estas libertades. De modo que deberías ser libre de redistribuir copias con o sin modificaciones, de forma gratuita o cobrando por su distribución, a cualquiera y en cualquier lugar.

La libertad para utilizar un programa significa que cualquier individuo u organización podrán ejecutarlo desde cualquier sistema informático, con cualquier fin y sin la obligación de comunicárselo subsiguientemente ni al desarrollador ni a ninguna entidad en concreto.

El software libre no significa que sea «no comercial». Cualquier programa libre estará disponible para su uso, desarrollo y distribución comercial. El desarrollo comercial del software libre ha dejado de ser excepcional y de hecho ese software libre comercial es muy importante.

Cuando se habla de software libre, es preferible evitar expresiones como «regalar» o «gratis», porque entonces caeremos en el error de interpretarlo como una mera cuestión de precio y no de libertad. Términos de uso frecuente que representan de mejor forma a la «piratería».

En las Tablas I.1 que se muestran a continuación se mencionan algunas alternativas a los programas comerciales, las cuales no siempre son una mejor opción pues depende de la habilidad y conocimientos del usuario, para que puedan funcionar correctamente en su computadora, además de que los archivos que se generan con el software libre no son totalmente compatibles con las aplicaciones de software comercial.

Sistemas Operativos	
Software comercial	Software libre
Windows XP Pro y Home	Red Hat SuSE Debian
Windows Server 2003	Red Hat Enterprise Linux AS

Suites de oficina	
Software comercial	Software libre
Office 2003 WordPerfect Office 12	Open Office AbiWord

Edición de Imágenes	
Software comercial	Software libre
PhotoshopPhoto Paint	GIMP

Tablas I.1. Alternativas a programas comerciales de software

La mayoría de este tipo de software puede ser descargadas gratuitamente en desde sus respectivos sitios en la Internet.

Software comercial

Se puede definir el software comercial del siguiente modo: *"Conjunto coherente e independiente constituido por programas, servicios y soportes de manipulación de información y de documentación, concebido para la realización de tratamientos informáticos estándar, difundido con carácter comercial y utilizable de manera autónoma por el usuario tras una breve fase de formación e implantación".*

El paquete de software es un producto concebido por las empresas de ingeniería con la idea de conseguir una amplia difusión del mismo. El objetivo de los proveedores es conseguir un considerable volumen de negocios como consecuencia de la comercialización del producto en un segmento de mercado (contabilidad, comercio al por mayor, etc.). Los productos están concebidos en términos de:

- **Aproximación a las necesidades de la empresa**, es decir, se lanzan al mercado con la vista puesta en un conjunto de empresas consumidoras.

- **Calidad técnica industrial** (establecimiento de una serie de parámetros, de lenguaje de alto nivel, etc.).

- **Economía de escala** (cuanto mayor sea el segmento de mercado, tanto más podrá el distribuidor reducir el precio «unitario» de un paquete de software).

La ventaja de las empresas consiste en beneficiarse de un *producto acabado* (fiable, experimentado, generalizado y documentado) disponible con carácter inmediato y concebido por un equipo humano que incluye múltiples disciplinas (gestores y técnicos especializados en temas de software).

Las empresas valoran el software que tratan de adquirir a la vista de la documentación (funcional y técnica) y de las demostraciones de utilización y mantenimiento con usuarios de los paquetes de software preseleccionados.

El precio del producto viene indicado en el catálogo junto con las funciones y opciones disponibles. Gracias a esto, la empresa puede determinar sus inversiones informáticas en software. De tal modo que la adquisición de un

software comercial no es tarea fácil ya que se tiene que tener en cuenta una serie de consideraciones a seguir como se muestra en la Fig. I.2.

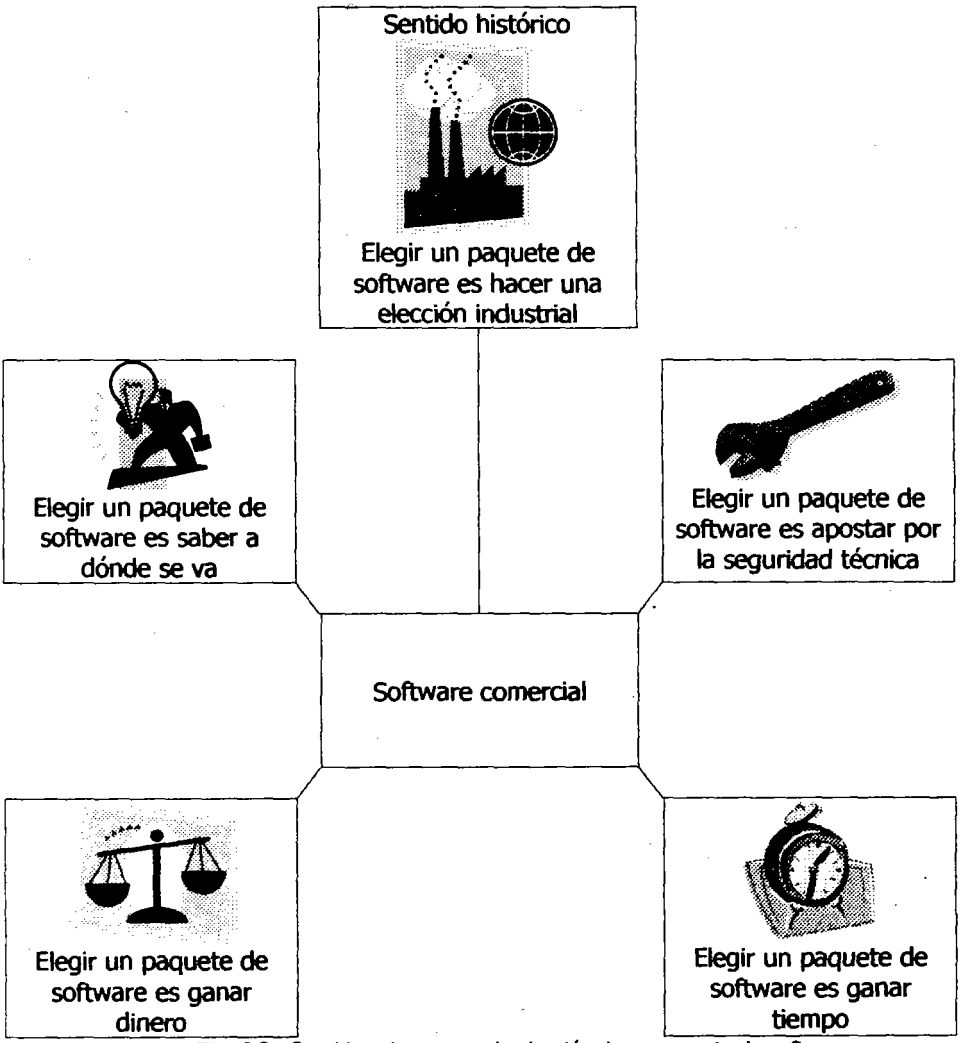

Fig. I.2. Consideraciones para la elección de un paquete de software

Por otro lado, los contratos de mantenimiento liberan a la empresa de los problemas de conservación del software.

A igualdad de funciones, un paquete de software es más barato que un software «hecho en casa», (no tiene costos de desarrollo y los de mantenimiento están aligerados).

El software estandarizado, debido frecuentemente a su nivel tecnológico y a la visión «panorámica» de sus creadores, aporta ideas innovadoras sobre el proceso de informatización, sobre las funciones a automatizar y sobre la nueva organización que se quiere concebir.

El software estandarizado aporta una importante y provechosa experiencia industrial.

En la Tabla I.2 podemos observar algunos elementos de comparación entre un software «hecho en casa» y un paquete de software.

	Software particular	**Software estandarizado**
Adaptación a las necesidades	Responde, en principio, a las necesidades de una aplicación concreta	Puede presentarse la dificultad de encontrar un paquete de software que corresponda estrictamente a la necesidades*
Origen	El creador pertenece a la empresa**	El creador es ajeno a la empresa (esto puede significar un aspecto positivo***)
Objetivo	Responder a unas necesidades actuales, limitadas y especificas	Responde a las necesidades de un amplio abanico de empresas
Organización	No pone en tela de juicio la organización existente	Puede provocar una modificación de la organización (lo que frecuentemente es una ventaja)

Plazo de entrega	Plazo de realización difícil de establecer (entre 6 meses y dos años)	Es inmediatamente utilizable
Costo y grado de control del precio	El costo depende de la complejidad de mismo.	Costo menor, ya que su distribución es masiva. Se puede conocer con precisión
Garantía en las especificaciones	No hay garantía real sobre el resultado; solamente puede juzgarse después de elaborado el programa	Funcionalidad descrita en el catalogo. Producto acabado, suministrado con la correspondiente documentación. Producto industrial
Evolutividad de la especificaciones	Existe una mayor dificultad de hacer evolucionar el software, a menos que se utilice un equipo de especialistas.	Normalmente no hay inconveniente alguno para la migración del un sistema a otro gracias a que se tienes utilizan normas estrictas
Mantenimiento	Regularmente resulta de mayor complejidad pues debe de estudiarse en mayor profundidad	Puede realizarlo el propio proveedor con tarifas fijadas
* Cada vez es menos cierto, ya que este tipo de mercado está experimentando un desarrollo considerable. ** Si es que no recurre a la ayuda de una consultaría de servicios informáticos. *** No existe la tentación de variar continuamente el paquete, y el producto ya documentado corresponde con la versión en explotación.		

Tabla I.2. Comparativa entre el software hecho en casa contra un paquete comercial.

"Las principales ventajas de un paquete de software residen en su precio y en su disponibilidad casi inmediata".[11]

En resumen, la diferencia entre el software propio y el software estandarizado es la misma que puede existir entre un producto único y un producto fabricado en serie; en respuesta a ellos se comienzan a concebir

[11] Tanguy Jacques. Elección y compra del software de gestión. Paginas 7. Ed. Deusto. Barraicúa, Bilbao. 1984.

sistemas que ofrezcan las ventajas del software estandarizado y a la medida con productos estandarizados con suficiente adaptabilidad.

I.8. Los criterios de selección

Siguiendo el método para llevar a cabo un proceso de informatización, la empresa va a explorar el mercado de paquetes de software para entresacar de él los productos que cubren la función a informatizar. La empresa se documentará sobre el mercado y sobre los medios para comprar productos. Su análisis se apoyará en determinados criterios a los que asignará un peso específico según la importancia concedida a los distintos aspectos del problema (funcionales, precios, imagen del proveedor...).

Los criterios de elección pueden agruparse en tres familias:

- Criterios funcionales
- Criterios técnicos
- Criterios comerciales

I.8.1. Criterios funcionales

El paquete debe poder resolver las funciones que desea informatizar el usuario. El análisis se funda en el contenido del estudio previo y de especificaciones que sirven de marco de referencia. El examen de los criterios funcionales es fundamental. El paquete se puede considerar como una «caja negra», perceptible únicamente por sus entradas y salidas, es decir, por sus funciones. La primera característica de un producto es el conjunto de objetivos que permite alcanzar en el plano funcional:

- ¿Qué puede admitir a la entrada?
- ¿Qué se obtiene a la salida?

Estas dos preguntas conllevan toda una serie de nuevas preguntas relativas a sus posibilidades y límites, a la flexibilidad del lenguaje de diálogo que permite comunicarse con el paquete de software, a su capacidad de modulación funcional, a su juego de parámetros, etc.

Funcionalidad del paquete

Conocer la funcionalidad de un paquete de software nos expresa y apoya a vislumbrar los alcances que podremos lograr al utilizarlo tomando en cuenta desde lo limitado de que puedan sus funciones hasta lo complejo que sus procesos.

Por ejemplo: un paquete de contabilidad puede no referirse más que a contabilidad general, otro propondrá una gestión completa de contabilidad de clientes y proveedores, etc.

Es preciso desconfiar de los productos de aplicación que se presentan como universales y que realizan un conjunto ilimitado de funciones. Originan, a veces, mal entendidos entre cliente y proveedor, pues hacen pensar que el producto está suficientemente abierto en el plano funcional como para adaptarse a todo tipo de problemas. Esto puede ser cierto, pero la experiencia demuestra que el producto milagroso no existe. Un producto está destinado a cubrir un determinado campo, y aunque se puede utilizar en otros dominios, es preciso considerar que, sin duda, estará menos adaptado en estos últimos casos. A menudo es aconsejable adoptar un producto destinado estrictamente a la función de informatizar, ya que estará mejor equilibrado y al menos será satisfactorio en el desempeño de su función principal, antes que adoptar un producto de características más amplias, pero que puede llegar a presentar puntos débiles en muchos aspectos funcionales.

El análisis del producto lleva también a otra consideración: Se debe buscar un producto «hecho a medida» o se pueden cerrar los ojos y adoptar un producto por debajo de las expectativas, o incluso, tras una reflexión prospectiva, orientarse inmediatamente hacia un producto funcionalmente sobredimensionado.

En general, los productos que responden a una determinada necesidad, ¿cubren los objetivos funcionales entre un 80 y un 120% (o más)?

A primera vista, el ideal, a corto plazo, consiste en adoptar un producto que cubra el 100% de las necesidades a satisfacer, pero, ¿es éste realmente un buen razonamiento? ¿No será quizás mejor adoptar un producto por debajo de las expectativas antes que un producto que se adapte a la función en estudio, pero que presente lagunas? (lenguaje de diálogo limitado, compatibilidad escasa, etc.). Ciertamente, hoy se puede sentir la tentación de adoptar una tercera solución: el producto sobredimensionado que cubra el 120% de las necesidades (¡o más aún!).

Un producto sobredimensionado, si bien es normalmente más caro, presenta ventajas incontestables:

- **Permite esperar.** No todas sus funciones se utilizarán al principio; solamente se activarán las que en dicho momento sean útiles. Se hablará así de un aumento de carga funcional de la aplicación según las necesidades de la misma.

- **Contempla de lejos el delicado problema de la compatibilidad.** Evitando así un ciclo demasiado corto de desplazamiento de las aplicaciones.

- **Contribuye a hacer vivir la aplicación informatizada a medida que pasa el tiempo.** Si la aplicación se enriquece progresivamente con nuevos elementos funcionales, puede ser que la empresa, en base a las funciones del producto, condicione progresivamente su aplicación para que se ajuste a dicho producto. Se trata de realizar, desde ese momento, ajustes funcionales que serán la base para una buena aplicación del producto.

Lo importante es que el producto está ya en la empresa y que la existencia de funciones aún sin explotar permite dominar el sentido y la capacidad de desarrollo futuro de la aplicación.

Los criterios funcionales se pueden agrupar en cuatro familias:

a) Cobertura funcional del producto (comprendiendo la gestión de ficheros y los procedimientos)
b) Lenguaje de diálogo
c) Controles y procedimientos de corrección
d) Estados y formatos de pantalla

Cobertura funcional del producto

Para cuantificar la cobertura funcional de un paquete de software es preciso contestarnos algunas preguntas de importancia:

- ¿Realiza el paquete de software un 80%, un 100 % o un 120% de las funciones que se necesita realizar? Si no se cumple esta condición, es mejor rechazar el paquete por ser funcionalmente inadecuado.
- ¿Presenta el producto la posibilidad de modulación funcional? Por ejemplo tratando una gestión comercial ¿la gestión de pedidos y facturación van por separado, pero dentro del mismo paquete?

- ¿Se integra el producto dentro de una gama de paquetes? Es importante para mantener viva la aplicación, pero no es un método que deba seguirse a ciegas. Por ejemplo ¿nuestro paquete podría importar sus tablas a un procesador de textos u hoja de cálculo distintas a la aplicación misma o todos sus datos son excusitos a ella?
- ¿Posee funciones exclusivas para ciertos tipos de cálculo, de investigación de documentos, clasificación selectiva, generación de estados, etc.? Es decir, contiene herramientas de uso más particulares a nuestras necesidades como lo podría ser algún tipo de búsqueda u ordenación de los datos.
- La combinación de parámetros del producto, ¿permite responder exactamente al desarrollo de la aplicación? (Por ejemplo: en el caso de que varíe el impuesto sobre el valor añadido.) Es preciso prestar atención a los productos demasiado rígidos y poco evolutivos, satisfactorios al principio, pero que se abandonarán por falta de la capacidad de evolución que exige la aplicación.

Conocer estas respuestas nos da una mayor seguridad de adquirir un producto que a la larga nos de facilidad de expansión y mayor vigencia productiva.

El lenguaje de diálogo

Un paquete de software, para ser satisfactorio, debe respetar el lenguaje del usuario y proponer fórmulas de lenguaje claras, sin ambigüedad y potentes.

El lenguaje de diálogo traduce las manipulaciones diarias realizadas delante del terminal (entradas de asientos de contabilidad, por ejemplo). Debe ser de fácil comprensión y uso, a riesgo si no, de quedar infrautilizado o incluso abandonado, por más que las funciones sean satisfactorias.

La concatenación de menús y la disposición de las informaciones en pantalla difieren de un paquete a otro, tanto para funciones idénticas como para funciones distintas. En ciertos casos, un lenguaje demasiado articulado (por ejemplo, el lenguaje usado en informática, donde se dice no sólo lo que se quiere hace, sino también cómo se quiere hacer) o un lenguaje demasiado pesado (un número excesivo de menús insertados) pueden hacer que su utilización regular resulte fatigosa. La potencia de un lenguaje se representa por la capacidad de decir lo que se quiere en pocas etapas a través de módulos u operadores semánticos. Un lenguaje demasiado rico puede ser tan peligroso como uno demasiado pobre.

Ciertos paquetes de software ofrecen máscaras de pantallas prefijadas, mientras otros ofrecen algunas máscaras estandarizadas susceptibles de adaptación o mejora.

Hay también otros paquetes que permiten al usuario, a partir del lenguaje, describir todas las máscaras útiles en la aplicación.

El lenguaje de diálogo se apoya también en la definición de los parámetros «estables» de la aplicación (descripción de ficheros, secuencia de menús, descripción de datos de base de la aplicación). El lenguaje de diálogo debe ser objeto de un examen tan concienzudo como el de las funciones, puesto que revela las posibilidades de definir los parámetros de las funciones del paquete y su aptitud para hacer evolucionar la aplicación. La documentación, con frecuencia limitada al producto, no siempre describe su nivel de convivencia. La confrontación con usuarios del producto o las demostraciones se revelan casi indispensables para conocer realmente el modo de utilizar el producto y su grado de flexibilidad.

Controles y procedimientos de corrección

Cualquiera que sea la aplicación que se ha de informatizar, las fases de introducción y control de los datos la preceden.

En general, los paquetes de software poseen seguridades y sistemas de protección de los datos elaborados.

La vulnerabilidad de un equipo pequeño es considerable, de ahí la necesidad de examinar cuáles serán los cerrojos de acceso que el usuario pueda introducir. Los controles son la garantía de la calidad e integridad de los tratamientos.

Un error de entrada de datos no detectado puede costar caro en tiempo de búsqueda del error por tener que rehacer el ciclo de trabajo y por sus implicaciones en la gestión empresarial (por ejemplo: el saldo de un cliente mal actualizado).

Los procedimientos de control se sitúan, en principio, dentro de los paquetes de software en cinco niveles:

- Controles relativos a ficheros para asegurar la coherencia en el conjunto de la aplicación (parámetros utilizados, reglas de gestión, etc.),
- Controles de acceso a través de códigos o de contraseñas,
- Controles de validez de las informaciones memorizadas en el terminal,
- Controles relativos a los tratamientos,
- Controles de sistemas, gobernados por el sistema operativo y transparentes para el usuario.

Las anomalías pueden aparecer en forma de mensajes de error que se visualizan en la pantalla o por medio de una alarma sonora o incluso por medio de un cursor que emite destellos.

Es interesante conocer los tipos de mensaje, su forma y su contenido con el fin de asegurar los niveles de control establecidos por el paquete. Es preciso destacar igualmente la importancia de que sean claramente legibles (en ocasiones hay muchos mensajes que están codificados y es necesario buscarlos en la documentación).

Finalmente, el examen del paquete de software pasará por el análisis como producto en lo referente a sus posibilidades: maniobra en caso de errores diversos en su manejo:

- ¿Se puede volver atrás?
- ¿Se pueden corregir inmediatamente las anomalías?
- ¿Se puede obtener un listado de las anomalías detectadas? (¿Se describen claramente las anomalías? ¿Existen varios niveles de error?)
- ¿Son los procedimientos de diálogo relativamente rígidos o por el contrario, permiten total libertad en la elección de menús y de imágenes en pantalla en caso de errores de entrada o falsas maniobras?
- Terminado el período de aprendizaje, ¿se pueden suprimir o introducir nuevas seguridades?
- ¿Pueden memorizarse las anomalías en un fichero de espera antes de su ulterior tratamiento cuando se disponga de los elementos informáticos que permitan efectuar las correspondientes correcciones?
- La actualización de datos ¿se realiza a partir de una función explícita o según un proceso al azar?

Los controles y procedimientos de corrección aunque en la práctica suelen ser molestos internamente hacen de una Base de Datos un lugar más organizado y mejor claridad además de apoya a prevenir pequeños errores que a la larga pueden terminar por ser enormes conflictos difíciles de corregir.

Estados y visualizaciones

Un paquete de software trabaja en forma interactiva con el usuario. El órgano de este diálogo es la pantalla:

- ¿Qué informaciones permite representar?
- ¿En qué forma?

Estas dos preguntas conllevan un estudio de la información que puede aparecer en pantalla con el fin de evaluar, su número por cada imagen en pantalla, el número de imágenes en pantalla que pueden establecerse en el curso de la aplicación: «generación, actualización, consulta», la facilidad de lectura de la información (muchos paquetes de software están muy mal concebidos en este sentido); por ello es importante definir con claridad una relación hombre - máquina con el fin de lograr un mayor entendimiento entre nuestra forma de trabajar y nuestra herramienta como se muestra en la Fig. I.3.

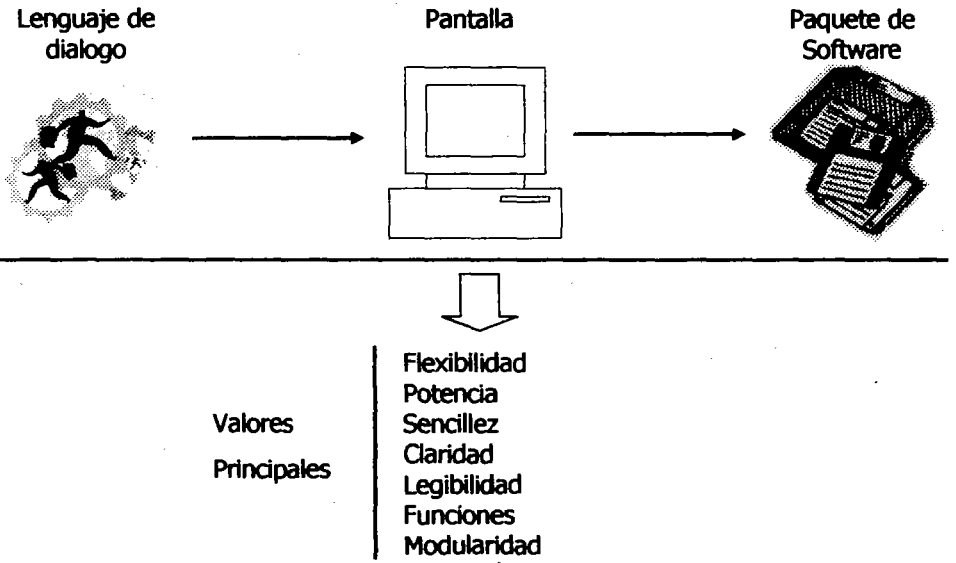

Fig. I.3. Relación Hombre/Sistema

Es preciso no olvidar en este examen el juego de parámetros que permiten «vestir» la pantalla con cuadriculados adecuados para la generación, actualización y consulta (¿estas máscaras están sujetas a ciertos parámetros?). A nivel de edición sobre papel, ¿qué posibilidades ofrece el paquete? ¿Propone un generador de estados asociado o integrado en el producto? Los mencionados estados, ¿son fijos o dependen de parámetros? Hay muchas preguntas similares que pondrán en evidencia a aquellos productos que sean los más flexibles de utilización y los mejor adaptados a las necesidades que se quieren satisfacer. Dadas las anteriores preguntas es necesario hacer una recapitulación de los criterios funcionales con que debe contar una aplicación como se muestra en la Tabla I.3.

Tipo de criterio	Enunciado del criterio	Notas
Cobertura funcional del producto	Cobertura de entre el 80 y 120% de las necesidades	Si es menor se rechazar el paquete
	Modularidad funcional, opciones, etc.	Interesa saber si el aumento de carga de la aplicación es progresivo
	Producto incluido en la gama	Muy importante sobre todo desde el punto de vista de una gestión integrada
	Juego de parámetros	Encontrar el punto medio entre un paquete «insuficiente» y otro «superabundancia de parámetros»
Lenguaje de diálogo utilizado para definición, consulta, creación, modificación, anulación y tratamiento de datos	Flexibilidad de lenguaje	Criterio fundamental para personal no informático como son los usuarios habituales del producto
	Potencia del lenguaje	Sin ser fundamental la falta de potencia puede resultar un freno para la utilización de producto
	Lenguaje de definición de la aplicación	Debe de ser sencillo y sin ambigüedades
	En castellano/español	Fundamental para que el paquete sea aceptado
Controles	Definición de parámetros de control	Interesa que el usuario pueda definirlos
	Niveles de control	Si el paquete distingue entre varios controles, será más fácil definir los parámetros y usar el paquete
	Modalidades en la aparición de anomalías	Examen del número de mensajes y de su legitimidad. Puede resultar importante para la utilización
	Reciclaje de anomalías	La respuesta a la pregunta puede condicionar la elección o rechazo del paquete

Estados y visualizaciones	Estados y visualizaciones estandarizados	Es absolutamente esencial en la elección del producto
	Estados y visualizaciones sujetos a parámetros	Muy importante si el producto sólo cubre el 80% de las necesidades
	Calidad de los estados	Aspecto agradable y comodidad visual de las informaciones ofrecidas por el paquete
	Visualización en castellano	Este criterio puede ser muy importante en la elección del paquete
	Generador de cuadrículas de pantalla	Sin duda es importante para la flexibilidad de la aplicación
	Generador de estados	Muy importante si los estados son diferentes y peculiares

Tabla I.3. Criterios funcionales

I.8.2. Criterios técnicos

Los aspectos técnicos del paquete de software no se deben despreciar. Un producto «bueno» (teniendo en cuenta la aplicación a satisfacer), puede ser rechazado por imperativos del hardware (rendimiento reducido, escasa ergonomía, etc). La evolución de una aplicación debe combinar estrechamente las posibilidades del paquete de software y del hardware. Es preciso no olvidar que la vida del paquete es frecuentemente superior a la del hardware (problema delicado de compatibilidad).

- **Duración de una aplicación:** alrededor de 7 años.
- **Duración de un equipo de cómputo:** alrededor de 4 años.

Regularmente después de 4 años, la aplicación deberá emigrar a otra plataforma.

I.8.3. Criterios comerciales

En este caso hay que permanecer atentos a las modalidades de utilización del producto.

Si el producto es modular, cuál es el precio de la versión base y de los módulos adicionales.

Algunos proveedores proponen varias versiones de un mismo producto según el hardware. Los precios pueden ser sensiblemente diferentes (esto se debe al nivel de las dificultades técnicas de adaptación del paquete). Es preciso distinguir todavía:

- Costo y modalidades de mantenimiento
- Qué mantenimiento garantiza el proveedor

En conjunto existen tres modalidades de mantenimiento:

- **Mantenimiento funcional**: El producto va siendo mejorado por el proveedor que se compromete a suministrar automáticamente la última versión aparecida.
- **Mantenimiento de modificación**: Un cambio producido en la legislación vigente puede imponer una modificación en las funciones del producto.
- **Mantenimiento técnico**: Está relacionado con la evolución técnica del hardware o del software estandarizado, que pueden ser objeto de mejoras que tiendan, por ejemplo, a aumentar su rendimiento, a soportar por etapas sucesivas un sistema de gestión de datos, etc. El mantenimiento se relaciona también con todos los errores, anomalías de funcionamiento, etc. que el usuario puede descubrir durante la utilización del producto.

Por lo general, al cliente se le propone un mantenimiento gratuito durante el primer año y después, un importe anual del orden del 12 % del precio de cesión del paquete.

Los contratos de mantenimiento son esenciales, puesto que en cierta forma, la empresa está ligada a un proveedor de paquetes.

En contra de la razón, es más fácil generalmente cambiar de hardware que de software, solamente por razón de las implicaciones mucho más profundas de este último en los fundamentos de la empresa. Se ha formado todo un mundo en torno al paquete de software, y el usuario y su organización se han preparado y concebido alrededor del mencionado paquete. Se hace más difícil, entonces, adoptar un nuevo paquete de software, que exigiría de la empresa un importante esfuerzo de cambio y de adaptación a los procedimientos de diálogo, a la consulta de los resultados emitidos por el paquete, etc. Un paquete supone todo un proceso cultural. No es este el caso del hardware, que es neutral en la formación de la empresa.

Tomar una decisión sobre el paquete a adoptar supone asegurarse de que el suministrador mantendrá el producto. En caso contrario, se corre el riesgo de utilizar un producto que se «estanque» con el tiempo, falto de actualización. Incluso es una razón por la que a veces es prudente adoptar un producto sobredimensionado, con objeto de disminuir este riesgo. Existe otra forma de actuar en la práctica, sin embargo, poco común: el proveedor suministra el lenguaje fuente de su producto y no propone el mantenimiento. La empresa lo asegura directamente a partir de los manuales técnicos de programación. Esta práctica implica conocimientos técnicos del producto y de informática dentro de la empresa. No es aconsejable salvo en raras excepciones.

En la negociación previa del contrato, el usuario debe poder conseguir documentación funcional y técnica sobre los productos seleccionados. En la práctica, no se deberá comprar basándose en la simple lectura de un folleto comercial o en la conversación con un vendedor. Los riesgos de sufrir una gran decepción son demasiado grandes.

Sin embargo, se pueden omitir estas reservas si el producto es muy conocido y ofrece, por tanto, garantía de calidad.

En el plano económico, la documentación (guía técnica, manuales de utilización, etc.), puede quedar integrada en el precio base del producto o ser objeto de factura separada.

Es aconsejable dominar plenamente este aspecto de la informatización en consideración al riesgo que supone la decisión a tomar. Dejando aparte que el precio de la documentación puede ser superior al del propio producto, ésta representa un medio de conocer dicho producto en sus múltiples facetas y posibilidades antes de firmar el contrato. La documentación constituye también un apoyo de formación indispensable después de establecido el contrato.

En informática ligera la documentación es con frecuencia pobre (demasiado anticuada). La formación viene a suponer también un criterio para la elección del paquete. Una formación mala o incompleta puede tener consecuencias desagradables para el usuario, quien no podrá evitar establecer una correlación entre tales consecuencias y el producto, incluso si es satisfactorio en principio.

Por el contrario, el suministrador del paquete podrá quizá intervenir después de la firma del contrato en la instalación del producto para explicar cómo conseguir el máximo rendimiento (si el curso de formación y la documentación no dan la respuesta adecuada). Podrá ser consultado también en un momento concreto

para colaborar en la solución de determinados problemas (determinación de los parámetros de un estado complejo o activación de una nueva función).

En el caso de los establecimientos comerciales y de ciertos proveedores, este tipo de prestación de servicio no existe.

El asesoramiento informático cuesta caro. Si se dispone de una buena documentación, de formación y de un producto que permita al usuario determinar los parámetros directamente, se pueden reducir considerablemente los posibles gastos de asesoramiento después de firmado el contrato.

Para valorar la seriedad del proveedor se pueden formular algunas recomendaciones generales; no son siempre obligatorias:

- Asistir a una demostración del producto
- Pedir una documentación más detallada que la sola documentación comercial para medir su calidad y, correlativamente, la del proveedor
- Obtener los nombres de usuarios que han adquirido el producto
- Conocer los recursos humanos, técnicos y comerciales que han creado el producto; el hecho de que un producto esté inscrito en una gama es un argumento positivo de seguridad porque, si bien es posible que el proveedor abandone un paquete concreto, resulta casi impensable que abandone toda la gama de paquetes
- Saber si el creador o creadores del producto pertenecen a la sociedad proveedora o si dicho producto ha sido concebido por otra empresa.

El conocer la respuesta a estas preguntas nos brinda una mayor claridad en cuanto al impacto que el software causa en el mercado sin dar oportunidad a falsas expectativas que pudiesen ofrecerse respecto al producto.

I.9. Las personas

En cuanto a las **personas** que participan en el manejo de los dispositivos computacionales, conjuntamente con los programas y procedimientos dentro de un centro de procesamiento de datos, se observan las labores que desempeñan los siguientes profesionales:

- **Operador de entrada de datos o "capturista":** es la persona que introduce los datos en la computadora por medio de un teclado u otro dispositivo de entrada, tales como una lectora óptica o lectora de tarjetas. En este caso resulta muy conveniente que la persona encargada de área sea la delegada de hacer estas capturas, dado que es quien conoce mejor los datos a capturar y, más aún, si la implementación del sistema podría conllevar ciertos cambios en sus políticas u operación.

- **Procesador de datos:** persona que trabaja en el procesamiento de los datos. Esta persona suele ser el encargado de convertir las tablas entregadas por el "capturista" normalmente en un formato conocido (MS Excel) a uno de mayor nivel que sería bajo en el que trabaje la base de datos (SQL).

- **Administrador de Base de Datos:** persona responsable del diseño físico, manejo de la Base de Datos, evaluación, grabar consultas, dar de alta usuarios y privilegios, proveedores, clientes y llevar a cabo los respaldos de las bases de datos que contenga el sistema. Esta persona es idealmente un trabajador de la misma empresa, con un cargo de suma responsabilidad, quien se encarga del uso técnico y conocimiento humano que la aplicación requiera, como lo es el conocer el manejo del mismo sistema hasta los niveles jerárquicos de la empresa.

- **Administrador de datos o gerente de proyecto:** persona que coordina las actividades dentro del departamento de administración de datos; también conocido como gerente de proyecto, que es el encargado de validar la información proporcionada por los usuarios durante el periodo de levantamiento de información. Esta actividad es desempeñada por un administrativo de la misma empresa.

Es importante recalcar que las personas son base importante del éxito o fracaso en la implementación de cualquier sistema ya que son éstas las que deben sentirse cómodas con la aplicación ya que suele ser esta su herramienta de trabajo cotidiano.

I.10. Panorama del mercado actual de software

Hace años el software era limitado en su desarrollo debido a los equipos con que se contaba; sin embargo, desde hace no muchos años, con la introducción de programas cada vez más gráficos y didácticos es que el software tomo la delantera al acortar la vida útil de un equipo de computo, al exigir para su óptima operación mayores niveles de procesamiento, un mejor uso de la memoria y ocasionalmente falta de compatibilidad entre versiones, lo que va rezagando muy rápidamente a usuarios de equipos antiguos hasta dejarlos fuera de competencia respecto a las nuevas formas de trabajo.

Y por otra parte está la nueva división de opciones que nos trae el uso de software libre y hecho a la medida, que resulta ser de mejores resultados pero carece de una sólida administración que dé soporte a estas nuevas tecnologías aún en expansión contra la ya conocida y experimentada gran empresa del software de soluciones menos específicas pero con un poder económico mayor que les respalda. De tal forma que podemos llegar al desarrollo de software particular o

hecho a la medida mediante dos caminos distintos que son el uso de herramientas de software comercial así como libre, como se muestra en la Fig. I.4.

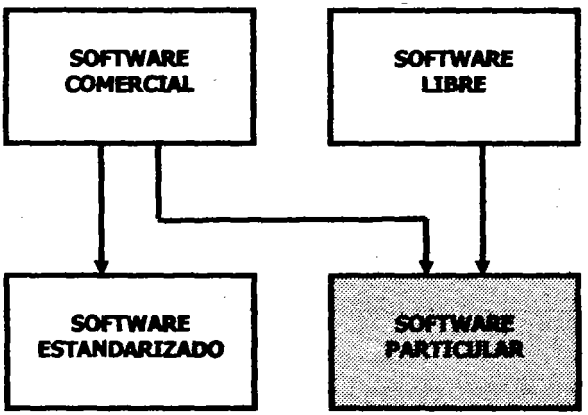

Fig. I.4. Actuales rutas hacia el desarrollo de software particular.

Los caminos hacia un desarrollo pueden ser muchos, sin embargo, muchas de estas decisiones son resueltas por muchas personas de distintas áreas de la empresa las cuales analizan factores técnicos, económicos y de recursos humanos entre otros.

I.10.1. Presentación del mercado

El mercado de paquetes de software está en constante evolución. Su tasa de crecimiento en número de productos por año es elevada. Los especialistas prevén que esta expansión continuará aún durante varios años.

El paquete lo concibe una sociedad que busca obtener un beneficio con su comercialización. El estudio previo se apoya en el marketing del segmento de mercado explorado. Si el mercado es estrecho (insuficiencia prevista de ventas) o está ya ocupado por otros proveedores, es posible que el promotor del paquete

renuncie a su idea al no darse una rentabilidad interesante para su proyecto. También lo abandonará si el segmento de mercado estudiado inicialmente no es solvente (sector de actividad en dificultades).

Se podría dibujar así un mapa de la economía nacional tomando como base los segmentos de actividad cubiertos por los paquetes de software.

Con el tiempo, se irán cubriendo más y más sectores, y únicamente algunas actividades muy específicas o marginales quedarán por cubrir. Al mismo tiempo se producirá la desaparición de sociedades y productos que no han sabido posicionarse en el mercado.

Lamentablemente, los grandes corporativos del software capaces de agrupar a grandes expertos están inmersos en una correlación clásica: cuanto más bajo es el precio unitario y se logra satisfacer a más persona dejando a un lado las soluciones específicas de la empresa el producto más se vende. Hasta hace unos años, la gran informática no se ajustaba a esta economía, lo que hacia que los productos fuesen más caros, sin embargo la nueva tendencia del cooperar del software libre y los altos costos de las grandes empresas de software han dado nuevo rumbo y posibilidades de expansión al software hecho a la medida de tal forma que este tipo de software va ganando camino poco a poco en el mercado mundial como se muestra en la Tabla I.4.

Industria			
Micro	**Pequeña**	**Mediana**	**Corporativo**
Software Particular*	**Software Particular***	**Software Particular*/Comercial**	**Software Comercial**
* Software que es desarrollado a necesidades específicas de la empresa			

Tabla I.4. Elección del software en la industria

A nivel de aplicación, los paquetes de software se dividen en dos grandes grupos:

- Horizontales
- Verticales

Paquetes horizontales

Todas las empresas tienen en común una serie de funciones tradicionales, como contabilidad y pagos, reguladas por las leyes o por normas de gestión estables.

Este tipo de aplicaciones se denominan horizontales en razón de su «universalidad nacional».

A estas aplicaciones corresponden paquetes multisectoriales que siguen la horizontalidad de las aplicaciones.

Es el caso, por ejemplo, de paquetes para tratamiento de textos, productos de simulación financiera, etc. no se dedican a una actividad, sino que operan en mercados generales sin un objetivo particular (en efecto, los proveedores son obligados a delimitar su mercado en función de su potencial de ventas e infraestructura).

Paquetes verticales

En oposición a los horizontales los verticales se dedican a ciertos sectores de actividad: expertos contables, comercio, agricultura, banca, etc.

De este modo ciertos sectores (por poco que actúen financieramente como «portadores» para los proveedores) están cubiertos por paquetes que se proyectan hacia sus necesidades específicas y sus particularidades. Estos productos han sido concebidos por especialistas del sector y no tienen sentido fuera de él.

Una empresa puede muy bien utilizar paquetes de software horizontales y verticales. Técnicamente nada los distingue y es solamente su campo funcional lo que les sitúa en una o en otra categoría.

En ciertos sectores todos los productos son casi verticales: gestión de producción en las compañías de seguros, gestión financiera en bancos, etc.

Por regla general, los paquetes verticales son más caros que los horizontales; la explicación estriba en la economía de escala y en los gastos de desarrollo, con frecuencia más importantes.

Lo anteriormente mostrado resulta de suma importancia como base para aquellos que inician el estudio de las Bases de Datos ya que sirve de apoyo y contexto de estudio, desde sus orígenes, constitución básica, organización física y lógica hasta de algunos criterios de selección importantes, así como un perfil general de las personas que laboran en torno a ella y un panorama de mercado actual del software.

Con ellos es que estamos listos para comprender con mayor claridad temas de mayor profundidad que se estudiarán en los siguientes capítulos.

Capítulo II

Análisis Técnico de la Base de Datos

Este capítulo nos pone en marcha en el desarrollo de sistemas de Base de Datos.

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

El diseño conceptual parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la base de datos. Un esquema conceptual es una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD que se vaya a utilizar para manipularla. Un modelo conceptual es un lenguaje que se utiliza para describir esquemas conceptuales. El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un esquema lógico es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. Un modelo lógico es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto.

El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un esquema físico es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos. En la fig. II.1 se muestran los grupos de fases del diseño de una base de datos que son necesarias desglosar para un buen diseño.

Fig. II.1. Componentes de una base de datos.

El diseño conceptual es el punto de partida para el estudio de los componentes de una Base de Datos y los diseños lógico y físico pueden comenzarse indistintamente o paralelamente.

II.1. Una metodología para el desarrollo de Bases de Datos relacionales

Analizadas en líneas generales las distintas fases que lleva consigo la creación de una base de datos, se presenta una metodología para el desarrollo de BD, estudiando en primer lugar el concepto y sus componentes básicos, para pasar después a considerar la metodología propuesta con sus diferentes fases, y terminar analizando las características que ha de poseer una metodología de desarrollo. Aunque la metodología propuesta podría ser aplicada al desarrollo de una base de datos, sea cual sea el modelo que la soporte, está especialmente orientada al desarrollo en el modelo relacional.

Concepto de metodología

Gracias al avance tecnológico, actualmente hay una gran difusión de los Sistemas gestionadotes de Base de Datos (SGBD), que ya existen para todo tipo de plataformas, desde los grandes computadores a los computadores personales. Sin embargo, y a pesar del esfuerzo realizado por numerosos investigadores y estudiosos del tema, el desarrollo de una BD sigue siendo una tarea larga y costosa.

Vencer estas dificultades inherentes al desarrollo de una base de datos obliga a abordarlo con procedimientos y metódicos, tal como reconocen numerosos autores. Así, "Las dificultades de la concepción de una base de datos, claman por una respuesta metodológica".[12]

En la ingeniería del software se han realizado importantes esfuerzos para encontrar las metodologías más adecuadas; esto se debe al gran impacto que una metodología tiene en el desarrollo de un producto de software, ya sea en lo que se refiere en los costos y plazos de entrega del mismo, como a la calidad y mantenimiento del producto.

Como se señala: "Un buen diseño es la clave de una eficiente ingeniería de software. Un sistema de software bien diseñado es fácil de aplicar y mantener, además de ser comprensible y fiable. Los sistemas mal diseñados, aunque puedan funcionar, serán costosos de mantener, difíciles de probar y poco fiables".[13]

A veces el diseño de una BD se ha limitado simple y llanamente a la teoría de la normalización, cuando en realidad debe abarcar muchas otras etapas que van desde la concepción hasta la implementación.

Por tanto, tenemos que definir en primer lugar qué puede entenderse *por una metodología para el desarrollo de BD*, cuestión que ha sido abordada por numerosos autores a lo largo de los últimos años. También se han propuesto varias metodologías como MERISE, SSADM, MÉTRICA y DATAID-1 entre otras integrando. En la Tabla II.1 se muestran algunas de las metodologías más usadas a través de los años para el desarrollo de software de Base de Datos.

[12] Rolland, C., Foucaut, O y Benci, G. (1988). *Conception des systemes d'information. La méthole REMORA.* París, Francia. Eyrolles.
[13] Sommerville, I. (1988). Ingeniería del Software. Addison-Wesley Iberoamericana. México.

Año	Metodología	Fases para su Desarrollo
1968	Conceptos sobre la programación estructurada de DIJKSTRA	
1974	Técnicas de programación estructurada de WARNIER y JACKSON	
1975	Primeros conceptos sobre diseño estructurado de MYERS y YOURDON	1. Realizar los DFD del sistema 2. Realizar el diagrama de estructuras 3. Evaluar el diseño 4. Preparar el diseño para la implantación
1977	Primeros conceptos sobre análisis estructurado GANE y SARSON	1. Construir el modelo lógico actual (DFD lógico actual) 2. Construir el modelo del nuevo sistema: elaborar una especificación estructurada y construir un modelo lógico de datos en tercera forma normal que exprese el contenido los almacenes de datos. 3. Seleccionar un modelo lógico 4. Crear el nuevo modelo físico del sistema 5. Empaquetar la especificación
1978	Análisis estructurado: DEMARCO y WEINBERG	1. Construir el modelo físico actual (DFD físico actual) 2. Construir el modelo lógico actual (DFD lógico actual) 3. Crear un conjunto de modelos físicos alternativos 4. Estimar los costes y tiempos de cada opción 5. Seleccionar un modelo 6. Empaquetar la especificación
1978	Nace MERISE	1. Estudio Preliminar 2. Estudio Detallado 3. Implementación 4. Realización y puesta en marcha
1981	SSADM "structured Sydtem Analysis Design Method" (versión inicial) Information Engineering (versión inicial)	Planificación Estratégica 1. Estudio de Viabilidad 2. Estudio completo Análisis de Requisitos Especificación de Requisitos Especificación Lógica del Sistema 3. Desarrollo Diseño Físico Construcción y Pruebas Puesta en Marcha
1985	Análisis y Diseño estructurado para sistemas de tiempo real de WARD y MELLOR	
1986	SSADM Versión 3	

1987	Análisis y Diseño estructurado para sistemas de tiempo real de HATLEY y PIRHBAY	
1989	METRICA (versión inicial)	1. Plan de Sistemas de Información 2. Análisis de Sistemas 3. Diseño de Sistemas 4. Construcción de Sistemas 5. Implantación de Sistemas
1990	SSADM Versión 4	
1993	METRICA Versión 2	
1995	METRICA Versión 2.1	
	DATAID-1 (versión inicial)	1. Análisis de los requerimientos 2. Diseño conceptual 3. Diseño lógico 4. Diseño físico
	DATAID-D	1. El diseño de fragmentación 2. La colocación no redundante 3. Colocación redundante 4. La reconstrucción del esquema local

Tabla II.1. Relación histórica de las principales metodologías y sus fases de desarrollo

Sin embargo y a pesar de las muchas técnicas realizadas cualquier centro de desarrollo puede montar su propia metodología, esta alternativa implica disponer del tiempo necesario para el su desarrollo; por tanto, lo más práctico es seguir los métodos que ya han demostrado su validez y son de aplicación universal.

Se afirma que "una metodología de diseño puede concebirse como un conjunto de herramientas y técnicas empleadas dentro de un marco organizacional, que puede ser aplicado consistentemente a proyectos sucesivos de desarrollo de la estructura de una base de datos".[14] Y el diseño de una base de datos se define como "el proceso de desarrollo de una estructura de base de datos a partir de los requisitos de los usuarios".[15]

Otra definición digna de ser destacada es la que señala que "una metodología es una colección de medios propuestos para controlar el proceso de

[14] Wasserman, A. I. (1979). *The Data Management Facilities of PLAIN.* Proc. 1979 ACM SIGMOD International Conference on the Management of Data. Boston, Mass., mayo.
[15] Teorey, T. J. y Fry, J. P. (1982). Design of Database Structures. Englewood Cliffs, N. J.: Prentice-Hall.

desarrollo".[16] De forma parecida se afirma que "una metodología es una serie de métodos que pueden ser aceptados ampliamente y utilizados en el ciclo de la vida completo del diseño de la base de datos. Estos métodos cumplen distintas tareas en distintos pasos".[17]

La metodología propuesta en éste trabajo sigue el espíritu de las definiciones anteriores en el sentido de considerar el proceso de desarrollo como un conjunto de medios a aplicar en las distintas etapas del ciclo de la vida de una base de datos. Más precisamente, se ajusta a la definición de Inforsid, posteriormente ampliada al considerar que: "Una metodología es un conjunto de modelos y herramientas que nos permiten pasar de una etapa a la siguiente en el proceso de diseño de la base de datos".[18]

Teniendo en cuenta que una metodología es "un conjunto de modelos, lenguajes y herramientas que nos facilitan la representación de los datos en cada fase del proceso de diseño de una base de datos, junto con las reglas que permiten el paso de fase a la siguiente", el análisis de todos estos elementos es fundamental para poder comprender y aplicar correctamente una metodología de diseño.

Entendemos por herramienta "cualquier recurso particular a disposición de la metodología para realizar las operaciones que en ella se prevén"[19]; herramientas serán los diagramas, grafos, teorías, etc. que se han de aplicar a las distintas fases del desarrollo. Los modelos, los lenguajes y la documentación son también herramientas, pero dado su especial interés se consideran de forma individual.

---

[16] Rochefeld, A. (1986). MERISE, an Information System Design and Development Methodology. En 5th International Conference on Entity-Relationship Approach, Dijon, Francia, 17-19 de noviembre.
[17] Shan y Shixuan (1984). "Normal Entity-Relationship Model, A New Method for Enterprice Schema Design". En: *IEEE*.
[18] Rolland, C., Foucaut, O y Benci G. (1988). *Conception des systemes d' information. La méthode REMORA*. París, Francia. Eyrolles.
[19] Batini el al. (1981)

Ya hemos definido un modelo de datos como un conjunto de conceptos, reglas y convenciones que permiten describir y manipular los datos de la parcela del mundo real que constituye nuestro universo del discurso; el esquema obtenido al aplicar un modelo de datos a un cierto universo del discurso constituirá la visión que del mundo real tiene el diseñador, el cual lo contempla bajo los objetivos impuestos por el sistema de información que está creando.

Un lenguaje de datos está siempre basado en un determinado modelo de datos y es el resultado de definir una sintáxis para el mismo, lo que va a permitir expresar un esquema (basado, por ejemplo, en el modelo relacional) en una sintáxis concreta (por ejemplo, la del SQL).

La documentación nos permitirá describir de forma normalizada los resultados de cada etapa, facilitando así la labor del diseñador y ayudando al mantenimiento de la base.

Las reglas actuarán sobre los elementos de entrada en cada fase para conseguir de una manera semiprogramable las salidas de cada una de ellas, permitiendo en algunos casos elaborar distintas alternativas de diseño.

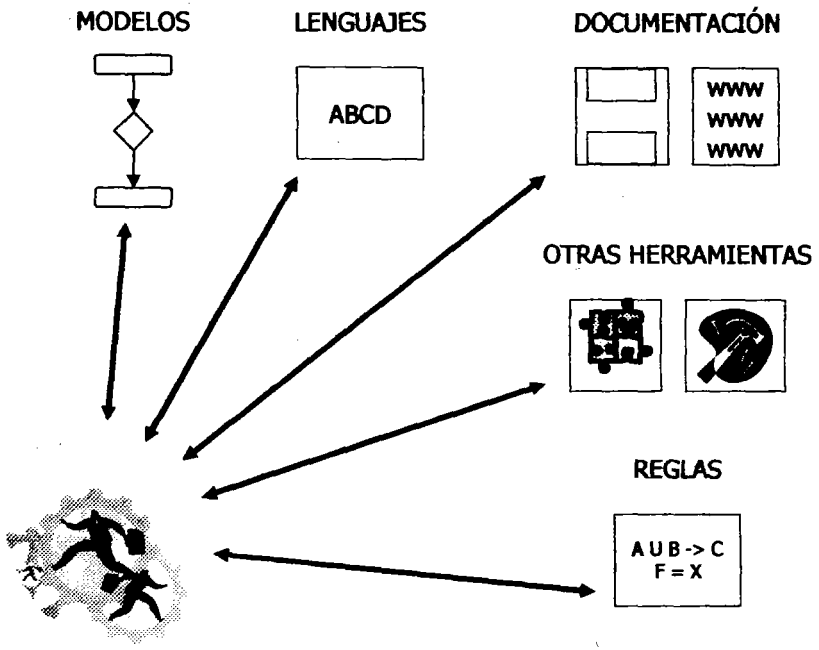

Fig. II.2. Herramientas para el diseño de una Base de Datos

Estos cinco conceptos (modelos, lenguajes, documentación, otras herramientas y reglas), que se presentan en la Fig. II.2, están estrechamente ligados: un lenguaje permite la expresión organizada de los conceptos del modelo, los modelos no pueden aplicarse de forma satisfactoria sin una metodología, y una metodología será más eficaz con el apoyo de herramientas que faciliten su aplicación y con reglas que permitan pasar de una etapa a otra, ayudando a resolver los problemas que van apareciendo en el proceso de diseño, el cual debe estar perfectamente documentado para que puedan llevarse a cabo las revisiones y el mantenimiento. Los participantes (directivos, usuarios e informáticos) constituyen un elemento esencial del desarrollo.

II.2. Proceso de creación y metodología de desarrollo de bases de datos

El proceso de creación de una base de datos se integra dentro de un proceso sumamente amplio, que incluye desde que se toma la iniciativa de crear la base de datos hasta que se encuentra totalmente operativa.

Una buena metodología arma a sus desarrolladores con un juego de herramientas, técnicas confiables y repetibles que se adecuan particularmente bien a los problemas que están tratando de resolver. No todos los problemas se generan en un mismo sitio. Algunos son ricos en datos, pero tienen muy pocos requerimientos de procesamiento y viceversa. Una metodología balanceada incluye técnicas que le dan a los analistas y diseñadores una amplia cobertura de todos los aspectos que deban modelar, pero les permite desviar su énfasis de modelado para adaptarse a la tendencia del problema.

Una buena metodología de análisis o diseño debe tener cinco características principales:[20]

1. **Motivar la actividad pretendida.** El buen diseño debe motivar la toma de decisiones ayudando a evaluar alternativas.

2. **Ser completa.** El diseño necesita ser completo, de tal forma que cubra cada uno de los aspectos principales del software que necesita construirse.

3. **Ser modificable en su corrección.** El diseño debe ser verificable antes de su construcción.

[20] Ruble, David A.. Análisis y diseño practico de sistemas cliente/servidor con GUI. Pagina 15. Ed. Prentice Hall. México. 1997.

4. **Producir productos contra los que se pueda medir el avance.** Una buena metodología de diseño crea productos diferenciados que son cuantificables.

5. **Ser fácilmente aprovechable en la fase subsecuente.** El diseño debe ser fácilmente aprovechado en el producto final. Debe expresar el uso y la estructura del sistema en una forma muy cercana al resultado pretendido.

Este capítulo resume el ciclo de vida de una base de datos, abordando desde el aspecto técnico de su creación y organización llevada de la mano de una metodología bien ordenada.

II.3. Introducción al ciclo de vida de una base de datos

La creación de una base de datos es generalmente, una operación difícil, larga y costosa, que no puede improvisarse. No se trata solamente de un problema técnico, ya que esta decisión puede tener repercusiones a todos los niveles de la empresa (transferencia de responsabilidades entre personas y servicios, reorganización del departamento de informática, formación de los usuarios, cambio de determinados métodos de trabajo, etc.) esto hace de ella una decisión que atañe a la política empresarial, por lo que no debe ser abordada en exclusiva por los técnicos.

Así pues, la responsabilidad de las decisiones relativas a todo el proceso de creación de una base de datos no corresponde únicamente a los informáticos, sino que, en ciertas fases, son los directivos y los usuarios quienes poseen el protagonismo.

De forma sistemática las distintas fases que comprende la puesta en marcha de un sistema de información orientado hacia las bases de datos se encuentra resumida en la siguiente Fig. II.3.

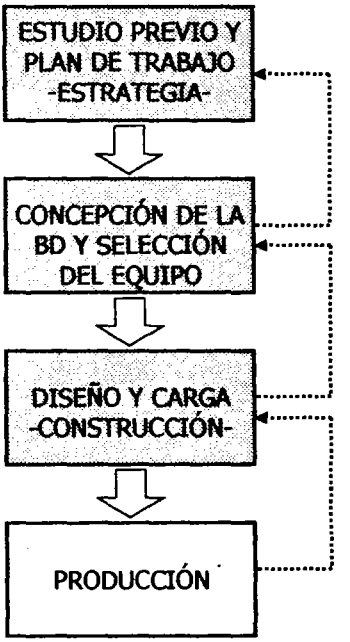

Fig. II.3. Resumen de las fases para la puesta en marcha de una BD

A grandes rasgos esta es la estrategia que se sigue para la puesta en marcha de un sistema.

II.4. Concepción de la base de datos y selección del equipo

Una fase y quizá una de las más importantes es la del análisis o levantamiento de información que se ha de integrar en la base de datos a fin de

alcanzar los objetivos propuestos y se representa esta información en un modelo conceptual de datos independiente del SGBD que se vaya a utilizar. Además, si no se dispusiese del equipo físico y/o lógico, se ha de llevar a cabo la evaluación y selección del mismo. En la Fig. II.4 se representan las actividades que integran esta fase.

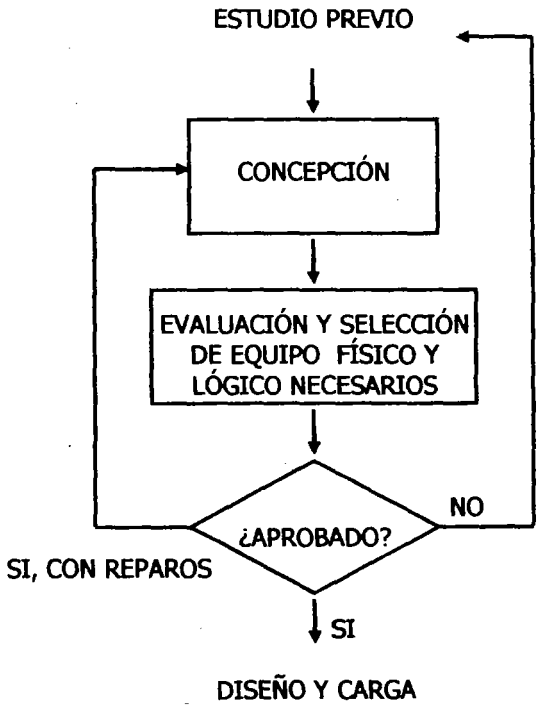

Fig. II.4. Actividades de la fase de concepción de la base de datos y de selección del equipo

En la mayoría de las ocasiones, la empresa que solicita la implementación de una herramienta de software ya trabaja con cierto equipo físico al cual hay que adaptarse; por ello es que es uno de los puntos fundamentales para la toma de decisiones ya que esto conlleva siempre un gasto extra, al cual la empresa no

siempre contempla, ya que generalmente conduce a fricciones laborales en términos de contratos.

Concepción de la base de datos

En la fase de concepción se analizarán los modelos de trabajo mediante los cuales funciona la empresa y así mismo como son los sistemas de información anteriores (Bases de Datos anteriores) lo que dará una primera imagen, probablemente deformada, del mundo real (empresa u organismo), y a continuación se determinarán las necesidades de los usuarios, concretándose las funciones que hay que integrar en la base de datos y las modificaciones que habrá que introducir en las aplicaciones existentes para que se adapten mejor a los fines de la organización y al nuevo enfoque que supone la puesta en marcha de dicha base.

Mediante estos dos pasos llegaremos a una lista de las informaciones que la organización necesita, así como de los requisitos del sistema, a partir de los cuales se podrán concretar qué datos de entrada, procedimientos y medios se precisarán para obtener dichas informaciones.

También habrá que describir las actividades de la organización, analizándolas en términos de sistema, de subsistemas y de entorno.

Todo ello nos permitirá determinar, por un lado, las características del sistema (requisitos en cuanto a protección de los datos, flexibilidad, etc.) y de su arquitectura (exigencias de acceso, lenguajes, etc.) y, por otro lado, el contenido de la base –datos–, con especificación de su volumen, volatilidad, normas de validación, y una lista de reglas de gestión.

En la construcción del esquema conceptual se puede contar con la ayuda de la computadora que, en un proceso interactivo, irá poniendo de manifiesto las consistencias del esquema propuesto por el diseñador, el cual lo podrá ir depurando paso a paso con ayuda de las indicaciones suministradas por la máquina.

La fase de concepción termina contrastando el esquema conceptual que podríamos llamar *bruto*, con la realidad y sometiéndolo a sucesivas adaptaciones, hasta conseguir, en un proceso interactivo, una representación que sea una síntesis de los esquemas externos de los distintos usuarios, a partir del cual se debe poder obtener de nuevo los esquemas externos.

Como se puede deducir, esta fase de concepción es totalmente independiente de la máquina donde el sistema será implementado, y su enfoque está dirigido a obtener un análisis de la información ajeno a cualquier consideración que se relacione con las características de la computadora o del SGBD que se utilizará, posteriormente, para su puesta en marcha.

II.5. Diseño y Carga

Esta fase comprende tanto el diseño lógico de la base de datos y su codificación, como la carga de los datos y la prueba de los programas. Al igual que las fases anteriores se trata de un proceso iterativo, al final del cual la base de datos puede entrar en explotación. En la Fig. II.5 se representan las distintas actividades de esta fase.

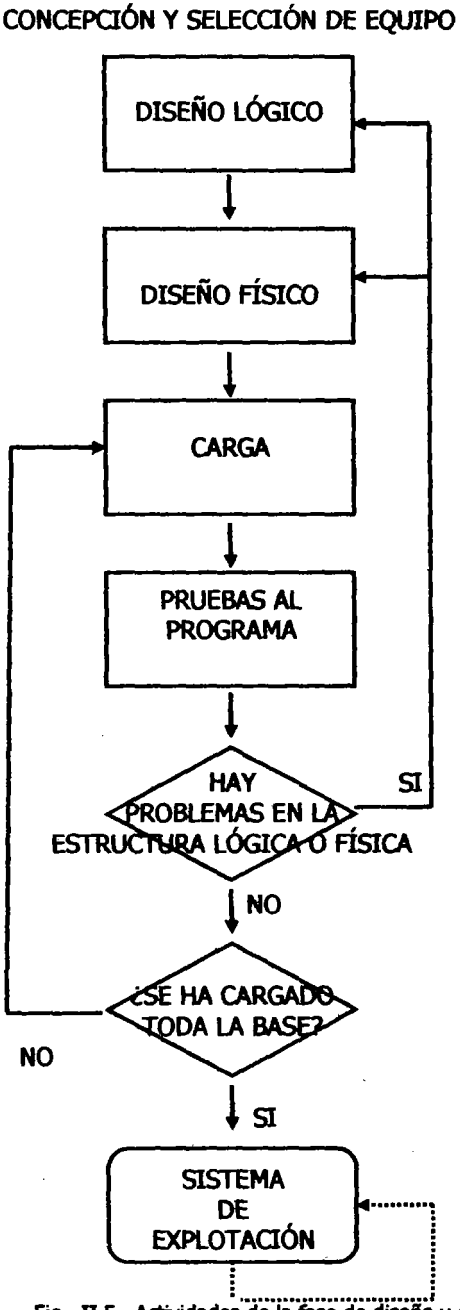

Fig. II.5. Actividades de la fase de diseño y carga

Todo sistema ha tener una fase de pruebas a pesar de ser un procedimiento probado este no siempre responde de igual manera así como los usuarios para quienes ha sido implementada no son los mismo y por tanto el sistema siempre esta expuesto a ser forzado de distintos "maltratos".

II.5.1. Diseño lógico y físico

El esquema conceptual, obtenido en la fase anterior, ha de ser estructurado teniendo en cuenta las características del SGBD elegido y de acuerdo con el modelo implementado en el mismo. Definida la estructura lógica de la base de datos se pasará a obtener la estructura física (esquema de almacenamiento interno).

IV.5.2. Carga y optimización de la base

Definida la estructura física de la base de datos, es preciso cargar los datos en la misma. En general, muchos de estos datos proceden de aplicaciones anteriores, en cuyo caso, lo único que habrá que hacer es proceder a la carga de los archivos; muchos SGBD dan facilidades para la migración, evitando escribir los correspondientes programas. Para los datos que no se encuentran capturados, habrá que recogerlos mediante los adecuados impresos e introducirlos en base de datos.

Al realizar el plan de trabajo hay que contar con esta fase, que puede resultar frustrante, tanto en plazos como en costos, especialmente si los datos no se encuentran capturados o debidamente codificados.

Paralelamente a la fase de diseño se deberán ir desarrollando los programas y procedimientos necesarios para implementar las reglas de gestión que se

definieron en la fase de concepción, de forma que, cuando se vayan cargando en la base de datos los distintos conjuntos de información, se puedan ir probando los programas que manejan esos datos.

Cargados en la base algunos archivos, se debe comenzar inmediatamente las pruebas de la base de datos y medir sus rendimientos, con objeto de poder ir ajustando la estructura física e incluso, a veces, la estructura lógica con fines de optimización.

Existen herramientas capaces de desarrollar prototipos a partir de unas especificaciones iniciales que, si bien no constituyen el sistema de información real, sí pueden ayudar mucho a comprobar si las especificaciones son correctas y mostrar a los usuarios cual va a ser la interfaz y el comportamiento del sistema.

Si la base de datos ha sido bien diseñada, la independencia entre las distintas estructuras permitirá la optimización de la base por sucesivos retoques sin que ello afecte a los programas de aplicación que acceden a la misma. No suele ser conveniente cargar simultáneamente todos los conjuntos de información que constituirán la base de datos completa; por el contrario, un desarrollo gradual dará, probablemente, resultados más satisfactorios.

II.6. Herramientas de diseño de Bases de Datos

A lo largo de los años, el desarrollo de los proyectos de software causan bastantes confusiones y malas interpretaciones en los requerimientos de los clientes y usuarios, en parte debido a la abundancia de notaciones, metodologías y conceptos que hace que los desarrolladores de sistemas no se pongan de acuerdo en que es lo que realmente están elaborando.

Las herramientas de diseño nacen como una disciplina para aplicar los principios, técnicas y herramientas de desarrollo de software; surgió porque todos los desarrolladores en la década de los 80's realizaban el software de forma artística, es decir, utilizando métodos y técnicas donde la experiencia (el ensayo-error) era el camino a seguir. Este enfoque produjo grandes y exitosos productos de programación pero conforme los proyectos se volvieron más complejos fue necesario el apoyo con herramientas que acortaran los tiempos de plantación y desarrollo además de ser de un uso sencillo.

Clases de herramientas

Se pueden utilizar tres clases de herramientas para el diseño de bases de datos:

- **Sistemas o prototipos de investigación:** Resultan, sin duda, los más completos a la hora de soportar las metodologías de diseño y de ofrecer ayudas al diseñador; pero que, como contrapartida, suelen ser difíciles de encontrar y no cuentan con soporte, documentación y evolución adecuados.
- **Herramientas comerciales específicas:** Para el diseño de bases de datos, dentro de las que destacan muchas desarrolladas por los propios fabricantes de SGBD. Este tipo de herramienta es muy adecuado para aquellos que desarrollen hacia entornos en los que el componente de datos tenga un peso fundamental, como sucede en muchas aplicaciones de gestión. Un ejemplo de éstas son las herramientas de diseño de tablas mediante gráficas de E/R con las que cuentan muchas aplicaciones como SQL o MS Access.
- **Herramientas CASE generales:** Incluyen junto con técnicas para el desarrollo de procesos (diagramas de flujo de datos, diagramas de estructuras de cuadros, etc.) otras para el diseño de bases de datos (como el modelo E/R). Constituyen entornos muy completos, integrándose con

numerosas herramientas, lenguajes de cuarta generación, generadores de código, etc. Por ejemplo el "Excelerator" (1984) que fue la primera herramienta CASE como hoy la conocemos y actualmente el EASYCASE o WINPROJECT.

El primer tipo de herramientas suele soportar de forma exhaustiva la fase de modelado conceptual, ofreciendo la posibilidad de crear esquemas a partir del lenguaje natural; así como de integrar vistas (técnica que consiste en elaborar esquemas conceptuales parciales y a partir de éstos ir obteniendo esquemas conceptuales mayores que los engloben). Desafortunadamente, existen muy pocas herramientas comerciales que soporten de manera satisfactoria esta técnica.

Las herramientas comerciales específicas para diseño de bases de datos, si bien permiten construir un esquema conceptual bastante completo, se suelen limitan a realizar la detección de sinónimos y homónimos entre las diferentes vistas, pero no asisten al diseñador durante el proceso de integración. Donde destacan las herramientas construidas por los fabricantes de SGBD es en los aspectos de diseño lógico específico y de diseño físico de la base de datos que soportan.

Por último, las herramientas CASE generales normalmente ponen más énfasis en el soporte del análisis y construcción de procesos y programas que en la base de datos; a pesar de esto, suelen soportar el diagrama E/R y la producción de esquemas SQL.

II.7. Marco para la evaluación de herramientas de diseño de Bases de Datos

Dada la gran heterogeneidad de herramientas existentes para el diseño de bases de datos, han aparecido varias propuestas para su clasificación (por ejemplo, CHEN et al. (1982), NAVATHE (1985) o STOREY y GOLDSTEIN (1993)). A continuación se presentan el marco propuesto en RAM (1994) por considerar que es uno de los más completos.

Ram propone evaluar las herramientas de diseño de bases de datos según 18 parámetros:

- **Origen:** Que permite distinguir entre herramientas desarrolladas en entornos académicos de las creadas en entornos comerciales.
- **Fases de desarrollo soportadas.**
- **Modelo/conceptos subyacentes:** Que pueden ser semánticos, orientados a objetos, basados en formas, lenguaje natural, relacional, etc.
- **Metodología/algoritmos utilizados:** Que examina las reglas o los heurísticos empleados para producir un diseño.
- **Entradas:** Describe el tipo de entradas que requiere la herramienta.
- **Salidas:** Describe las salidas producidas por la herramienta.
- **Medios de representación/Interfaces:** Este parámetro identifica interfaces gráficas y textuales de la herramienta.
- **Repositorio de información:** En el sentido de conocer qué diseño y estructura utiliza (archivos, SGBD relacional, SGBO, etc.).
- **Documentación.**
- **Análisis de alternativas:** La herramienta presenta alternativas y ayuda al diseñador a tomar decisiones.
- **Verificación y validación del diseño.**

- **Público:** Por ejemplo, usuarios finales, diseñadores, etc.
- **Validación/Difusión de uso:** Para conocer en qué casos se ha utilizado.
- **Características operativas:** Identifica el entorno hardware en el que opera, el lenguaje en que ha sido desarrollada, etc.
- **Facilidad de modificación:** Examina la capacidad de la herramienta para llevar a cabo cambios en la herramienta.
- **Facilidad de reutilización.**
- **Extensiones futuras.**
- **Comentarios generales.**

II.8. Herramientas CASE y diseño de Bases de Datos

El desarrollo del software ha sido abordado durante años sin el apoyo de unos principios de ingeniería y de unas metodologías, mediante los cuales se consiguiera una mayor productividad y calidad de los productos desarrollados. En los años setenta, se advierten señales de preocupación por estos problemas y aparecen las primeras metodologías y técnicas destinadas a ofrecer soporte al desarrollo de sistemas de información, las cuales empiezan unos años más tarde a ser integradas en herramientas destinadas a automatizar algunas fases del ciclo de vida.

Introducción a las herramientas CASE para el diseño de software

La tecnología conocida con el nombre de CASE (*Computer Aided/Assisted/Software/System Engineering*) "Sistema de Ingeniería de/Software/Asistido por Computadora" se puede definir como el conjunto de herramientas y metodologías que soportan un enfoque de ingeniería para las distintas fases del desarrollo de software.

Esta tecnología surge a mediados de los años setenta, cuando empiezan a aparecer las primeras metodologías estructuradas y se inician las investigaciones sobre entornos de desarrollo. A mediados de los años ochenta, se populariza y surgen las primeras herramientas de documentación y diagramación automática. También ha desempeñado un papel importante en este desarrollo la aparición de las estaciones de trabajo, que aportan una buena interfaz gráfica asociada a una gran capacidad de proceso, dos de los requisitos básicos para el CASE.

Además en la década pasada surge el concepto de repositorio como núcleo de un entorno CASE, así como generadores de programas y aplicaciones que automatizan gran parte de las últimas fases del ciclo de vida. En paralelo también aparecen los gestores de proyectos, algunos de los cuales se integran con herramientas de desarrollo.

Sin embargo, esta primera generación de herramientas fracasa, debido principalmente a tres factores:

- Limitaciones de los productos.
- Falsas expectativas sobre sus posibilidades.
- Incorrecta implantación.

Actualmente, se ha llegado a una fase de madurez en la que se empieza a implantar una *segunda generación* de herramientas (algunas de las cuales ya no aparecen bajo el término CASE, para no rememorar el fracaso anterior), que superan gran parte de las limitaciones anteriores. Además, los usuarios conocen mejor sus posibilidades y han aprendido a poner unas expectativas más justas sobre éstas, mejorando también los procesos de adopción de metodologías y herramientas.

La tecnología CASE supone la "*informatización de la informática*", es decir "*la automatización del desarrollo del software*", contribuyendo así a elevar la productividad y la calidad en el desarrollo de sistemas de información de forma análoga a lo que suponen las técnicas CAD/CAM[21] en el área de fabricación.

Este nuevo enfoque a la hora de construir software, persigue los siguientes objetivos:

- Permitir la aplicación práctica de metodologías estructuradas en un primer momento y, en la actualidad, también orientadas a objetos, lo que resulta muy difícil sin emplear herramientas.
- Mejorar la calidad del software.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento de los programas.
- Estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes software.
- Permitir un desarrollo visual de las aplicaciones, mediante la utilización de gráficos.

Estos nuevos enfoques dan orden a nuestras aplicaciones al estandarizar la documentación y presentar de manera cuantificable los avances mediante gráficas.

De una manera muy esquemática, la Fig. II.6, afirma que una herramienta CASE se compone de los siguientes elementos:

[21] CAD (*Computer Aided Design*), diseño asistido por computador. CAM (*Computer Aided Manufacturing*), fabricación asistida por computador.

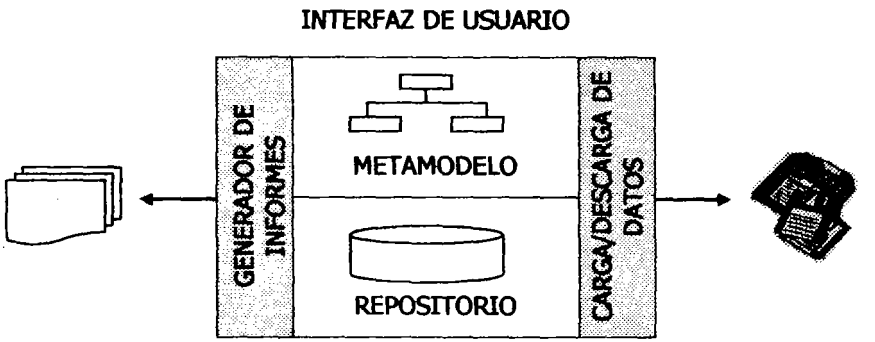

Fig. II.6. Elementos de una herramienta CASE

- **Interfaz de usuario:** Que constará de editores de texto y herramientas de diseño gráfico, que permitan mediante la utilización de un sistema de ventanas, iconos y menús, con la ayuda del ratón, definir los diagramas, matrices, etc. que incluyen las distintas metodologías.
- **Sistema de gestión del repositorio (diccionario):** Donde se almacenan los elementos definidos o creados por la herramienta, y que se basa en un SGBD o en un sistema de gestión de archivos.
- ***Metamodelo* (no siempre visible):** Que define las técnicas y metodologías soportadas por la herramienta, y que es conveniente que pueda ser extensible por parte del usuario.
- **Generador de informes:** Que permite obtener toda la documentación asociada a las técnicas y metodologías.
- **Herramienta de carga/descarga de datos:** Que permite cargar el repositorio de la herramienta CASE con datos provenientes de otros sistemas, o bien generar a partir de la propia herramienta esquemas de bases de datos, programas, etc.

Una herramienta CASE puede ser para nosotros un primer y firme paso para un desarrollo de software más libre de fallas y una útil herramienta para el establecimiento de nuestro objetivos.

Categorías de herramientas CASE

Como sucede en otras áreas de la informática, la tecnología CASE emplea procesos que puede resultar a veces confusa.

Atendiendo a la fase del ciclo de vida que soportan, se suele distinguir como:

- **CASE frontales** ("*front-end*") o **superiores** ("*Upper CASE*"); abarcan las primeras fases de análisis y diseño.
- **CASE dorsales** ("*back-end*") o **inferiores** ("*Lower CASE*"); cuyo objetivo suele ser el diseño detallado y la generación de código.

Y del corto ciclo de vida que soportan las CASE frontales y dorsales surgen los denominados:

- **ICASE** (*Integrated CASE*) o **CASE integrados**; a las herramientas que engloban ambos aspectos
- **IPSE** (*Integrated Programming Support Environment*) o **Ambiente Integrado de Soporte para la Programación**; a aquellas que, además, incluyen componentes para la gestión de proyectos y la gestión de configuración.

A continuación se presentan de manera muy resumida las categorías de herramientas CASE más frecuentes.

A) Herramientas de análisis y diseño

Dentro de las herramientas de análisis y diseño, destacan las herramientas que permiten crear y modificar diagramas E/R, diagramas de flujo de datos, diagramas de estructura de cuadros, diagramas de clases, etc.

También son muy importantes las herramientas de prototipado como los diseñadores de pantallas, generadores de menús, generadores de informes y lenguajes de especificación ejecutables.

Un aspecto a destacar es la capacidad de análisis y verificación de especificaciones que soporta la herramienta, no sólo sintáctica sino también semántica, por ejemplo, la capacidad de normalizar un diagrama de datos (usualmente hasta tercera forma normal).

B) Generación de código

A partir de las especificaciones del diseño se puede generar código tanto para los programas (por ejemplo, en C, COBOL, C++ o JAVA) como el relativo a los esquemas de bases de datos (sentencias de definición en SQL). Actualmente, las herramientas CASE ofrecen interfaces con diversos lenguajes de cuarta generación para la construcción de sistemas de manera rápida.

C) Documentación

Las herramientas CASE también soportan la creación automatizada de un conjunto muy variado de documentación (obtenido a partir de la información almacenada en el directorio) que va desde la descripción textual de un pseudocódigo hasta diagramas más o menos complejos. Aquí podemos encontrar aplicaciones como el MS Project o la utilización de formatos de MS Word.

D) Herramientas de prueba

Las herramientas de prueba se conocen también por las siglas CAST (*Computer Aided Software Testing*) Asistente Computarizado para Pruebas de Software, y es un área bastante reciente dentro de la tecnología CASE.

E) Herramientas de gestión de configuración

En entornos de desarrollo complejo, especialmente si se integran diversas herramientas de ingeniería de software, se hace imprescindible la incorporación de una herramienta capaz de gestionar la configuración de los sistemas, las distintas versiones de sus componentes, etc.

F) Herramientas de ingeniería inversa

Dentro de este apartado destacan diversas herramientas, como las que llevan a cabo:

- **Ingeniería inversa de datos:** Son capaces de extraer la información del código fuente que describe la estructura de los elementos de datos; construyendo así diagramas OO o E/R partiendo de esquemas relacionales, en red o, incluso, archivos.
- **Ingeniería inversa de procesos:** Estas permiten aislar la lógica de las entidades, y las reglas del negocio a partir del código.
- **Reestructuración de código fuente:** Modifican su formato o implantan un formato estándar.
- **Redocumentación:** Permiten generar diagramas a fin de que se comprenda mejor el código.

- **Análisis de código:** Cuyas funcionalidades van desde la identificación automática del código fuente hasta la posibilidad de ir visualizando dinámicamente las llamadas del mismo.

II.9. Consideraciones primordiales en el desarrollo de un sistema de Base de Datos

Como Murphy señala en una de sus conocidas leyes "Si una cosa puede fallar, fallará; si dos cosas pueden fallar, las dos fallaran; y si tres cosas pueden fallar, las tres fallaran" de este escrito es que se puede partir al estudio de importantes y diversas consideraciones que hay que tomar en cuenta en la construcción o implementación de cualquier sistema y muy en especial en el de las bases de datos, dada la repercusión que generalmente éstas tienen en la empresa; por ello se describen a continuación una serie de puntos que (a simple vista algunos de ellos pudiesen parecer insignificantes) llegan a ser clave en la detección oportuna de posibles problemas a los que se pudiesen enfrentar de no ser detectados oportunamente:

- Estudio previo, cargos y responsabilidades
- Política
- Fijación de objetivos (estudio de viabilidad)
- Evaluación previa de medios y costos
- Aprobación de una estructura orgánica
- Plan de trabajo detallado
- Especificaciones de las necesidades de equipo físico y lógico
- Especificación de diseño escrita
- Productos del diseño de la interfaz externa

II.9.1. Estudio previo, cargos y responsabilidades

El estudio previo a una implementación es labor del gerente del proyecto que es el encargado de planear, asignar y fijar responsabilidades para el desarrollo de la implementación, medir el avance continuamente y ajustar el plan con base en sus mediciones.

"Mientras que los analistas, diseñadores y desarrolladores individuales son responsables del dominio y la ejecución de las técnicas, el gerente de proyecto sirve como la fuerza conductora para ordenar las tareas en una metodología coherente a fin de satisfacer los objetivos del proyecto".[22]

II.9.2. Política

En este punto hay que tener claro que las políticas son todas aquellas actividades que realiza la empresa para la toma de una decisión ya sea sobre una persona o activo; son también todos pasos que ésta sigue para la realización de alguna actividad en especial. Siempre es recomendable conocer las políticas con que la empresa realiza sus actividades, ya que en ocasiones estas pueden limitar nuestras acciones desde los procedimientos con que se administran sus equipos e instalan aplicaciones, hasta el tiempo en que se pueda permanecer trabajando en la empresa o si hay que hacerlo bajo alguna supervisión o el llenado de ciertos formatos para ejercer ciertas acciones.

II.9.3. Fijación de objetivos (estudio de viabilidad)

La fijación de objetivos va de acuerdo al avance que vaya alcanzando el proyecto y se haya estipulado desde el principio del proyecto que puede ser:

[22] Ruble, David A.. Análisis y diseño practico de sistemas cliente/servidor con GUI. Pagina 9. Ed. Prentice Hall. México. 1997.

levantamiento de información (políticas y análisis técnico), estudio de viabilidad, implementación, pruebas y puesta en marcha.

El llamado análisis previo, *estudio de oportunidad* o *viabilidad*, debe preceder a cualquier operación de concepción o diseño de una base de datos; en ella se ha de concretar la voluntad de los directivos de abordar el proyecto, definiendo unos objetivos claros y concretos que sirvan de pauta en todo el desarrollo.

El estudio de viabilidad es de muy corta duración y limitado a la intervención de los técnicos, sin embargo, es fundamental para conseguir que el nuevo sistema de información, articulado alrededor de la base de datos, una vez puesto en marcha se integre en el organismo, adaptándose a sus objetivos y prestando el servicio que de él se esperaba cuando se decidió su implantación.

Los técnicos implicados en el proceso de creación de un sistema de base de datos deben tener presente, que si no cuentan con el pleno apoyo de los directivos (los cuales no sólo han de conocer que se va a abordar en el proyecto, sino que también han de comprender el alcance y significado del mismo), será mejor hacer una pausa en el desarrollo de la aplicación, aplazándola para otra ocasión más propicia, ya que si no existe una decidida voluntad de la organización en su conjunto, impulsada por sus directivos, de llevar a buen fin el sistema, aumentan las probabilidades de fracaso, debido a que los costos del proyecto, a menudo muy elevados, no estarían justificados.

II.9.4. Evaluación previa de medios y costos

Una vez que la dirección ha tomado la decisión inicial de emprender las operaciones que conduzcan al establecimiento de un sistema de base de datos y ha definido los objetivos generales del proyecto, es preciso realizar una evaluación

aproximada de los medios y de los costos que requerirá la puesta en marcha del sistema.

Se tratará sólo de dar un orden de magnitud del costo global del proyecto, ya que será prácticamente imposible, sin un estudio más profundo del sistema que se va a desarrollar, hacer un análisis detallado de costos. Sin embargo, es imprescindible que a los responsables del organismo se les proporcione una cifra aproximada de los gastos que representará la implantación del sistema y su mantenimiento, así como de los medios, en especial del personal, que han a ser precisos.

II.9.5. Aprobación de una estructura orgánica

Antes de comenzar el desarrollo del sistema será preciso definir la organización de la unidad administrativa que tendrá la responsabilidad de la gestión y control de la base de datos, así como determinar la estructura y los componentes del equipo encargado del desarrollo.

Las funciones del administrador de la base de datos, su responsabilidad respecto al contenido de la base, la actualización de la misma, la estandarización de la información, etc., son aspectos fundamentales que han de ser considerados desde un principio y que pueden ser decisivos para conseguir que el proyecto llegue a buen fin.

No es posible establecer una normativa general que determine cuál es la mejor organización que lleve a definir de forma óptima las funciones y responsabilidades del administrador de la base, porque en cada caso, las características de la organización y del entorno condicionan como es natural, la decisión.

Sin embargo, es necesario destacar la necesidad de establecer, como fase previa a la concepción de la base de datos, la organización de la misma y el equipo que va a intervenir en su creación. Ésta es una responsabilidad de la dirección, la cual deberá decidir y aprobar, de un modo formal, la estructura organizativa del equipo, que tendrá a su cargo la creación de la base de datos así como de la unidad que se encargaría posteriormente de su soporte y funcionamiento.

Las líneas generales de quién y cómo va a utilizar y actualizar la base de datos, también serán aprobadas por la dirección, y posteriormente el administrador de la base de datos que deberá redactar una detallada normativa que regule estos aspectos como lo pueden ser tiempos y permisos o privilegios.

II.9.6. Plan de trabajo detallado

Obtenida la aprobación por parte de la dirección para emprender el proyecto, será preciso crear un plan de trabajo detallado en el que se especifiquen las distintas fases, plazos y medios que requerirán cada una de ellas.

En general, da mejores resultados prácticos el desarrollo del sistema de forma gradual, sin intentar integrar a la vez todas las aplicaciones en la base; de esta forma, se consiguen varios objetivos:

a) La propia experiencia va mostrando los errores cometidos y la forma de solucionarlos.

b) Por otra parte, una evolución, en lugar de una revolución, permitirá la adaptación y formación de los usuarios, tanto informáticos (analistas y programadores) como no informáticos, los cuales no tendrán que enfrentarse bruscamente, y todos a la vez, con un sistema que, al cambiar sus hábitos de trabajo, siempre traerá dificultades y despertará recelos.

c) Se obtendrán resultados prácticos en menores plazos, lo que suele ser muy conveniente de cara a los directivos y a los usuarios.

Hay que tener en cuenta que la anterior afirmación respecto a la conveniencia de una implantación gradual no significa que no haya que hacer un estudio global detallado y a fondo de toda la información que formará parte de la base de datos.

Para realizar esta planificación es muy importante contar con el apoyo de los usuarios, ya que en varias de las etapas será obligada su participación, por lo que no se puede prescindir de su opinión.

En la Fig. II.7 se presentan gráficamente estas actividades iniciales de la acción de una base de datos que corresponden a la estrategia del proceso.

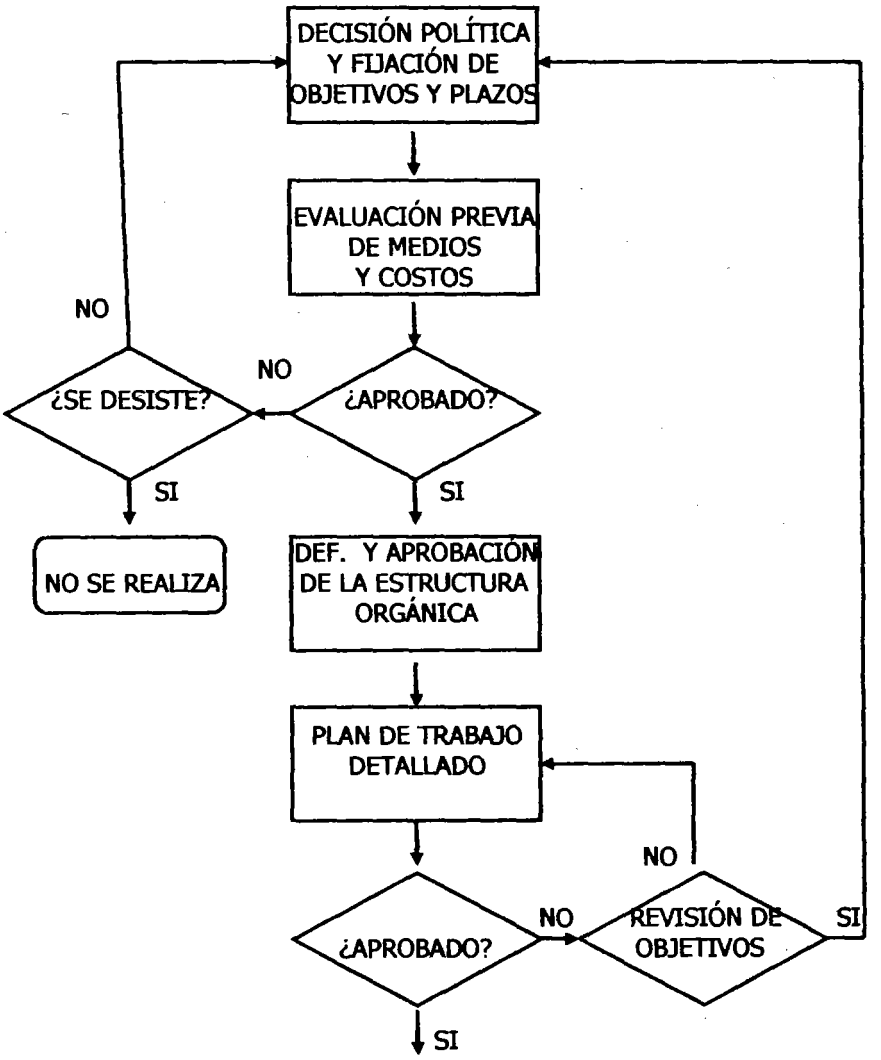

Fig. II.7. Actividades de la fase de estudio previo

El plan de trabajo detallado ha de ser aprobado por la dirección antes de pasar a la siguiente etapa, y su rechazo puede obligar bien a una reelaboración del

mismo o incluso, a una vuelta a la etapa inicial de estudio de oportunidad, reconsiderando los objetivos, medios y plazos, lo cual generalmente tiene repercusiones en términos de tiempos y contratos lo que puede desestabilizar todo el plan de trabajo planteado al inicio del proyecto y, por tanto, su entrega satisfactoria.

II.9.7. Especificaciones de las necesidades de equipo lógico y físico

Una vez determinados los requisitos y características que el sistema definido en la fase de concepción necesitará para su puesta en marcha, será preciso evaluar las exigencias en cuanto a equipo, en especial respecto al SGBD y a las características de la computadora (memorias principal y secundarias, capacidad de proceso, etc.).

En general, el organismo dispondrá de un equipo que será el que se utilice para implementar el sistema (en ocasiones será preciso acudir a una ampliación); por otra parte, no hay diferencia en la metodología para el problema general de la evaluación, selección y adquisición de equipos.

En cuanto a la selección del SGBD, aunque a veces el problema viene resuelto por condicionantes externos (tipo de máquina, costos, etc.) que obligan a utilizar un determinado producto, en general, se deberá proceder a realizar un estudio de los SGBD existentes en el mercado, de sus características y de las posibilidades que ofrecen, para poder elegir aquel que mejor se adapte a los requisitos específicos del sistema de información que se está diseñando.

II.9.8. Especificación de diseño escrita

Toda implementación requiere de una especificación escrita la cual debe contener el historial del proceso llevado de principio a fin marcando en ella cargos, tiempos e imprevistos, así como toda la documentación obtenida durante el levantamiento de información (catálogos de producto, clientes, etc.), políticas y trabajo de implementación, así como todo aquello que puede ser de relevancia en el futuro.

Estas especificaciones de diseño suelen hacerse por duplicado para poder dejar una copia al encargado del mantenimiento futuro de la base de datos y otra para el implementador como seguridad para casos de soporte o fallas.

La especificación de diseño escrita sirve a varios propósitos. Proporciona un método para documentar y revisar las ideas. Aun la especificación de diseño más pequeña proporciona un gran avance hacia el cumplimiento de estándares para la apariencia y sensación a lo largo de la aplicación.

La disposición, descripción, diagramas de navegación de las ventanas y flujo son herramientas excelentes para la validación de la propuesta de diseño con los usuarios.

La especificación de diseño también es una herramienta de administración poderosa. Permite que la aplicación se divida entre varios programadores para su construcción a un nivel de detalle que es mucho menor para lograr un diseño más coherente.

Una vez que existe una especificación de diseño escrita, el esfuerzo de programación puede seguirse dividiendo, aún dentro de la misma aplicación, para asignarlo a más de un programador. Con un diseño escrito se pueden asignar

varios programadores para que construyan ventanas específicas y sus funciones subyacentes dentro de la misma aplicación. El código se integra fácilmente, debido a que están trabajando a partir de una especificación coherente. El resultado final es una entrega más rápida que genere un producto terminado de más alta calidad como se muestra en la Fig. II.8 en que se toma el todo de un proyecto y éste se divide en distintas personas con capacidades distintas en el desarrollo de software pero con un fin especifico que es el desarrollo completo de la aplicación.

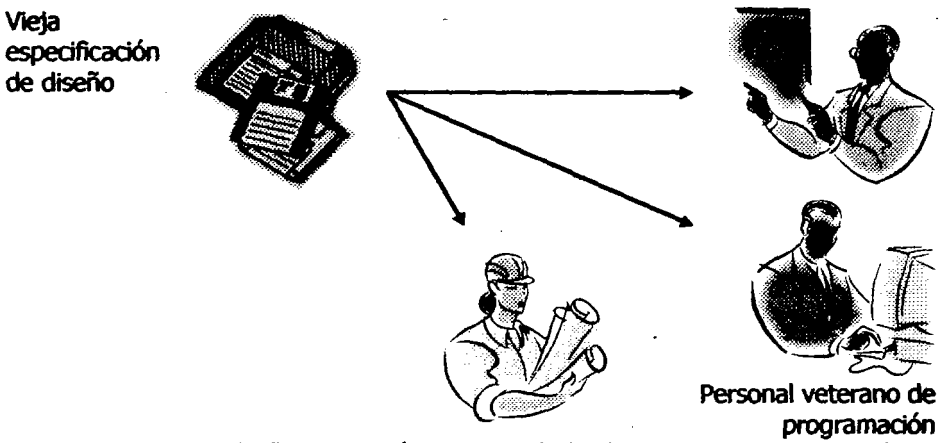

Fig. II.8. Un diseño de aplicación grande puede distribuirse entre varios programadores para ser codificado

Sin especificación escrita, el desarrollador debe diseñar y codificar al mismo tiempo, forzando al administrador a dividir el esfuerzo de programación en fragmentos más grandes. Esto puede causar cuellos de botella serios, debido a que el diseño de la aplicación permanece en la cabeza del programador. Sin una especificación escrita, el administrador es incapaz de añadir más programadores al equipo para agilizar la construcción en un punto en que se está acabando el tiempo de entrega del proyecto. Si el administrador trata de dividir la carga de programación a un nivel de detalle menor que el adecuado para el diseño, sin tener especificación escrita, el código resultante puede requerir la participación y el

esfuerzo de todos los recursos disponibles para volverlo a ensamblar y obtener una aplicación coherente.

La especificación de diseño escrita permite que la aplicación sea probada en forma independiente. Sin la especificación, cualquier intento para probar el producto terminado está en gran desventaja. Una buena especificación de diseño no requiere ninguna herramienta CASE novedosa. Bastará cualquier procesador de palabras y una herramienta de dibujo simple. Una buena herramienta CASE puede hacer más fácil la vida, pero una que no se adecue correctamente a las tareas específicas puede complicar el proyecto.

II.9.9. Productos del diseño de la interfaz externa

Los componentes de un diseño de interfaz externa para una interfaz gráfica de usuario incluyen:

- **Panorama del sistema:** Una descripción escrita que orienta al lector sobre el objetivo y la función del sistema completo.

- **Panorama de la aplicación:** Una descripción escrita para cada aplicación que está contenida dentro del sistema (por ejemplo, captura de pedidos, facturación, reporte de ventas, manejo de inventarios) que define las características disponibles dentro de la aplicación.

- **Diagrama de navegación de ventanas:** Para cada aplicación dentro del sistema, un diagrama de navegación de ventanas declara cuáles ventanas están disponibles y muestra las rutas de navegación posibles entre ellas.

- **Disposición de ventanas:** Para cada ventana del diagrama de navegación, una disposición de ventana muestra la manera en que ésta aparecerá ante el usuario.

- **Descripción de la ventana:** El texto que acompaña a cada disposición de ventana define claramente la función y características de ésta, en forma tal que un usuario potencial pueda comprender el comportamiento del diseño.

- **Miniespecificación de ventana:** La especificación técnica de la ventana define el comportamiento para la apertura y cierre de la ventana y la activación y ejecución de cada botón, control y elemento de menú.

- **Especificación de campo:** Define los campos y ediciones asociadas para todos los datos que aparecen en la ventana. La especificación de campo debe incluir una miniespecificación sobre la manera en que se adquieren los datos, listando nombres de tablas, nombres de columnas, indicando como unir tablas y describiendo la manera de aplicar cualquier criterio de selección.

El diseño de una buena interfaz de usuario aunque si bien no representa la parte fundamental de un sistema, si lo puede ser para la empresa y las personas que lo utilizan ya que de un ambiente amigable y bien estructurado depende mucho la aceptación de la aplicación por ello es que el diseño de la misma debe planearse con mucho detalle como se muestra en la Fig. II.9.

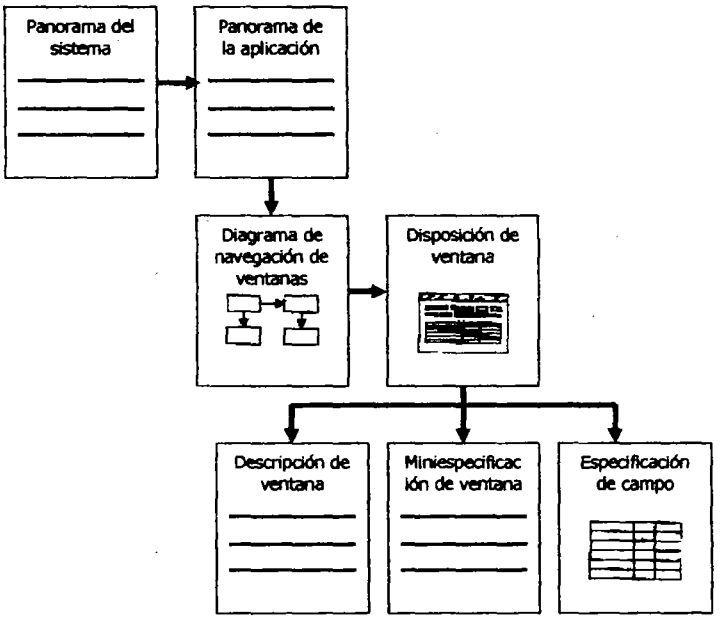

Fig. II.9. Diseño de la interfaz estudiada a detalle

Gracias a lo anteriormente descrito es que podemos darnos cuenta de que el diseño de un sistema, y en nuestro caso el de una Base de Datos, no es sólo un trabajo de una parte ya que muchas de las decisiones que hemos de ir tomando dependen el equipo con el que contemos, presupuestos y de muchas personas con sus propias ideas y las cuales debemos respetar, así como la importancia que tiene el respaldo de una especificación escrita que no sólo nos apoye como documentación y sustento inmediato a nuestros trabajo sino de igual forma viendo a futuro alguna implementación extra al sistema.

Capítulo III

Lenguajes para Base de Datos

Este capítulo estudia a grandes rasgos los lenguajes en que son escritos los sistemas de Base de Datos.

Desde tiempos remotos, los datos han sido registrados por el hombre en algún tipo de soporte (papel, piedra, madera, etc.) a fin de que quedara constancia de un fenómeno o idea. Los datos han de ser interpretados (incorporándoles significado) para que se conviertan en información útil.

Cuando utilizamos el lenguaje natural y decimos, por ejemplo, que una persona ha nacido en 1965; el dato "1965" va acompañado de su interpretación (año de nacimiento de una cierta persona); sin embargo, en la informática, desde sus inicios, se separó el dato de su significado. Por ello, a fin de facilitar la interpretación de los datos, surgen los modelos de datos como Instrumentos que ayudan a incorporar significado a los datos

III.1. Introducción al lenguaje

EL lenguaje es parte importante en el desarrollo de una aplicación informática cualquiera que ésta sea, ya que éstos cuentan con características especiales dependiendo de su propósito, que puede ir desde aplicaciones lógicas, matemáticas, de manejo de información como lo es una base de datos o una combinación de ellas; por ello es importante conocer en nuestro caso particular las características con que cuentan los lenguaje para el desarrollo de un sistema de Base de Datos.

III.2. Objetivos de un sistema de gestión de Base de Datos

Los principales objetivos de los sistemas de gestión de base de datos se resumen en:

- Independencia física
- Independencia lógica
- Manipulación de los datos por no informáticos
- Eficacia de los accesos a los datos
- Administración centralizada de los datos
- No redundancia de los datos
- Coherencia de los datos
- Compatibilidad de los datos
- Seguridad de los datos

En la realidad, estos objetivos sólo se alcanzan parcialmente.

Independencia física

Un dato elemental describe un suceso atómico del mundo real. Por ejemplo, 10.000 Pts. es un dato elemental. Los grupos de datos elementales que existen en el mundo real tienen su propio sentido independientemente del mundo informático. Más concretamente, existen unas relaciones intrínsecas entre unos datos elementales y unas reglas de controles pertinentes para utilizar esos datos.

Frecuentemente, los datos elementales del mundo real se agrupan para describir las entidades y las asociaciones entre entidades que son directamente perceptibles en el mundo real. Aunque dos grupos de trabajo agrupen, por regla general, de distinta manera datos elementales, la empresa puede definir una estructura distinta de datos a ellos a pesar de contenerlos. A esta agrupación se le puede considerar como una integración de los intereses particulares de cada grupo de trabajo. Obedece a unas reglas que traducen lo esencial de las propiedades de los datos elementales del mundo real.

En contrapartida, la estructura del almacenamiento de los datos pertenece al mundo de los informáticos y sólo tiene sentido en el universo del sistema informático. Se trata de una agrupación particular de los datos en artículos, ficheros, caminos de acceso (organizaciones y métodos de acceso siendo de éstos ficheros los modos de ubicación de los artículos según sus propios criterios de clasificación). Esta agrupación propia del mundo informático debe basarse en consideraciones relacionadas con las prestaciones y la flexibilidad de los accesos.

Uno de los objetivos primordiales de los SGBD es conseguir la independencia entre las estructuras de almacenamiento y las estructuras de datos del mundo real. Se trata pues de poder definir la agrupación de los datos elementales en el sistema informático, independientemente de la agrupación realizada en el mundo real, teniendo en cuenta solamente criterios de prestaciones y flexibilidad de los accesos.

Las ventajas de este tipo de independencia, llamada independencia física, se comprenden fácilmente cuando se consideran los inconvenientes de la no independencia física. La no independencia implicaría que la forma en la que están organizados los datos en la memoria secundaria fuese la imagen directa de la organización canónica de los datos en el mundo real. Para poder conservar las posibilidades de optimización de las prestaciones vitales para los sistemas informáticos, los conceptos de métodos de acceso, modos de ubicación, criterios de clasificación, encadenamientos, codificación de los datos, etc., deberían aparecer directamente en el mundo real, y de rechazo en las aplicaciones. Todo cambio informático repercutiría en la vida de una empresa e implicaría una reconstrucción de las aplicaciones. Desde luego, esto no es práctico y de aquí surge la necesidad de independencia entre las estructuras de almacenamiento y los datos del mundo real.

Independencia lógica

Se acaba de admitir la existencia de una estructura canónica que describe completamente la semántica inherente a los datos en el mundo real. Esta estructura es una síntesis de los intereses particulares de cada grupo de trabajo. En consecuencia, cada grupo de trabajo encargado de desarrollar una aplicación debe poder agrupar los datos con arreglo a sus necesidades, para formar las entidades y las asociaciones. Además, cada grupo debe poder concentrarse en aquellos elementos que constituyen su centro de interés, es decir, que debe poder trabajar conociendo solamente una parte de la semántica de los datos y viendo solamente una parte de los datos.

Por ejemplo, a partir de las agrupaciones de datos:

VEHICULO (No MAT, MARCA, TIPO, COLOR)
PERSONA (No SS, APELLIDO, NOMBRE)
PROPIETARIO (No SS, No MAT, FECHA-COMPRA)

un grupo podrá ver las personas que poseen coches, es decir, una agrupación del tipo:

PERSONA (No SS, NOMBRE, APELLIDO, No COCHE)

mientras que otro verá solamente los vehículos vendidos en una fecha determinada, es decir, una agrupación del tipo:

COCHE (No COCHE, TIPO, MARCA, FECHA VENTA)

Resulta pues conveniente permitir una cierta independencia entre los datos vistos por las aplicaciones y la estructura canónica de los datos dentro de la

empresa. Las ventajas de este tipo de independencia, frecuentemente llamada independencia lógica, son las siguientes:

- Permitir que cada grupo de trabajo vea los datos tal y como le interesan.
- Permitir la evolución de la visión de cada grupo de trabajo y de la visión canónica de la empresa, sin tener que cuestionar, al menos en cierta medida, la visión particular de cada grupo de trabajo; así pues se debe poder añadir grupos de datos elementales, suprimir otros, añadir y suprimir grupos de asociaciones, redefinir las agrupaciones, etc. en cada visión particular, pero también en la visión canónica de los datos sin que sea necesario modificar la mayoría de las aplicaciones.

Manipulación de los datos por los no informáticos

Dado que los no informáticos ven los datos, deben poder manipularlos, es decir, consultarlos y actualizarlos. Más concretamente, si se han alcanzado los objetivos 1 y 2, los grupos no informáticos verán los datos independientemente de cómo se haya realizado su implantación en máquina. Por este motivo, podrán manipular los datos por medio de lenguajes no sujetos a procedimientos, es decir, describiendo los datos que desean consultar (o actualizar) sin describir la forma en que hay que consultarlos (o actualizarlos) que es propia de la máquina.

Eficacia de los accesos a los datos

Este objetivo tiene dos aspectos. En primer lugar, hay que tener en cuenta que determinados accesos a los datos son realizados por informáticos expertos, por ejemplo, programadores de sistemas, que conocen la estructura del almacenamiento de los datos y que deben contar con unos lenguajes de manipulación de datos muy eficaces que permiten acceder rápidamente a los datos

a través de los caminos de acceso definidos en la estructura de almacenamiento, Se podrán utilizar estos lenguajes dentro de programas clásicos escritos en ensamblador o en lenguajes de más alto nivel como COBOL, PL1 o PASCAL..

En segundo lugar, tanto los no informáticos como los informáticos que desconocen la estructura de almacenamiento, deben poder utilizar un lenguaje que ofrezca salidas sencillas y cuyo resultado sea muy eficaz, especialmente a nivel de los accesos a discos que siguen siendo el cuello de botella fundamental de los SGBD.

Administración coherente de los datos

Unas funciones esenciales de los SGBD son la definición de las estructuras de almacenamiento y las estructuras de los datos y el seguimiento de sus evoluciones. Estas funciones reciben el nombre de administración de los datos. Con el fin de permitir un control eficaz de los datos, resolver los conflictos entre distintos puntos de vista no siempre coherentes, poder optimizar los accesos a los datos y la utilización de los medios informáticos, es esencial responsabilizar de estas funciones a un pequeño grupo de personas altamente cualificadas. Este es el motivo por el que la administración de los datos suele estar centralizada. El objetivo es únicamente el de permitir una administración coherente y eficaz de los datos.

No redundancia de los datos

En los sistemas clásicos de ficheros no integrados, cada aplicación tiene sus propios datos. Esto conduce generalmente a numerosas duplicaciones de los datos con un importante desperdicio de recursos humanos para capturar y

actualizar los mismos datos varias veces, además de la consiguiente pérdida de espacio de la memoria secundaria.

En una base de datos, los ficheros más o menos redundantes se integran en un solo fichero compartido por las distintas aplicaciones. La administración coherente de los datos debe velar por la no duplicación física de los datos a fin de evitar las actualizaciones múltiples.

Coherencia de los datos

Aunque el objetivo descrito en el apartado anterior era el de evitar las redundancias de los datos, éstos se encuentran sujetos a ciertas reglas, ya sea a nivel del dato elemental, ya sea a nivel de un conjunto de datos en el que pueden existir determinadas dependencias entre datos. Por ejemplo: un sueldo puede estar comprendido entre $ 5.000.00 y $ 50.000.00 pesos. Otro ejemplo: un dato que represente el número de peticiones de un cliente debe corresponder al número de peticiones de la base. Un sistema de gestión de base de datos debe vigilar que las aplicaciones respeten estas reglas durante las modificaciones de los datos asegurando así la coherencia de los mismos.

Compartición de los datos

Su objetivo es permitir que las aplicaciones compartan los datos de la base no sólo por una consulta sino también simultáneamente. Una aplicación debe poder acceder a los datos como si fuese la única que los está utilizando sin esperar y también sin necesidad de saber que cualquier otra aplicación puede modificarlos concurrentemente.

Seguridad de los datos

Los datos deben estar protegidos contra los accesos no autorizados o mal intencionados. Deben existir mecanismos adecuados para controlar o retirar las autorizaciones de acceso de cualquier usuario para cualquier conjunto de datos. Los derechos de acceso pueden depender igualmente del valor de los datos o de los accesos anteriormente realizados por el usuario.

Por ejemplo, un empleado podrá conocer el sueldo de las personas que dependen de él pero no el del resto de empleados de la empresa...

III.3. Herramientas de desarrollo (Lenguajes de cuarta generación)

El proceso evolutivo de los SGBD ha tenido una considerable influencia en la aparición y evolución de herramientas de desarrollo, entre ellas, los lenguajes de cuarta generación (L4G). En este capítulo se presenta una rápida visión de la evolución de los lenguajes hasta llegar a los L4G, analizando sus ventajas e inconvenientes y describiendo brevemente las características de estos y de otros tipos de lenguajes, como orientados a objetos y visuales.

Evolución de los lenguajes de programación

El análisis de la evolución de los lenguajes de programación tiene interés para comprender las características de las herramientas de desarrollo de cuarta generación.

Lenguajes de primera y segunda generación

Aunque sea sólo por motivos históricos, es necesario incluir en la evolución de lenguajes de programación los lenguajes de primera generación (código máquina) y de segunda generación (ensamblador) que se consideran aplicables a aquellas partes de un sistema que requieren una gran velocidad, pero cuya utilización no se encuentra en la actualidad demasiado extendida.

Al utilizar un lenguaje ensamblador, el código fuente se ensambla, línea a línea, en código máquina, informando al programador de si existe algún problema con dicho código. Además, los lenguajes de segunda generación permiten combinar diferentes archivos de código fuente en un solo programa ejecutable utilizando un "enlazador" (*linker*).

Lenguajes de tercera generación

Debido a la espectacular evolución de los rendimientos del hardware respecto a su precio, los lenguajes de segunda generación perdieron su protagonismo en función de lenguajes de alto nivel como COBOL, C, PASCAL, FORTRAN, etc. En estos lenguajes, que son más comprensibles para el programador, una sentencia se traduce en varias instrucciones de máquina.

Con esta generación empiezan a aparecer realmente los entornos de programación, que permiten compilar y depurar programas, incluso en máquinas distintas de la destino.

En general podemos distinguir dos tipos de traducción del código fuente a código máquina:

- **De una sola vez;** como una tarea preparatoria llevada a cabo por un compilador.

- **En cada ejecución;** por un intérprete, el cual va traduciendo cada sentencia del programa fuente a código de máquina, ejecutándolo a continuación y advirtiendo si encuentra algún error, en cuyo caso permite la corrección del mismo y la continuación del proceso.

La compilación y el enlace pueden llegar a ser lentos, como consecuencia de la corrección de los errores introducidos en el código, pero una vez que se ha compilado con éxito, el programa es más rápido y presenta un código más compacto que un intérprete.

En un compilador podemos distinguir:

- Un analizador sintáctico.
- Un analizador semántico.
- Un analizador lexicográfico.
- Un optimizador.
- Traductores de código fuente a lenguaje máquina.

Un intérprete, por su parte, suele aparecer integrado con un editor del lenguaje así como con un generador de código máquina.

Lenguajes de cuarta generación

Aunque existen diversas definiciones de lenguaje de cuarta generación (L4G) y más de trescientos productos que se engloban en esta categoría, todas ellas coinciden en que un L4G se caracteriza por ser independiente lo que permite al usuario preocuparse de "QUÉ" sin necesidad de especificar "CÓMO" hacerlo. En

definitiva se trata de especificar el resultado deseado más que las acciones necesarias para obtener el resultado.

Tradicionalmente se les clasifica en dos grandes grupos:

- **Lenguajes para usuario final;** lo que se ha denominado centro de información o infocentro (*information center*), que nacieron con la idea de independizar a los usuarios del control del departamento de informática. Se centran sobre todo en la facilidad de uso y flexibilidad.

- **Lenguajes para el informático (analista/programador);** es decir, para el centro de desarrollo (*development center*), cuyo fin es facilitar el desarrollo de aplicaciones sofisticadas y permitir la construcción rápida de aplicaciones por medio del prototipado.

Este tipo de lenguajes surge al extenderse la utilización de los SGBD; "*los SGBD han desempeñado un importante papel en el proceso evolutivo de los lenguajes informáticos. De hecho, sin la independencia de los datos que ofrecen los SGBD, los lenguajes de cuarta generación nunca habrían existido*".[23]

La última evolución de los L4G en este sentido, es la aparición de lenguajes independientes del SGBD, que ofrecen una gran flexibilidad, permitiendo elegir el SGBD que mejor se adapte a cada caso. Sobre todo resultan útiles para fabricantes de aplicaciones que tienen que ejecutarse sobre diferentes SGBD.

[23] Cobb. In praise of 4GLs. En: *Datamation*, nro.15 de julio. Páginas 90. 1985.

Lenguajes de quinta generación

A veces se emplea el término lenguajes de quinta generación (L5G) para englobar los lenguajes que vienen apareciendo desde finales de los años ochenta y se caracterizan por su especialización en ciertas áreas de aplicación tales como:

- **Lenguajes funcionales.** Los matemáticos desde hace un buen tiempo están resolviendo problemas usando el concepto de función. Una función convierte ciertos datos en resultados. Si supiéramos cómo evaluar una función, usando la computadora, podríamos resolver automáticamente muchos problemas. Así pensaron algunos matemáticos, que no le tenían miedo a la máquina, e inventaron los lenguajes de programación funcionales. Además, aprovecharon la posibilidad que tienen las funciones para manipular datos simbólicos, y no solamente numéricos, y la propiedad de las funciones que les permite componer, creando de esta manera, la oportunidad para resolver problemas complejos a partir de las soluciones a otros más sencillos. También se incluyó la posibilidad de definir funciones recursivamente.

  Un lenguaje funcional ofrece conceptos que son muy entendibles y relativamente fáciles de manejar. El lenguaje funcional más antiguo, y seguramente el más popular hasta la fecha, es LISP, diseñado por McCarthy[24] en la segunda mitad de los años 50 del siglo XX. Su área de aplicación es principalmente la Inteligencia Artificial. En la década de los 80 del siglo pasado hubo una nueva ola de interés por los lenguajes funcionales, añadiendo la tipificación y algunos conceptos modernos de modularización y polimorfismo, como es el caso del lenguaje ML.

[24] Mcarthy, J. et al., LISP 1.5 "Programming Manual Cambridge" MA. MIT Press, 1965.

Programar en un lenguaje funcional significa construir funciones a partir de las ya existentes. Por lo tanto es importante conocer y comprender bien las funciones que conforman la base del lenguaje, así como las que ya fueron definidas previamente. De esta manera se pueden ir construyendo aplicaciones cada vez más complejas. La desventaja de este modelo es que resulta bastante alejado del modelo de la máquina de von Neumann y, por lo tanto, la eficiencia de ejecución de los intérpretes de lenguajes funcionales no es comparable con la ejecución de los programas imperativos precompilados. Para remediar la deficiencia, se está buscando utilizar arquitecturas paralelas que mejoren el desempeño de los programas funcionales, sin que hasta la fecha estos intentos tengan un impacto real importante.

- **Lenguajes lógicos.** Otra forma de razonar para resolver problemas en matemáticas se fundamenta en la lógica de primer orden. El conocimiento básico de las matemáticas se puede representar en la lógica en forma de axiomas, a los cuales se añaden reglas formales para deducir cosas verdaderas (teoremas) a partir de los axiomas. Gracias al trabajo de algunos matemáticos, de finales de siglo pasado y principios de éste, se encontró la manera de automatizar computacionalmente el razonamiento lógico --particularmente para un subconjunto significativo de la lógica de primer orden-- que permitió que la lógica matemática diera origen a otro tipo de lenguajes de programación, conocidos como lenguajes lógicos. También se conoce a estos lenguajes, y a los funcionales, como lenguajes declarativos, porque el programador, para solucionar un problema, todo lo que tiene que hacer es describirlo vía axiomas y reglas de deducción en el caso de la programación lógica y vía funciones en el caso de la programación funcional.

El PROLOG es el primer lenguaje lógico y el más conocido y utilizado. Sus orígenes se remotan a los inicios de la década de los 70 con los trabajos del grupo de A. Colmerauer[25] en Marsella, Francia. También en este caso, las aplicaciones a la Inteligencia Artificial mantienen el lenguaje vivo y útil.

En el caso de la programación lógica, el trabajo del programador se restringe a la buena descripción del problema en forma de hechos y reglas. A partir de ésta se pueden encontrar muchas soluciones dependiendo de como se formulen las preguntas (metas), que tienen sentido para el problema. Si el programa está bien definido, el sistema encuentra automáticamente las respuestas a las preguntas formuladas. En este caso ya no es necesario definir el algoritmo de solución. En programación lógica, al igual que en programación funcional, el programa, en este caso los hechos y las reglas, están muy alejados del modelo von Neumann que posee la máquina en la que tienen que ser interpretados; por lo tanto, la eficiencia de la ejecución no puede ser comparable con la de un programa equivalente escrito en un lenguaje imperativo. Sin embargo, para cierto tipo de problemas, la formulación del programa mismo puede ser mucho más sencilla y natural para un programador experimentado.

- **Lenguajes paralelos.** Cuando los procesadores cambiaron de tamaño y de precio, se abrió la posibilidad de contar con varios procesadores en una máquina y ofrecer el procesamiento en paralelo, es decir, procesar varios programas al mismo tiempo. Esto dio el impulso a la creación de lenguajes que permitían expresar el paralelismo. Finalmente, llegaron las redes de computadoras, que también ofrecen la posibilidad de ejecución en paralelo, pero con procesadores distantes, lo cual conocemos como la programación distribuida.

[25] Colmeraure, A., Roussel, P., " The Birth of PROLOG", ACM SIgPlan Notices, Vol. 28, No. 3, March 1993.

En resumen, el origen del concepto de paralelismo está en el deseo de aprovechar al máximo la arquitectura von Neumann y sus modalidades reflejadas en conexiones paralelas y distribuidas.

Históricamente, encontramos en la literatura soluciones conceptuales y mecanismos tales como: semáforos, regiones críticas, monitores, envío de mensajes (CSP), llamadas a procedimientos remotos (RPC), que posteriormente se incluyeron como partes de los lenguajes de programación en Concurrent Pascal, Modula, Ada, OCCAM, y últimamente en Java.

Lenguajes orientados a objetos

Aunque algunos expertos los consideran como L5G, la historia de los lenguajes orientados al objeto empezó a finales de la década de los sesenta con el lenguaje Simula, diseñado en el *Norwegian Computing Centre* para modelar y simular sistemas. Por otro lado, a principios de los setenta en el *Xerox Palo Alto Research Center* se desarrolló, dentro de las investigaciones orientadas al sistema Dynabook (computadora personal orientado a gráficos), el lenguaje Smalltalk.

A principios de los ochenta se empezó a trabajar en extensiones del lenguaje C (C++ y Objective-C), de importancia creciente debido a la difusión de los entornos UNIX.

A mediados de los ochenta apareció otro lenguaje importante en esta área, Eiffel, así como enfoques híbridos del tipo Object Pascal. A mediados de los noventa se han desarrollado extensiones orientadas al objeto de COBOL (OOCOBOL) y de Ada (Ada-95), a la vez que aparecen nuevos lenguajes como Dylan o Java. Hay que destacar el gran impacto de este último, debido a la explosión del fenómeno Internet.

También en la década de los noventa se modifican los L4G existentes, a los que se les incorporan varios de los principios del paradigma de la orientación a objetos, como pueden ser el encapsulamiento, la generalización o el polimorfismo.

Los entornos orientados al objeto se caracterizan por poseer, además de todos los elementos de los lenguajes de tercera generación, un visualizador o inspector (browser), que permite recorrer las diferentes jerarquías de objetos de las aplicaciones o bibliotecas.

Lenguajes "visuales"

Dentro de este apartado se pueden distinguir dos grandes grupos:[26]

- **Lenguajes de programación visual:** Son aquellos que tienen una "sintaxis visual", esto es, que algunos de los elementos de su gramática son gráficos (imágenes, formas, animaciones, etc.) e incorporan información especial (intersección, contenido, etc.) y atributos visuales (color, alineación, etc.). Ejemplos de este tipo son HI-VISUAL, ObjectWorld, Visual C, Visual Basic, etc.

- **Entornos visuales:** Son herramientas gráficas que se utilizan para crear, modificar y examinar programas (con ayuda del ratón) escritos en un lenguaje de programación textual, como puede ser el caso de VisualBasic.

Recientemente empiezan a aparecer lenguajes que combinan las facilidades visuales con la tecnología de objetos, y que se conocen por las siglas VOOPL Lenguaje de Programación Visual Orientada a Objetos (Visual Object Oriented Programming Language).[27]

[26] Burnett et al. (1997).
[27] Burnett et al. (1997).

Hay que destacar que la facilidad de aprendizaje de los lenguajes visuales ha determinado su acogida por los usuarios y su correspondiente difusión. "*Si se comparan las curvas de aprendizaje de estos lenguajes con los requeridos por los Sistemas de Gestión de Bases de Objetos (SGBO), los lenguajes tradicionales y las herramientas CASE, podemos entender la gran difusión de los primeros. También puede explicarse de esta manera parte del fracaso que tuvieron las herramientas CASE, debido a que presentan curvas de aprendizaje demasiado elevadas*".[28]

III.4. Componentes de un L4G

Como se ha señalado, la variedad de L4G es muy considerable ya que no existe un modelo de referencia o estándar que especifique los componentes que debe de poseer. A pesar de ello, típicamente se pueden encontrar en un entorno de este tipo:

- **Diccionario:** Es el núcleo del sistema que almacena información sobre las aplicaciones que se construyen (definición de campos, pantallas, informes, etc.), por lo que sirve de fuente de documentación del sistema. Además proporciona valores por defecto que permiten al usuario ahorrar tiempo y esfuerzo, reduciendo la cantidad de información que debe suministrar.

- **Entorno de desarrollo:** Compuesto de compilador y/o intérprete. En este entorno suele ser interesante que el sistema permita, apoyándose en los valores del diccionario de datos, controlar el comportamiento de las aplicaciones en tiempo de ejecución sin tener que recompilar ante modificaciones.

[28] Lewis, T. The big software chill. En: *IEEE Computer*, marzo. Páginas 12. 1996.

- **Diseñador de pantallas y menús:** Ofrece una serie de facilidades para que el usuario pueda directamente (de forma gráfica) especificar los formatos de pantalla, las posiciones de cada campo, etc. Suele complementarse con herramientas especializadas en gráficos en 2 y 3 dimensiones (gráficas, barras, etc.).

- **Generador de informes:** Es una herramienta para crear informes a partir de los datos almacenados en la Base de Datos. Se parece a un lenguaje de consultas en que permite al usuario hacer preguntas sobre la Base de Datos y obtener información de ella para un informe. Sin embargo, en el generador de informes se tiene un mayor control sobre el aspecto de la salida (especificar las cabeceras, pies, rupturas, cálculos, etc.). Se puede dejar que el generador determine automáticamente el aspecto de la salida o se puede diseñar ésta para que tenga el aspecto que desee el usuario final.

- **Generador de gráficos:** Es una herramienta para obtener datos de la Base de Datos y visualizarlos en un gráfico mostrando tendencias y relaciones entre datos. Normalmente se pueden diseñar distintos tipos de gráficos: barras, líneas, etc.

- **Interfaces:** Con distintos elementos del entorno, que permitan, por ejemplo, el intercambio de datos entre estaciones de trabajo y computadoras principales (servidores o *mainframe*).

- **Plantillas (*templates*) pre-elaboradas:** Permiten llevar a cabo ciertas operaciones, como leer un registro de un archivo, liberando al desarrollador de tareas repetitivas.

- **Generador de aplicaciones:** Es una herramienta para crear programas que hagan de interfaz entre el usuario y la Base de Datos. El uso de un

generador de aplicaciones puede reducir el tiempo que se necesita para diseñar un programa de aplicación. Los generadores de aplicaciones constan de procedimientos que realizan las funciones fundamentales que se utilizan en la mayoría de los programas. Estos procedimientos están escritos en un lenguaje de programación de alto nivel y forman una librería de funciones en las que el usuario puede especificar qué debe hacer el programa y el generador de aplicaciones es quien determina cómo realizar la tarea.

- **Generador de formularios o Componentes:** Es una herramienta interactiva que permite crear rápidamente formularios de pantalla para introducir o visualizar datos (ventanas de diverso tipo, botones, etc.) que se pueden incluir en las aplicaciones con sólo copiarlas con la ayuda del ratón. Los generadores de formularios permiten que el usuario defina el aspecto de la pantalla, qué información se debe visualizar y en qué lugar de la pantalla debe visualizarse. Algunos generadores de formularios permiten la creación de atributos derivados utilizando operadores aritméticos y también permiten especificar controles para la validación de los datos de entrada.

III.5. Ventajas e inconvenientes de los L4G

Los lenguajes de cuarta generación presentan una serie de características respecto a los de generaciones anteriores. A continuación se señalan las principales ventajas e inconvenientes que, en general, presenta esta tecnología, aunque debe tenerse en cuenta que es necesario realizar un análisis detallado de cada lenguaje y entorno en particular.

Ventajas

Un aspecto común a todos los entornos de cuarta generación es que son sistemas cada vez más "amistosos" (*user-friendly*) fáciles de usar, para lo cual se intenta llenar el vacío existente entre la forma de descripción del sistema inteligible para el usuario y la manera en que lo describe el informático. Así, por ejemplo, los L4G facilitan la navegación por la base de datos evitando que el usuario tenga que hacer las combinaciones (*join*) entre tablas.

Este aspecto va relacionado con la mejora de la productividad respecto a lenguajes tradicionales como COBOL. El aumento de productividad, que en los folletos de exposición de algunos productos se eleva hasta el 3000%, puede situarlo de forma realista en una proporción de 5 a 1. Hay que recordar, sin embargo, que esta ganancia en productividad se consigue a costa de eficiencia de procesamiento; teniendo en cuenta el continuo cambio que se observa desde hace años en la proporción del costo del equipo físico (hardware) y sus prestaciones en comparación al soporte lógico (software), resulta evidente que se favorece cada vez más el ahorro de "ciclos de memoria humana" en lugar de "ciclos de memoria de la computadora".

Otro importante aspecto de los L4G es facilitar el desarrollo mediante "maquetas", que permiten disminuir la desconfianza del usuario, hacer más fluida la comunicación entre el usuario y el desarrollador, contrastar creencias, confirmar la factibilidad de ciertos diseños, etc.

Incluso, "la mayor parte de los L4G permiten crear verdaderos "prototipos", que transforman el ciclo de vida de las aplicaciones, adoptándose un enfoque evolutivo e incremental; lo que tiene grandes efectos positivos tanto para el

producto software como para el proceso de desarrollo, y que ha tenido gran éxito en diferentes situaciones".[29]

Los entornos de cuarta generación ofrecen bastante funcionalidad "por defecto", que se puede incorporar en los sistemas que hay que desarrollar, lo cual no sólo mejora, como se ha señalado, la productividad, sino que también disminuye el número de errores al acceder a archivos y al manipular pantallas.

Una última ventaja que se puede destacar es la facilidad de construir aplicaciones portables y con grandes posibilidades de adaptación que suelen ofrecer los L4G.

Inconvenientes

A pesar de todas las ventajas enumeradas en el apartado anterior, cabe destacar también algunos inconvenientes que presentan todavía los L4G.

Por lo que respecta al diccionario, este suele ser propietario y no permite una fácil integración con otros componentes desarrollados en diferentes entornos. De hecho, es importante considerar la necesidad de integrar y coordinar el conjunto de herramientas que forman el entorno de cuarta generación, que debe permitir comunicar los resultados de una con otra sin que se tenga que invertir tiempo en este tipo de tareas. Un problema añadido en este sentido es la falta de mecanismos de sincronización automáticos entre el catálogo de los SGBD, los diccionarios de los L4G y los repositorios de herramientas CASE.

Ya se señaló el problema de la eficiencia, que no sólo viene condicionado por el propio entorno (que suele consumir gran cantidad de recursos de máquina),

[29] Gordon y Bieman (1994).

sino que depende muchas veces de que el L4G "conozca" como funciona el optimizador del SGBD para que pueda mejorar la velocidad de acceso.

En lo que respecta al desarrollo mediante prototipos, muchos L4G se prestan a un desarrollo "rápido y sucio" (*quick & dirty*), por lo que evidentemente se resiente la calidad de las aplicaciones. A esto se añade "la falta de buenas facilidades de depuración, que hacen que la etapa de prueba en los entornos de cuarta generación consuman mucho tiempo y muchas veces no sea tan clara como en los L3G, en los que se puede seguir más fácilmente el código. Por tanto, la utilización de L4G no debe evitar utilizar métodos de análisis y diseño rigurosos, así como técnicas de aseguramiento de calidad, seguridad y auditoria".[30]

Otra desventaja de "algunos L4G es su falta de flexibilidad a la hora de abordar algunas tareas de bajo nivel, lo que puede hacer necesario la utilización de lenguajes de tercera generación (como el lenguaje C) para construir una aplicación".[31] En efecto, "los L4G están mejor adaptados a aplicaciones intensivas en datos en las que se realizan operaciones sencillas sobre grandes volúmenes de datos, que a aplicaciones de proceso complejo sobre pocos datos. Muchos L4G ofrecen para este último caso salidas a código en L3G, aunque lo ideal es no tener que salir al L3G sino que el L4G sea completo en cuanto a operadores, gestión de errores, funciones de bajo nivel, etc".[32]

Por último, cabe destacar que los L4G desdibujan la frontera todavía existente en muchas organizaciones entre analista y programador, cobrando mayor importancia la existencia de analistas/programadores que diseñan las aplicaciones y las implementan con ayuda de un entorno de herramientas integradas.

---

[30] Piattini, M. Control interno y auditoria en un entorno de base de datos relacionales. En: ALIBase nro. 21 y 22 noviembre/diciembre. 1992.

[31] Leber, J.B. 4GLs and 3GLs: Leading Dual Lives. En: Database Programming & Design 4:1 enero. 1991.

[32] De Miguel, A., Piattini, M. Concepción y diseño de base de datos: del modelo E/R al modelo relacional. Madrid. Ed. Ra-Ma. 1999.

III.6. Lenguaje de definición de datos

Una vez finalizado el diseño de una Base de Datos y escogido un SGBD para su implementación, el primer paso consiste en especificar el esquema conceptual y el esquema interno de la Base de Datos, y la correspondencia entre ambos. En muchos SGBD no se mantiene una separación estricta de niveles, por lo que el administrador de la Base de Datos y los diseñadores utilizan el mismo lenguaje para definir ambos esquemas, es el lenguaje de definición de datos (LDD). El SGBD posee un compilador de LDD cuya función consiste en procesar las sentencias del lenguaje para identificar las descripciones de los distintos elementos de los esquemas y almacenar la descripción del esquema en el catálogo o diccionario de datos. El diccionario de datos describe los objetos de la Base de Datos.

Cuando en un SGBD hay una clara separación entre los niveles conceptual e interno, el LDD sólo sirve para especificar el esquema conceptual. Para especificar el esquema interno se utiliza un lenguaje de definición de almacenamiento (LDA). Las correspondencias entre ambos esquemas se pueden especificar en cualquiera de los dos lenguajes. Para tener una verdadera arquitectura de tres niveles será necesario disponer de un tercer lenguaje, el lenguaje de definición de vistas (LDV), que se utilizaría para especificar las vistas de los usuarios y su correspondencia con el esquema conceptual.

III.6.1. Lenguaje de manejo de datos

Una vez creados los esquemas de la Base de Datos, los usuarios necesitan un lenguaje que les permita manipular los datos de la Base de Datos: realizar consultas, inserciones, eliminaciones y modificaciones. Este lenguaje es el que se denomina lenguaje de manejo de datos (LMD).

Hay dos tipos de LMD:

- Procedurales y
- No procedurales.

Con un **LMD procedural** el usuario (normalmente será un programador) especifica qué datos se necesitan y cómo hay que obtenerlos. Esto quiere decir que el usuario debe especificar todas las operaciones de acceso a datos llamando a los procedimientos necesarios para obtener la información requerida. Estos lenguajes acceden a un registro, lo procesan y basándose en los resultados obtenidos, acceden a otro registro, que también deben procesar. Así se va accediendo a registros y se van procesando hasta que se obtienen los datos deseados. Las sentencias de un LMD procedural deben estar embebidas en un lenguaje de alto nivel, ya que se necesitan sus estructuras (bucles, condicionales, etc.) para obtener y procesar cada registro individual. A este lenguaje se le denomina lenguaje anfitrión. Las bases de datos jerárquicas y de red utilizan LMD procedurales.

Un **LMD no procedural** se puede utilizar de manera independiente para especificar operaciones complejas sobre la Base de Datos de forma concisa. En muchos SGBD se pueden introducir interactivamente instrucciones del LMD desde una terminal o bien embeberlas en un lenguaje de programación de alto nivel. Los LMD no procedurales permiten especificar los datos a obtener en una consulta o los datos que se deben actualizar, mediante una sola y sencilla sentencia. El usuario o programador especifica qué datos quiere obtener sin decir cómo se debe acceder a ellos. El SGBD traduce las sentencias del LMD en uno o varios procedimientos que manipulan los conjuntos de registros necesarios. Esto libera al usuario de tener que conocer cuál es la estructura física de los datos y qué algoritmos se deben utilizar para acceder a ellos. A los LMD no procedurales también se les denomina declarativos. Las bases de datos relacionales utilizan LMD no procedurales, como SQL (Structured Query Language) o QBE (Query-By-

Example). Los lenguajes no procedurales son más fáciles de aprender y de usar que los procedurales, y el usuario debe realizar menos trabajo, siendo el SGBD quien hace la mayor parte.

La parte de los LMD no procedurales que realiza la obtención de datos es lo que se denomina un lenguaje de consultas. En general, las órdenes tanto de obtención como de actualización de datos de un LMD no procedural se pueden utilizar interactivamente, por lo que al conjunto completo de sentencias del LMD se le denomina lenguaje de consultas, aunque es técnicamente incorrecto.

Todo lo anterior nos enseña la importancia de los lenguajes con que se trabaja en el desarrollo de una Base de Datos ya que de ello depende en mucho el tiempo en que se esté sentado frente a la computadora escribiendo las sentencias de conexiones, carga de datos y consultas a la base de datos. Y la tendencia actual del mercado que es orientar todo hacia un mismo lenguaje de cuarta generación siendo el más popular el SQL (Standar Query Lenguaje) de Microsoft y en torno al cual trabajan los sistemas de gestión de mayor difusión.

Capítulo IV

Manejadores de Base de Datos (DBMS)

Este capítulo estudia a grandes rasgos los manejadores (DBMS) en que se desarrollan sistemas de Base de Datos.

Un DBMS o también conocido como SGBD es un sistema computacional de propósito general que manipula la base de datos.

Los SGBD proporcionan los mecanismos físicos que permiten a varios usuarios tener acceso de forma rápida y eficiente a diferentes datos relacionados. También utiliza mecanismos de bloqueo para que la actualización de más de un usuario simultáneamente no afecte a los datos, además el SGBD ofrece al programador una serie de herramientas que facilitan la creación de software de aplicación.

IV.1. ¿Qué son los DBMS?

Es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Una de las ventajas del DBMS es que puede ser invocado desde programas de aplicación que pertenecen a Sistemas Transaccionales escritos en algún lenguaje de alto nivel, para la creación o actualización de las bases de datos, o bien para efectos de consulta a través de lenguajes propios que tienen las bases de datos o lenguajes de cuarta generación.

El sistema manejador de bases de datos es la porción más importante del software de un sistema de Base de Datos. Un DBMS es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

IV.2. Funciones principales de los DBMS

- Crear y organizar la Base de Datos.

- Establecer y mantener las trayectorias de acceso a la Base de Datos de tal forma que los datos puedan ser accedidos rápidamente.

- Manejar los datos de acuerdo a las peticiones de los usuarios.

- Registrar el uso de las bases de datos.

- Interacción con el manejador de archivos. Esto a través de las sentencias en DML al comando del sistema de archivos. Así el Manejador de Base de Datos es el responsable del verdadero almacenamiento de los datos.

- Respaldo y recuperación. Consiste en contar con mecanismos implantados que permitan la recuperación fácilmente de los datos en caso de ocurrir fallas en el sistema de Base de Datos.

- Control de concurrencia. Consiste en controlar la interacción entre los usuarios concurrentes para no afectar la inconsistencia de los datos.

- Seguridad e integridad. Consiste en contar con mecanismos que permitan el control de la consistencia de los datos evitando que estos se vean perjudicados por cambios no autorizados o previstos.

El DBMS interpreta las peticiones de entrada/salida del usuario y las manda al sistema operativo para la transferencia de datos entre la unidad de memoria secundaria y la memoria principal como se muestra en la Fig. IV.1.

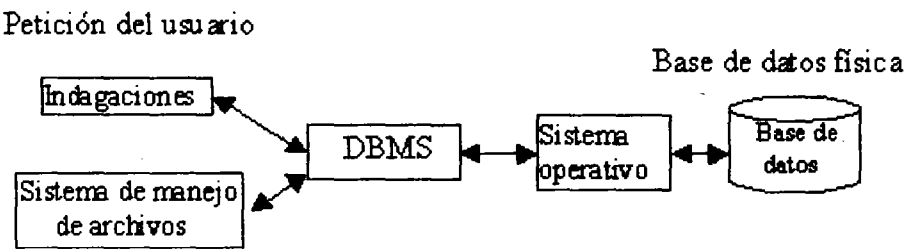

Fig. IV.1 El DBMS como interfase entre la Base de Datos física y las peticiones del usuario

En sí, un sistema manejador de Base de Datos es el corazón de la Base de Datos ya que se encarga del control total de los posibles aspectos que la puedan afectar.

IV.3 Consideraciones para la elección de sistemas de gestión de Bases de Datos

Las consideraciones que se deben tener en cuenta para la elección del sistema de gestión de Base de Datos son:

- Modelos lógicos
- Número de usuarios
- Distribución
- Costos
- Propósito

Estas consideraciones son de suma importancia dado que un error en la elección del sistema gestionador tendría una tremenda repercusión en el resultado final del proyecto de Base de Datos; desde su funcionamiento al arranque como al contemplar en una posible implementación de nuevos módulos los cuales añadan un extra a la aplicación que ofrezca una mayor vida útil del producto.

Modelos lógicos

Para representar el mundo real a través de esquemas conceptuales se han creado una serie de modelos. El criterio principal que se utiliza para clasificar los SGBD es el modelo lógico en que se basan. Los modelos lógicos empleados con mayor frecuencia en los SGBD comerciales actuales son el relacional, el de red y el jerárquico. Algunos SGBD más modernos se basan en modelos orientados a objetos.

a) **Modelo relacional:** Se basa en el concepto matemático denominado "relación", que gráficamente se puede representar como una tabla. En el modelo relacional, los datos y las relaciones existentes entre los datos se representan mediante dichas relaciones matemáticas, cada una con un nombre que es único y con un conjunto de columnas.

En el modelo relacional la Base de Datos es percibida por el usuario como un conjunto de tablas. Esta percepción es sólo a nivel lógico (en los niveles externo y conceptual de la arquitectura de tres niveles), ya que a nivel físico puede estar implementada mediante distintas estructuras de almacenamiento como se muestra en la Fig. IV.2.

Num_empleado	Nombre	Sección
33	Juan	25
34	Jorge	26

Num_sección	Nombre
25	Textil
26	Pintura

Fig. IV.2. Representa al mundo real mediante tablas relacionadas entre sí por columnas comunes.

Este modelo se emplea con más frecuencia en la práctica, debido a las ventajas que ofrece sobre otros modelos, entre ellas, el rápido

entendimiento por parte de usuarios que no tienen conocimientos profundos sobre Sistemas de Bases de Datos.

b) **Modelo de red:** Los datos se representan como colecciones de registros y las relaciones entre los datos se representan mediante conjuntos, que son punteros en la implementación física. Los registros se organizan como se muestra en la Fig. IV.3.

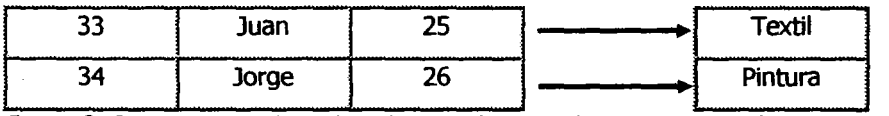

Fig. IV.3. Representamos al mundo real como colecciones de registros y sus relaciones se representan por medio de ligas o enlaces, los cuales pueden verse como punteros.

Este modelo permite la representación de muchos a muchos, de tal forma que cualquier registro dentro de la base de datos puede tener varias ocurrencias superiores a él. El modelo de red evita redundancia en la información, a través de la incorporación de un tipo de registro denominado el conector.

c) **Modelo jerárquico:** Es un tipo de modelo de red con algunas restricciones. De nuevo los datos se representan como colecciones de registros y las relaciones entre los datos se representan mediante conjuntos. Sin embargo, en el modelo jerárquico cada nodo puede tener un solo padre. Una Base de Datos jerárquica puede representarse mediante un árbol: los registros son los nodos, también denominados segmentos, y los arcos son los conjuntos.

La mayoría de los SGBD comerciales actuales están basados en el modelo relacional, mientras que los sistemas más antiguos estaban basados en el modelo de red o el modelo jerárquico. Estos dos últimos modelos requieren

que el usuario tenga conocimiento de la estructura física de la Base de Datos a la que se accede, mientras que el modelo relacional proporciona una mayor independencia de datos. Se dice que el modelo relacional es declarativo (se especifica qué datos se han de obtener) y los modelos de red y jerárquico son navegacionales (se especifica cómo se deben obtener los datos). En la Fig. IV.4 podemos ver un ejemplo de este tipo de modelo.

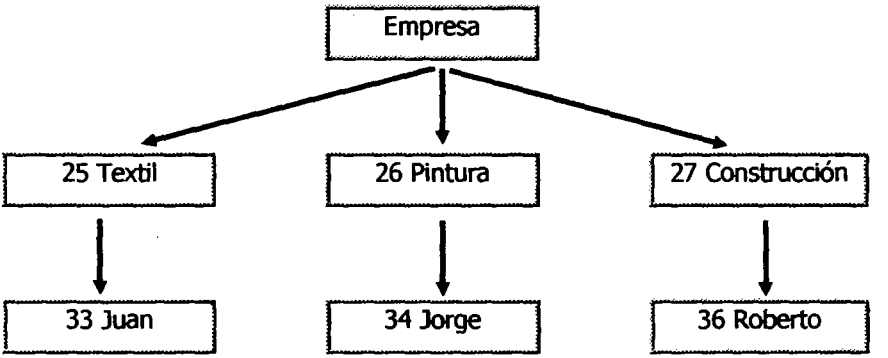

Fig. IV. 4. Tiene forma de árbol invertido. Un padre puede tener varios hijos pero cada hijo sólo puede tener un padre

Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos.

d) Modelo orientado a objetos: Define una Base de Datos en términos de objetos, sus propiedades y sus operaciones. Los objetos con la misma estructura y comportamiento pertenecen a una clase, y las clases se organizan en jerarquías o grafos acíclicos. Las operaciones de cada clase se especifican en términos de procedimientos predefinidos denominados métodos. Algunos SGBD relacionales existentes en el mercado han estado extendiendo sus modelos para incorporar conceptos orientados a objetos. A estos SGBD se les conoce como sistemas objeto-relacionales.

Por ejemplo:

Num_empleado-Nombre-Sección agregándole la entidad Num_gerente; donde los atributos considerados para cada uno son:

Num_empleado: Nombre, Dirección, Teléfono, RFC, Nivel
Num_gerente: Nombre, Dirección, Teléfono, RFC, CURP
Num_sección: Nombre

Los atributos de Nombre, Dirección, Teléfono y RFC se repiten en la entidad Num_empleado y Num_gerente, así que podemos agrupar estos elementos para formar la clase Persona con dichos campos. Quedando por separado en Num_empleado: Nivel. Y en Num_gerente: CURP; la Num_sección no entra en la agrupación (Clase persona) ya que la clase específica los datos de sólo personas, así que queda como clase sección.

Número de usuarios

Un segundo criterio para clasificar los SGBD es el número de usuarios a los que da servicio el sistema. Los sistemas monousuario sólo atienden a un usuario a la vez, y su principal uso se da en las computadoras personales. Los sistemas multiusuario, entre los que se encuentran la mayor parte de los SGBD, atienden a varios usuarios al mismo tiempo.

Distribución

Un tercer criterio es el número de sitios en los que está distribuida la Base de Datos. Casi todos los SGBD son centralizados: sus datos se almacenan en una sola computadora. Los SGBD centralizados pueden atender a varios usuarios, pero el SGBD y la Base de Datos en sí residen por completo en una sola máquina. En

los SGBD distribuidos la Base de Datos real y el propio software del SGBD pueden estar distribuidos en varios sitios conectados por una red. Los SGBD distribuidos homogéneos utilizan el mismo SGBD en múltiples sitios. Una tendencia reciente consiste en crear software para tener acceso a varias bases de datos autónomas preexistentes almacenadas en SGBD distribuidos heterogéneos. Esto da lugar a los SGBD federados o sistemas multiBase de Datos en los que los SGBD participantes tienen cierto grado de autonomía local. Muchos SGBD distribuidos emplean una arquitectura cliente-servidor.

Costo

Un cuarto criterio es el costo del SGBD. La mayor parte de los paquetes de SGBD cuestan entre $ 4,000.00 y $ 500,000.00 pesos. Todo esto depende en gran medida a las características que le preceden como las herramientas con las que cuenta, las plataformas en que puede ser utilizada o su capacidad de respuesta entre otras. Un resumen de los SGBD más populares se encuentra en la Tabla IV.1.

Producto	Resumen
SQL Server (Standar Query Lenguaje) Microsoft	Constituye un lanzamiento determinante para los productos de bases de datos de Microsoft, continuando con la base sólida establecida por SQL Server 6.5. Como la mejor base de datos para Windows NT, SQL Server es el RDBMS de elección para una amplia gama de clientes corporativos y Proveedores Independientes de Software (ISVs) que construyen aplicaciones de negocios. Las necesidades y requerimientos de los clientes han llevado a la creación de innovaciones de producto significativas para facilitar la utilización, escalabilidad, confiabilidad y almacenamiento de datos.

Oracle Corporación Oracle	Es manejador de base de datos relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información. Es el conjunto de datos que proporciona la capacidad de almacenar y acude a éstos de forma consecuente con un modelo definido como relacional. Además es una suite de productos que ofrece una gran variedad de herramientas.
Informix Dynamic Server (IDS) 9.30 IBM	Proporciona fiabilidad superior, atendiendo las necesidades de las exigentes prácticas actuales del e-business-particularmente para aplicativos que requieran transacciones de alto desempeño. Soporta requisitos de procesamiento de transacción online, complejos y rigurosos. Optimiza capacidades de inteligencia del negocio competitivas. Maximiza operaciones de datos para el grupo de trabajo y para la empresa en total. Proporciona la firmeza de una administración de base de datos comprobada, mejor de su especie.

Tabla IV. 1. Principales SGBD del mercado

Este punto siempre será de gran consideración para la empresa ya que cualquiera de dichos paquetes siempre implica un gasto que se acrecenta con el número de usuarios con que cuenta la empresa y que no se habrá de perder de vista.

Propósito

Por último, los SGBD pueden ser de propósito general o de propósito específico. Cuando el rendimiento es fundamental, se puede diseñar y construir un SGBD de propósito especial para una aplicación específica, y este sistema no sirve para otras aplicaciones. Muchos sistemas de reservas de líneas aéreas son SGBD de propósito especial y pertenecen a la categoría de sistemas de procesamiento de transacciones en línea (OLTP), que deben atender un gran número de transacciones concurrentes sin imponer excesivos retrasos.

Todo lo anterior nos enseña el papel tan importante que tienen los SGBD en el funcionamiento general de una aplicación de Base de Datos, además de auxiliarnos a tomar una decisión de que SGBD es el que más nos conviene según se ajuste a nuestros fines todo esto gracias al conocimiento de los elementos con que cuenta y el gasto que representa.

Capítulo V

Interfaces (IU)

Este capítulo estudia las características con que debe contar una buena interfaz para un sistema de Base de Datos.

Una Interfaz de Usuario a diferencia de las otras partes que conforman una Base de Datos, como el análisis o la gestión, no suele ser un elemento crucial en el desarrollo de un sistema, sin embargo este elemento es capaz por si solo de impulsar un sistema mediano al éxito, como de llevar una aplicación robusta y repleta de útiles herramientas al fracaso y esto es debido en principio al impacto estético que sirve de carta de presentación a la empresa, en segundo término a la facilidad de uso que representa un sistema bien ordenado y en tercero el ahorro que implica en capacitación una aplicación sencilla y de uso natural.

V.1. ¿Qué es una interfaz de usuario?

La Interfaz de Usuario (IU), de un programa es un conjunto de elementos hardware y software de una computadora que presentan información al usuario y le permiten interactuar entre la información y la computadora. También se puede considerar parte de la IU la documentación (manuales, ayuda, referencia, tutoriales) que acompaña al hardware y al software.

Si la IU está bien diseñada, el usuario encontrará la respuesta que espera a su acción. Si no es así puede ser frustrante su operación, ya que el usuario habitualmente tiende a culparse a sí mismo por no saber usar el objeto.

Los programas son usados por usuarios con distintos niveles de conocimientos, desde principiantes hasta expertos. Es por ello que no existe una interfaz válida para todos los usuarios y todas las tareas. Debe permitirse libertad al usuario para que elija el modo de interacción que más se adecué a sus objetivos en cada momento. La mayoría de los programas y sistemas operativos ofrecen varias formas de interacción al usuario.

V.2. ¿Qué es un prototipo?

Está aceptado generalmente que un prototipo es un modelo a escala o facsímile de lo real, pero no tan funcional para que equivalga al producto final. Los prototipos se presentan en muchas formas y tamaños. En la industria automotriz, los prototipos de autos pueden ir en complejidad desde un modelo tallado en madera hasta un vehículo motorizado real que se puede conducir.

El mismo rango de complejidad resulta cierto para los sistemas de software. El prototipo puede ser tan simple como un conjunto de disposiciones de pantallas en hojas de papel pegadas en la pared, o tan sofisticado como programas de software animado que permiten que los usuarios hagan clic en los botones e introduzcan datos.

Un prototipo puede ser útil en diferentes fases del proyecto. El objetivo del prototipo debe ser claro en cada fase. Durante la fase de análisis se puede usar un prototipo para obtener los requerimientos del usuario. En la fase de diseño un prototipo puede ayudar a evaluar muchos aspectos de la implementación seleccionada.

V.2.1. Beneficios de la creación temprana de prototipos

Más que cualquier tipo de modelo, el prototipo de interfaz realmente introduce a los usuarios en el proyecto. Por primera vez el sistema tiene una "cara". Inevitablemente, después de ver el prototipo, alguien le dirá al analista acerca de un evento que hasta ese momento no se había mencionado. También añadirá conceptos de datos a la ventana que no se mencionaron durante las entrevistas y posiblemente elimine algunos por irrelevantes. Llevado un poco más

allá, el prototipo puede usarse como una herramienta de entrevista que nos habla mucho de la satisfacción del usuario.

La creación de prototipos también causará que salgan a la superficie algunos asuntos en los procesos. Cuando un sistema comienza a salir de lo que antes era del dominio único de la computadora, tal vez encuentre personas que realizan las mismas tareas en formas radicalmente diferentes. El nuevo sistema puede eliminar la captura de datos redundantes en hojas de cálculo, en sistemas auxiliares, alterar o prescindir del trabajo de alguien. El prototipo es frecuentemente la primera vista clara del ambiente de trabajo para usuario futuro.

La creación de prototipos también hará evidentes asuntos técnicos en un punto del proyecto cuando todavía hay tiempo suficiente para hacer algo al respecto. Las disposiciones de ventanas pueden ser analizadas para obtener un sentido de qué tan complejo puede ser el acceso a los datos.

El prototipo también permite que se exploren ambientes de destino posibles. Tal vez una interfaz gráfica de usuario no sea lo óptimo para su aplicación particular. Se puede explorar la manera en que se vería la interfaz con sistemas basados en pluma, dispositivos portátiles pequeños, lectores de código de barras, una página Web o las antiguas pantallas de caracteres.

Es importante que los usuarios estén lo suficientemente familiarizados con la tecnología de destino para evaluar las disposiciones sin confundirse.

Otro punto importante es que con la creación de estas interfaces prototipo se podrá dar a nuevos usuarios una introducción y una demostración de la aplicación sin riesgo a un posible mal uso de ésta.

Sin duda el diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema y la calidad de la Interfaz de Usuario puede ser uno de los motivos que conduzcan al éxito o al fracaso de un sistema. Los principios que se presentan en este capítulo son de gran provecho para creación de interfaces funcionales y de fácil operación.

V.3. Modelos de Interfaz de Usuario

Como todo sistema éste se compone de diversas pantallas que pueden ser entendidas, accedidas y modificadas únicamente por algún tipo de usuario con permisos o privilegios especiales. De esta forma es como podemos distinguir distintos modelos de interfaz:

- Modelo de usuario
- Modelo de diseñador
- Modelo de programador

Modelo del usuario

El usuario tiene su visión personal del sistema, y espera que éste se comporte de una cierta forma. Se puede conocer el modelo del usuario estudiándolo, ya sea realizando tests de usabilidad, entrevistas, o a través de una realimentación. Una interfaz debe facilitar el proceso de crear un modelo mental efectivo.

Para ello son de gran utilidad las metáforas, que asocian un dominio nuevo a uno ya conocido por el usuario. Un ejemplo típico es describir al área de trabajo como "escritorio" que es lo más común en la mayoría de las interfaces gráficas actuales.

Modelo del diseñador

El diseñador mezcla las necesidades, ideas, deseos del usuario y los materiales de que dispone el programador para diseñar un producto de software. Es un intermediario entre ambos.

El modelo del diseñador describe los objetos que utiliza el usuario, su presentación al mismo y las técnicas de interacción para su manipulación. Consta de tres partes: presentación, interacción y relaciones entre los objetos que gráficamente se muestra Fig. V.1.

- La presentación es lo que primero capta la atención del usuario, pero más tarde pasa a un segundo plano, y adquiere más importancia la interacción con el producto para poder satisfacer sus expectativas. La presentación no es lo más relevante y un abuso en la misma (por ejemplo, en el color) puede ser contraproducente, distrayendo al usuario.

- La segunda parte del modelo define las técnicas de interacción del usuario, a través de diversos dispositivos.

- La tercera es la más importante, y es donde el diseñador determina la simbología adecuada que encaja con el modelo mental del usuario. El modelo debe comenzar por esta parte e ir hacia arriba. Una vez definida la simbología y los objetos del interfaz, los aspectos visuales saldrán de una manera lógica y fácil.

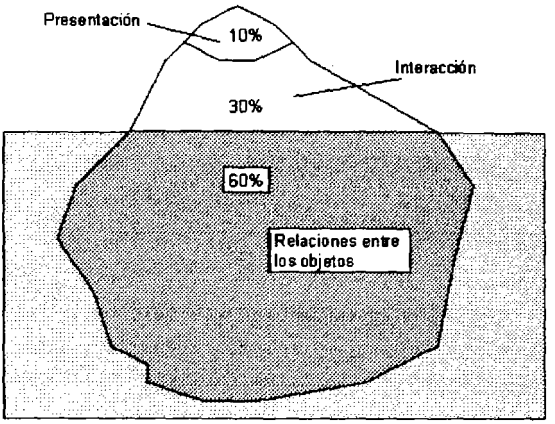

Fig V.1. Representación del modelo del diseñador: el look-and-feel iceberg, de IBM (1992).

Estos modelos deben estar claros para los participantes en el desarrollo de un producto, de forma que se consiga una interfaz atractiva y a la vez efectiva para el trabajo con el programa.

Una interfaz no es simplemente una cara bonita; esto puede impresionar a primera vista pero decepcionar a la larga. Lo importante es que el programa se adapte bien al modelo del usuario, cosa que se puede comprobar utilizando el programa más allá de la primera impresión.

Modelo del programador

Es el más fácil de visualizar, al poderse especificar formalmente. Está constituido por los objetos que manipula el programador, distintos de los que trata el usuario (ejemplo: el programador llama Base de Datos a lo que el usuario podría llamar agenda). Estos objetos deben esconderse del usuario.

Los conocimientos del programador incluyen la plataforma de desarrollo, el sistema operativo, las herramientas de desarrollo y especificaciones. Sin embargo,

esto no significa necesariamente que tenga la habilidad de proporcionar al usuario los modelos y metáforas más adecuadas. Muchos no consideran el modelo del usuario del programa, y sí sus propias expectativas acerca de cómo trabajar con la computadora.

V.4. Consideraciones para el diseño de Interfaces de Usuario

Muchos de los principios que están tras el buen diseño no son necesariamente nuevos. La mayoría de los usuarios preferiría usar sus antiguos sistemas a tener que enfrentarse al uso de un nuevo sistema con una interfaz pobremente diseñada. Lo que hace que un diseño sea mejor que otro son principios generales. Entre los principios relevantes para el diseño e implementación de IU, ya sea para las IU gráficas, como para la Web, tenemos:

Anticipación

Las aplicaciones deberían intentar anticiparse a las necesidades del usuario y no esperar a que el usuario tenga que buscar la información, recopilarla o invocar las herramientas que va a utilizar.

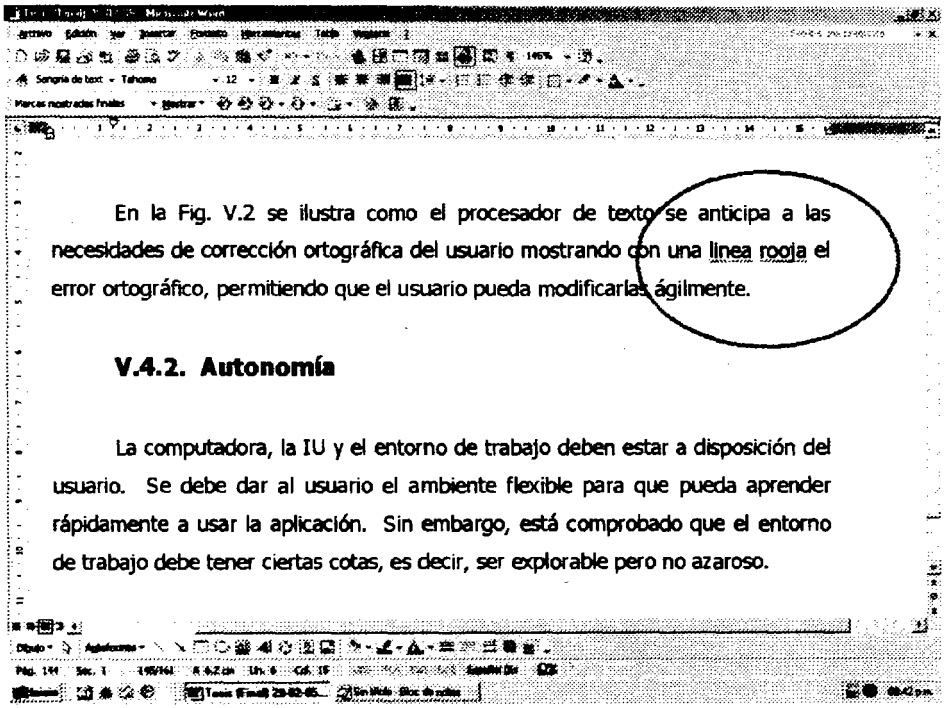

Fig. V.2. Ejemplo de características anticipadas

En la Fig. V.2 se ilustra como el procesador de texto se anticipa a las necesidades de corrección ortográfica del usuario mostrando con una línea roja el error ortográfico, permitiendo que el usuario pueda modificarlas ágilmente.

Autonomía

La computadora, la IU y el entorno de trabajo deben estar a disposición del usuario. Se debe dar al usuario el ambiente flexible para que pueda aprender rápidamente a usar la aplicación. Sin embargo, está comprobado que el entorno de trabajo debe tener ciertas cotas, es decir, ser explorable pero no azaroso.

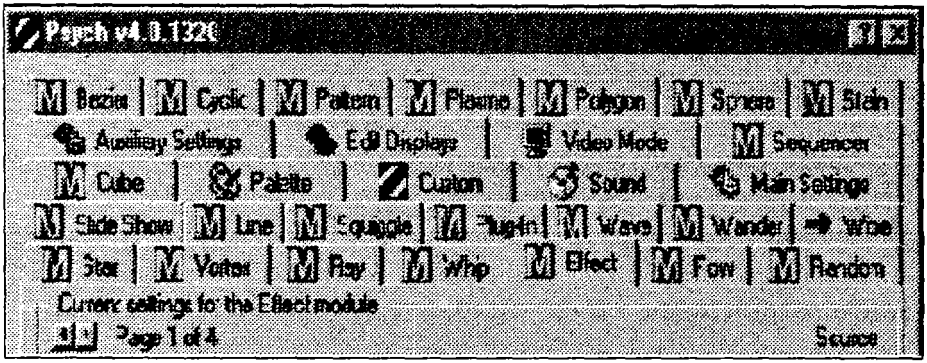

Fig. V.3. Ejemplo de ambiente complejo

En la Fig. V.3 se visualiza un diseño incorrecto de Interfaz de Usuario. La cantidad de opciones propuestas propone un grado de complejidad que no permite que el usuario pueda aprender a utilizar el sistema en forma progresiva.

Es importante utilizar mecanismos indicadores de estado del sistema que mantengan a los usuarios alertas e informados. No puede existir autonomía en ausencia de control, y el control no puede ser ejercido sin información suficiente. Además, se debe mantener información del estado del sistema en ubicaciones fáciles de visualizar.

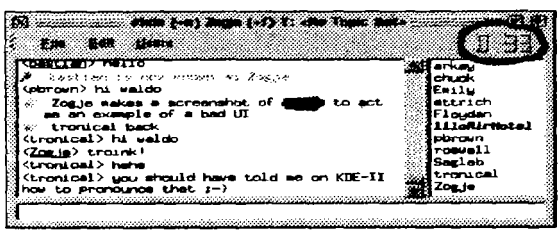

Fig. V.4. Ejemplo de información de estado inadecuada

En la Fig. V.4 se ejemplifica una incorrecta disposición de componentes en la IU. El reloj no debe ser incorporado en el menú del sistema ya que aporta

confusión al usuario. Para mantenerlo informado sería mas adecuado colocarlo en la barra de estado del sistema.

Percepción del color

Aunque se utilicen convenciones de color en la IU, se deberían usar otros mecanismos secundarios para proveer la información a aquellos usuarios con problemas en la visualización de colores

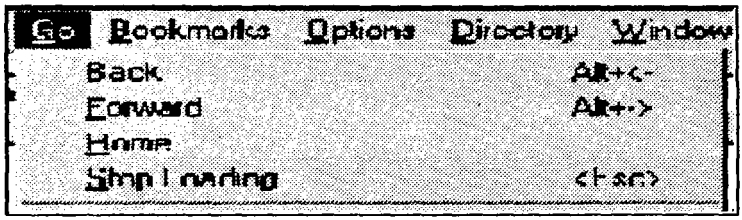

Fig. V.5. Ejemplo de color e inconsistencia

En la Fig. V.5 se representa un mecanismo secundario muy utilizado para ejecución de comandos: los comandos abreviados (shortcut-keys). Sin embargo la aplicación presenta un problema de inconsistencia ya que define combinaciones de teclas que difieren a lo esperado por el usuario, por ejemplo Alt+< en lugar de Alt+B.

Consistencia

Para lograr una mayor consistencia en la IU se requiere profundizar en diferentes aspectos que están catalogados en niveles. Se realiza un ordenamiento de mayor a menor consistencia:

- **Interpretación del comportamiento del usuario:** la IU debe comprender el significado que le atribuye un usuario a cada requerimiento. Ejemplo: mantener el significado de las los comandos abreviados (shortcut-keys) definidos por el usuario.
- **Estructuras invisibles:** se requiere una definición clara de las mismas, ya que sino el usuario nunca podría llegar a descubrir su uso. Ejemplo: la ampliación de ventanas mediante la extensión de sus bordes.
- **Pequeñas estructuras visibles:** se puede establecer un conjunto de objetos visibles capaces de ser controlados por el usuario, que permitan ahorrar tiempo en la ejecución de tareas específicas. Ejemplo: ícono y/o botón para impresión.
- **Una sola aplicación o servicio:** la IU permite visualizar a la aplicación o servicio utilizado como un componente único. Ejemplo: La IU despliega un único menú, pudiendo además acceder al mismo mediante comandos abreviados.
- **Un conjunto de aplicaciones o servicios:** la IU visualiza a la aplicación o servicio utilizado como un conjunto de componentes. Ejemplo: La IU se presenta como un conjunto de barras de comandos desplegadas en diferentes lugares de la pantalla, pudiendo ser desactivadas en forma independiente.
- **Consistencia del ambiente:** la IU se mantiene en concordancia con el ambiente de trabajo. Ejemplo: La IU utiliza objetos de control como menúes, botones de comandos de manera análoga a otras IU que se usen en el ambiente de trabajo.
- **Consistencia de la plataforma:** La IU es concordante con la plataforma. Ejemplo: La IU tiene un esquema basado en ventanas, el cual es acorde al manejo del sistema operativo Windows.

En la Fig. V.6 puede observarse la mejora en la consistencia de las pequeñas estructuras visibles para los sistemas gráficos basados en ventanas. La

inclusión de la opción ☒ para cerrar la ventana –operación comúnmente utilizada en estas aplicaciones- simplifica la operatividad del mismo.

Fig. V.6. Ejemplo de consistencia

La inconsistencia en el comportamiento de componentes de la IU debe ser fácil de visualizar. Se debe evitar la similitud en los componentes para aplicaciones distintas de la IU. Los objetos deben ser consistentes con su comportamiento; si dos objetos actúan en forma diferente, deben lucir diferentes. La única forma de verificar si la IU satisface las expectativas del usuario es mediante pruebas.

Eficiencia del usuario

Se debe considerar la productividad del usuario antes que la productividad de la máquina. Si el usuario debe esperar la respuesta del sistema por un período prolongado, estas pérdidas de tiempo se pueden convertir en pérdidas económicas para la organización. Los mensajes de ayuda deben ser sencillos y proveer respuestas a los problemas. Los menús y etiquetas de botones deberían tener las palabras claves del proceso.

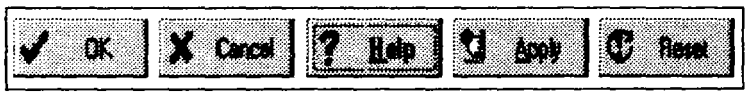

Fig. V.7. Ejemplo de definición incorrecta de botones de acción

En la Fig. V.7 se demuestra como una incorrecta definición de las palabras clave de las etiquetas de los botones de comando puede confundir al usuario. Los

botones **OK** y **Apply** aparentan realizar el mismo proceso. Esto puede solucionarse suprimiendo uno de ellos si realizan la misma tarea o etiquetándolos con los nombres de los procesos específicos que ejecutan.

Ley de Fitt

El tiempo para alcanzar un objetivo es una función de la distancia y tamaño del objetivo. Es por ello, que es conveniente usar objetos grandes para las funciones importantes.

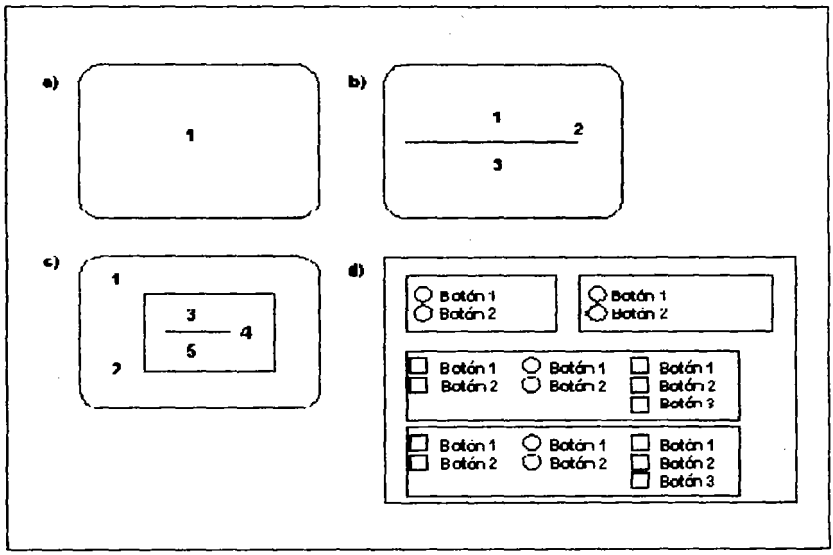

Fig. V.8. Ejemplo de percepción visual

En la Fig. V.8 se puede apreciar la relación entre los elementos de diseño de pantalla y su percepción visual. El número de elementos visuales que perciben son en el caso de:

a) 1 (el fondo)

b) 3 (la línea, lo que está encima y lo que está debajo)

c) 5 (el espacio fuera del recuadro, el recuadro, la línea y el espacio encima y debajo de ésta)

d) el número se eleva a 35, siguiendo el mismo criterio.

En conclusión: cada elemento nuevo que se añade influye más de lo que se piensa en el usuario.

Interfaces explorables

Siempre que sea posible se debe permitir que el usuario pueda salir ágilmente de la IU, dejando una marca del estado de avance de su trabajo, para que pueda continuarlo en otra oportunidad.

Para aquellos usuarios que sean noveles en el uso de la aplicación, se deberá proveer de guías para realizar tareas que no sean habituales.

Es conveniente que el usuario pueda incorporar elementos visuales estables que permitan, no solamente un desplazamiento rápido a ciertos puntos del trabajo que esté realizando, sino también un sentido de "casa" o punto de partida.

La IU debe poder realizar la inversa de cualquier acción que pueda llegar a ser de riesgo, de esta forma se apoya al usuario a explorar el sistema sin temores.

Siempre se debe contar con un comando "Deshacer". Este suprimirá la necesidad de tener que contar con diálogos de confirmación para cada acción que realice en sistema.

El usuario debe sentirse seguro de poder salir del sistema cuando lo desee. Es por ello que la IU debe tener un objeto fácil de accionar con el cual poder finalizar la aplicación.

Objetos de interfaz humana

Los objetos de interfaz humana no son necesariamente los objetos que se encuentran en los sistemas orientados a objetos. Estos pueden ser vistos, escuchados, tocados o percibidos de alguna forma. Además, estos objetos deberían ser entendibles, consistentes y estables.

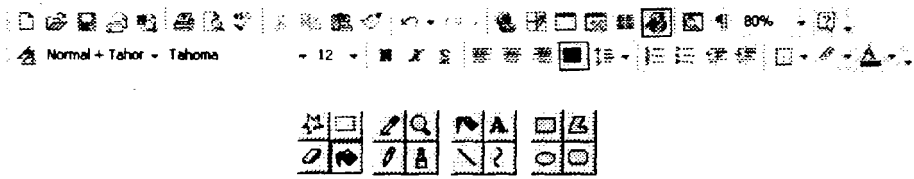

Fig. V.9. Ejemplo de barras de controles

En la Fig. V.9 se presentan barras de controles que simplifican la operación de un sistema. A través de las ilustraciones que poseen los mismos, el usuario puede aprender fácilmente su uso además de ser elementos de uso frecuente para distintos sistemas lo que hace que la comprensión del funcionamiento de los mismos se haga más sencilla.

Uso de metáforas

Las buenas metáforas crean figuras mentales fáciles de recordar. La IU puede contener objetos asociados al modelo conceptual en forma visual, con sonido u otra característica perceptible por el usuario que ayude a simplificar el uso del sistema.

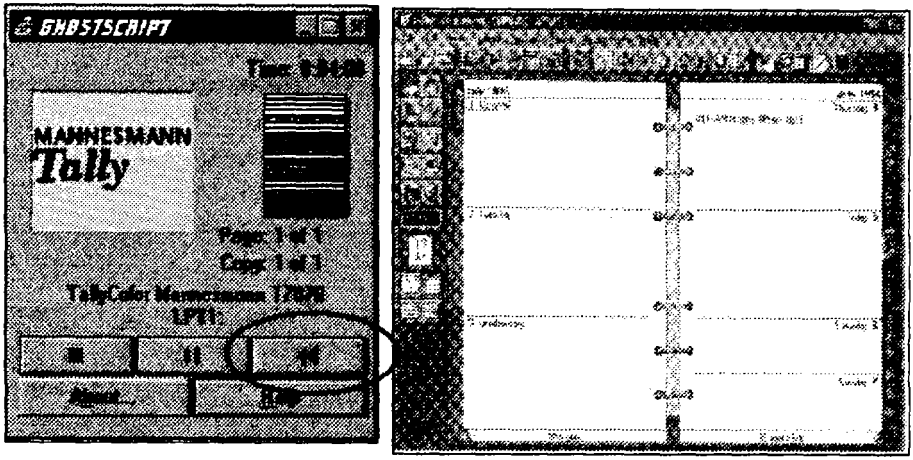

Fig. V.10. Ejemplo de metáforas

En la Fig. V.10 se compara la aplicación de metáforas en el desarrollo de una IU. En el primer caso, se utiliza incorrectamente la metáfora de una cámara de video para representar el procesamiento de un documento por una impresora. Se puede observar que el botón [<<] carece de sentido, ya que no se puede volver atrás un trabajo que ya ha sido impreso. En el segundo caso, la metáfora una agenda es utilizada correctamente para la implementación de una agenda electrónica.

Curva de aprendizaje

Se refiere a que tan rápido el usuario va a aprender a usar un sistema con el cual no había tenido contacto previamente. El ideal es que la curva de aprendizaje sea nula, y que el usuario principiante pueda alcanzar el dominio total de la aplicación sin esfuerzo. La facilidad de aprendizaje podría medirse en base al número de tareas completadas en cierto periodo, número de errores cometidos, o respecto al número de veces que utilizó la opción de ayuda.

Reducción de inactividad

Siempre que sea posible, el uso de "multi-threading" o también conocido como multiprocesos es conveniente el aprovechamiento de estos al colocar procesos en segundo plano o "background". Las técnicas de trabajo multitarea posibilitan el trabajo ininterrumpido del usuario, estas generalmente se concentran en la transmisión y computación de datos en segundo plano.

Protección del trabajo

Se debe poder asegurar que el usuario nunca pierda su trabajo, ya sea por error de su parte, problemas de transmisión de datos, de energía, o alguna otra razón inevitable.

Auditoria del sistema

La mayoría de los sistemas, no mantienen información acerca de la situación del usuario en el entorno, pero para cualquier aplicación es conveniente conocer un conjunto de características tales como: hora de acceso al sistema, ubicación del usuario en el sistema y lugares a los que ha accedido, entre otros. Todas estas aplicaciones se agrupan en un área conocida como seguridad. Además, el usuario debería poder salir del sistema y al volver a ingresar continuar trabajando en lugar dónde había dejado.

Legibilidad

Para que la IU favorezca la usabilidad del sistema de software, la información que se exhiba en ella debe ser fácil de ubicar y leer. Para lograr

obtener este resultado se deben tener en cuenta alguna como: el texto que aparezca en la IU debería tener un alto contraste, se debe utilizar combinaciones de colores como el texto en negro sobre fondo blanco o amarillo suave. El tamaño de las fuentes tiene que ser lo suficientemente grande como para poder ser leído en monitores estándar. Es importante hacer clara la presentación visual (colocación/agrupación de objetos, evitar la presentación de excesiva información.

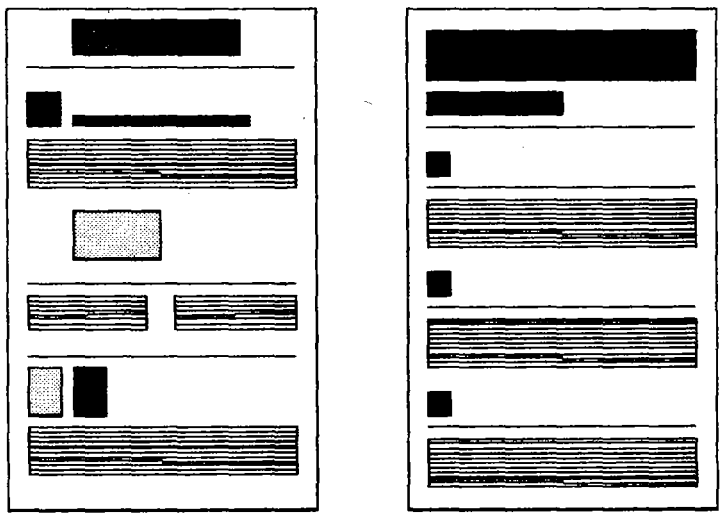

Fig. V.11. Ejemplo de legibilidad

En la Fig. V.11 se describe una comparación de disposición de los objetos en pantalla. La figura de la izquierda, combina una disposición asimétrica de la información con un conjunto de colores que no facilita la lectura. La figura de la derecha realiza la presentación de la información utilizando una gama de colores homogénea y una alineación del texto que favorece a la legibilidad del mismo.

Interfaces visibles

Esto significa que el usuario siempre concentre su atención en una ventana específica, y que el resto de las ventanas queden en segundo plano para su labor. La navegación dentro de la aplicación debe ser reducida a la mínima expresión. El usuario debe sentir que se mantiene en un único lugar y que el que va variando es su trabajo. Esto no solamente elimina la necesidad de mantener múltiples herramientas de exploración como lo pueden ser densas explicaciones de donde se encuentra situado o como llegar nuevamente al inicio, sino que además brindan al usuario una sensación de autonomía.

V.5. Funcionalidad

Cosntantine y Lockwood en 1999 al cuantificar la complejidad con que se aprende o memoriza una interfaz logran sintetizar los elementos clave de efectividad, eficiencia y satisfacción necesarias para cumplir con un diseño funcional de una interfaz como se muestra en la Fig. V.12.

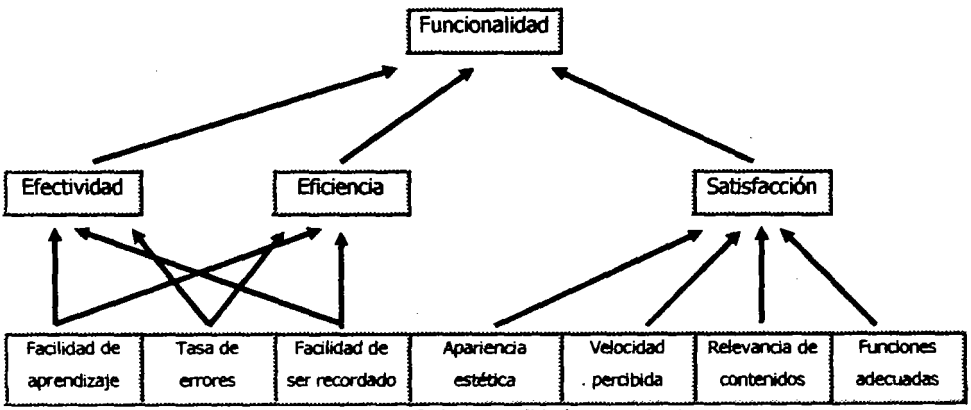

Fig. V.12. Funcionalidad y sus criterios

Una IU se puede considera funcional si se cumplen con dos puntos importantes que son: efectividad y eficiencia en un ambiente técnico y satisfacción desde el punto de vista del usuario.

El texto que comprende éste capítulo describe la importancia que tiene una interfaz de usuario consistente para el desarrollo de un sistema de Base de Datos y muestra con ejemplos claros las consideraciones a tomar en cuenta para su concepción, así como los criterios de efectividad, eficiencia y satisfacción que diseñadores como usuarios buscan en todo sistema.

Conclusiones

Este trabajo a través de sus cinco capítulos nos deja en claro que construir un sistema y en nuestro caso particular una Base de Datos no es únicamente horas de programación siguiendo la tendencia del momento sino que hay numerosas consideraciones a tomar en cuenta mucho antes de encender la computadora como lo son el conocimiento en los procedimientos, la organización, cargos, responsabilidades, localización de personas clave, infraestructura, solvencia económica y posibilidades de expansión de la empresa para la cual se desarrolle un sistema, así como la importante opinión de los usuarios finales y la elaboración prototipos que nos apoyen a visualizar un resultado terminal, además de poner énfasis en el hecho de generar una clara documentación que nos permita hacer modificaciones al sistema sin la laboriosa necesidad de re-examinar las muchas líneas de código que se generan.

A lo largo del trabajo se definen constantemente los conceptos esenciales y básicos con que debe contar cualquier persona que decida inmiscuirse en la labor de desarrollo de una Base de Datos sin importar su nivel académico; de igual forma se detallan las metodologías que se han utilizado para el desarrollo de una Base de Datos y se estudia paso a paso la forma actual con que se trabaja en el desarrollo de sistemas computacionales y paralelamente se hacen notar importantes consideraciones a tomar en cuenta desde aspectos de interacción con la empresa hasta asuntos técnicos que afectan directamente al diseño del sistema.

Finalmente la síntesis de todo esto da rumbo a importantes decisiones tanto técnicas como lo son la elección de un lenguaje adecuado para la aplicación, así como del sistema que gestiona el almacenamiento y consulta de los datos, todo esto repleto de ejemplos que revelan las fallas más comunes y fundamentales en el desarrollo de un sistema e Base de Datos óptimo en técnica, funcionalidad, costo y tiempo.

Glosario

A

Acceso directo: Es un ícono que permite abrir más fácilmente un determinado programa o archivo.

Algoritmo: Conjunto de reglas bien definidas para la resolución de un problema. Un programa de software es la transcripción, en lenguaje de programación, de un algoritmo.

ANSI: American National Standards Institute (Instituto de Estándares para las Naciones de América). Organización que promueve el desarrollo de estándares en los Estados Unidos. Es miembro de la ISO (International Organization for Standardization).

Árbol (tree): Estructura de datos en la cual los registros son almacenados de manera jerárquica.

ASCII: American Standard Code of Information Interchange (Código de Información de Intercambio de Estándares Americanos): Código normalizado estadounidense para el intercambio de la información. Código que permite definir caracteres alfanuméricos; se lo usa para lograr compatibilidad entre diversos procesadores de texto. Se pronuncia "aski".

B

Back-end processor: procesador que se utiliza para determinada función muy especializada, como por ejemplo, administrar una base de datos.

Backup: Copia de seguridad. Se hace para prevenir una posible pérdida de información.

Barra de herramientas: Conjunto de íconos que conducen a instrucciones.

Base de datos: Conjunto de datos organizados de modo tal que resulte fácil acceder a ellos, gestionarlos y actualizarlos.

BASIC: Beginner's All-Purpose Symbolic Instruction Code: Código de Instrucción Simbólica Multipropósito para Principiantes. Lenguaje de programación, creado en 1963, sencillo y muy difundido.

BIOS: Basic Input/Output System (Sistema Básico de Entrada y Salida de Datos): Sistema básico de ingreso/salida de datos. Conjunto de procedimientos que controla el flujo de datos entre el sistema operativo y dispositivos tales como el disco rígido, la placa de video, el teclado, el mouse y la impresora.

BIT: abreviatura de binary digit (dígito binario). El bit es la unidad más pequeña de

almacenamiento en un sistema binario dentro de una computadora.

boot (butear): Cargar el sistema operativo de una computadora.

Bps: Bits por segundo.

Bug: Bicho, insecto. Error de programación que genera problemas en las operaciones de una computadora.

Bus: Enlace común; conductor común; vía de interconexión. Método de interconexión de dispositivos mediante una sola línea compartida.

Bus serial: Método de transmisión de un bit por vez sobre una sola línea.

Byte: Contracción de binary digit. Tiene dos acepciones: 1) se refiere a la capacidad de un dispositivo físico de poder estar en dos estados perfectamente diferenciados. Cada uno de estos estados puede representarse por un dígito binario: "0" para uno de ellos y "1" para el otro. 2) es el aumento de la cantidad de información que se obtiene cuando ocurre un suceso entre dos posibles distintos. Es la unidad fundamental en Teoría de la Información. Cada byte está compuesto por ocho bits.

C

Cable-módem: módem que conecta una computadora con Internet a alta velocidad, por medio de una línea de TV por cable.

Caché de disco: Pequeña porción de memoria RAM que almacena datos recientemente leídos, con lo cual agiliza el acceso futuro a los mismos datos.

Caché: En un navegador, el caché guarda copias de documentos de acceso frecuente, para que en el futuro aparezcan más rápidamente.

CAD: Computer Aided Design: Diseño Asistido por Computadora. Software que permite crear dibujos de precisión, bidimensionales y tridimensionales. Lo usan principalmente arquitectos e ingenieros.

Carácter: Número, letra o símbolo en la computadora, conformado por un byte.

CD-ROM: Compact Disk - Read Only Memory (Disco Compactor – Memoria de Solo Lectura). Disco compacto de sólo lectura. Tiene una capacidad de almacenamiento de hasta 650 megabytes, mucho mayor que la de un disquete.

Chip: Abreviatura de "microchip". Circuito muy pequeño, compuesto por miles a millones de transistores impresos sobre una oblea de silicio.

Comando: (Command) Instrucción que un usuario da al sistema operativo de la computadora para realizar determinada tarea.

Compresión de datos: Codificar datos para ocupar el menor espacio de almacenamiento posible.

Comprimir: reducir el tamaño de un archivo para ahorrar espacio o para transmitirlo a mayor velocidad. Uno de los programas de compresión más populares de Windows es WinZip.

CPU: Central Processing Unit. Unidad central de procesamiento. Es el procesador que contiene los circuitos lógicos que realizan las instrucciones de la computadora.

Cursor: símbolo en pantalla que indica la posición activa: por ejemplo, la posición en que aparecerá el próximo carácter que entre.

D

Data entry: ingreso de datos. Proceso de ingresar datos a una computadora para su procesamiento.

Debugging: (depuración). Corrección de errores o bugs.

Directorio: (Directory) grupo de archivos relacionados entre sí que se guardan bajo un nombre.

Disco rígido: Soporte giratorio de almacenamiento en forma de placa circular revestida por una película magnética. Los datos se graban en pistas concéntricas en la película.

Display: Unidad de visualización; monitor; pantalla.

Dpi: Dots Per Inch (Puntos Por Pulgada). En las impresoras, la calidad de la imagen sobre el papel se expresa en dpi.

DVD: Digital Versatile Disc (Disco Versátil Digital). Disco que posee gran capacidad de almacenamiento y sirve también para almacenar películas.

E

Emulación: Proceso de compatibilización entre computadoras mediante un software.

Encriptar: Proteger archivos expresando su contenido en un lenguaje cifrado. Los lenguajes cifrados simples consisten, por ejemplo, en la sustitución de letras por números.

Fibra óptica: Tecnología para transmitir información como pulsos luminosos a través de un conducto de fibra de vidrio. La fibra óptica transporta mucha más información que el cable de cobre convencional. La mayoría de las líneas de larga distancia de las compañías telefónicas utilizan la fibra óptica.

F

Firewall: mecanismo de seguridad que impide el acceso a una red.

Floating point package: (Sistema aritmético de punto flotante). Conjunto de programas que realizan las operaciones llamadas de coma flotante: suma, resta, multiplicación, división.

FPU (floating point unit): Unidad de coma flotante. También conocido como "coprocesador numérico", es un microprocesador que manipula números más rápidamente que una computadora personal. Las computadoras que no tienen unidad de coma flotante pueden realizar las operaciones aritméticas por medio de software específico.

FTP: File Transfer Protocol (Protocolo de Transferencia de Archivos). Sirve para enviar y recibir archivos de Internet.

Fuente: variedad completa de caracteres de imprenta de un determinado estilo.

G

Giga: Prefijo que indica un múltiplo de 1.000 millones, o sea 10^9. Cuando se emplea el sistema binario, como ocurre en informática, significa un múltiplo de 2^{30}, o sea 1.073.741.824.

Gigabit: Aproximadamente 1.000 millones de bits (exactamente 1.073.741.824 bits)

Gigabyte (GB): Unidad de medida de una memoria. 1 gigabyte = 1024 megabytes = 1.073.741.824 bytes.

Grid: (Grilla, cuadricula, rejilla). Sirve para para representar conjuntos de datos en forma de tabla.

H

Hardware: Todos los componentes físicos de la computadora y sus periféricos.

Help desk: Mesa de ayuda. Soporte técnico brindado a los usuarios telefónicamente por un proveedor de servicios de Internet.

Hertz: (Hercio) Unidad de frecuencia electromagnética. Equivale a un ciclo por segundo.

HTML: Hyper Text Mark-up Language (Lenguaje de Programación de Hipertextos). Lenguaje de programación para armar páginas web.

HTTP: Hypertext Transfer Protocol (Protocolo de transferencia de hipertextos). Es un protocolo que permite transferir información en archivos de texto, gráficos, de video, de audio y otros recursos multimedia.

I

Ícono: Imagen que representa un programa u otro recurso; generalmente conduce a abrir un programa.

IEEE: Institute of Electrical and Electronics Engineers: importante asociación de técnicos y profesionales, con sede en los Estados Unidos. Fue fundada en 1884 y en 1998 tenía aproximadamente 320.000 miembros en 147 países. Favorece la investigación en campos diversos, como la tecnología aeroespacial, la computación, las comunicaciones y la tecnología biomédica. Promueve la estandarización de normas.

Impresora: Dispositivo periférico que reproduce textos e imágenes en papel. Los principales tipos son: de matriz de puntos, de chorro de tinta y láser.

Interfaz: Cualquier clase de recurso físico o lógico que habilita para la comunicación interactiva entre sistemas de procesamiento de datos o entre computadores de un mismo sistema o entre sistemas de procesamiento de datos y el exterior. El teclado, por ejemplo, es una interfase entre el usuario y la computadora.

Internet: red de redes. Sistema mundial de redes de computadoras interconectadas. Fue concebida a fines de la década de 1960 por el Departamento de Defensa de los Estados Unidos; más precisamente, por la ARPA. Se la llamó primero ARPAnet y fue pensada para cumplir funciones de investigación. Su uso se popularizó a partir de la creación de la World Wide Web. Actualmente es un espacio público utilizado por millones de personas en todo el mundo como herramienta de comunicación e información.

Intranet: Red de redes de una empresa. Su aspecto es similar al de las páginas de Internet.

ISO: International Organization for Standardization (Organización Internacional para la Estandarización). Fundada en 1946, es una federación internacional que unifica normas en unos cien países. Una de ellas es la norma OSI, modelo de referencia universal para protocolos de comunicación.

J

JPEG: Joint Photographic Experts Group: nombre del comité que diseñó un estándar para la compresión de imágenes.

K

Keyword: Palabra clave para cualquier búsqueda.

Kilobit: 1.024 bits.

Kilobyte (KB): Unidad de medida de una memoria. 1 kilobyte = 1024 bytes.

L

LAN: Local Area Network (Red de Área Local). Red de computadoras interconectadas en un área reducida, por ejemplo, una empresa.

Latencia: Lapso necesario para que un paquete de información viaje desde la fuente hasta su destino. La latencia y el ancho de banda, juntos, definen la capacidad y la velocidad de una red.

Lenguaje de programación: Sistema de escritura para la descripción precisa de algoritmos o programas informáticos.

Linux: Sistema operativo gratuito para computadoras personales derivado de Unix.

Login name: Nombre de identificación del usuario en un sistema online.

Login: Conexión. Entrada en una red.

M

Mainframe: Estructura principal. Computadora de gran tamaño de tipo multiusuario, utilizada en empresas.

MB: Megabyte.

Megabit: Aproximadamente 1 millón de bits. (1.048.576 bits).

Megabyte (MB): unidad de medida de una memoria. 1 megabyte = 1024 kilobytes = 1.048.576 bytes.

Megahertz (MHz): Un millón de hertz o hercios.

Memoria caché: Pequeña cantidad de memoria de alta velocidad que incrementa el rendimiento de la computadora almacenando datos temporariamente.

Memoria flash: Tipo de memoria que puede ser borrada y reprogramada en unidades de memoria llamadas "bloques". Su nombre se debe a que el microchip permite borrar fragmentos de memoria en una sola acción, o "flash". Se utiliza en teléfonos celulares, cámaras digitales y otros dispositivos.

Microprocesador (Microprocessor): es el chip más importante de una computadora. Su velocidad se mide en MHz (Megahertz).

Módem: Modulador-demodulador. Dispositivo periférico que conecta la computadora a la línea telefónica.

Monitor: unidad de visualización; pantalla.

Motherboard: Paca o Tarjeta madre. Placa que contiene los circuitos impresos básicos de la computadora, la CPU, la memoria RAM y slots en los que se puede insertar otras placas (de red, de audio, etc.).

MS-DOS: Microsoft Disk Operating System: Sistema operativo del Disco Microsoft.

N

Navegador: Programa para recorrer la World Wide Web. Algunos de los más conocidos son Netscape Navigator, Microsoft Explorer, Opera y Neoplanet.

Net: World Wide Web.

O

Office: Suite de Microsoft para trabajo de oficina; incluye procesador de texto, base de datos y planilla de cálculo.

Online: En línea, conectado. Estado en que se encuentra una computadora cuando se conecta directamente con la red a través de un dispositivo, por ejemplo, un módem.

P

Página web: Una de las páginas que componen un sitio de la World Wide Web. Un sitio web agrupa un conjunto de páginas afines. A la página de inicio se la llama "home page".

Paquete (packet): La parte de un mensaje que se transmite por una red. Antes de ser enviada a través de Internet, la información se divide en paquetes.

Password: contraseña.

Periférico: Todo dispositivo que se conecta a la computadora. Por ejemplo: teclado, monitor, mouse, impresora, escáner, etcétera.

Pixel: Combinación de "picture" y "element". Elemento gráfico mínimo con el que se componen las imágenes en la pantalla de una computadora.

Procesador (processor): Conjunto de circuitos lógicos que procesa las instrucciones básicas de una computadora.

Protocolo: Lenguaje que utilizan dos computadoras para comunicarse entre sí.

Puerto: En una computadora, es el lugar específico de conexión con otro dispositivo, generalmente mediante un enchufe. Puede tratarse de un puerto serial o de un puerto paralelo.

Q

QBL (Query By Example): Consulta por ejemplo. Método de consulta para la base de datos.

QL (Query Language): Lenguaje de consulta.

Query: Consulta. Búsqueda en una base de datos.

R

RAM (Random Acces Memory): Memoria de acceso aleatorio. Memoria donde la computadora almacena datos que le permiten al procesador acceder rápidamente al sistema operativo, las aplicaciones y los datos en uso. Tiene estrecha relación con la velocidad de la computadora. Se mide en megabytes.

Rebutear: Volver a cargar el sistema operativo de una computadora que se ha "congelado".

Red: En tecnología de la información, una red es un conjunto de dos o más computadoras interconectadas.

Red de área local: LAN.

Resolución: Número máximo de pixeles que se ven en una pantalla. Dos ejemplos: 800 x 600 y 640 x 480. / En una impresora, la resolución es la calidad de la imagen reproducida y se mide en dpi.

ROM (Read Only Memory): Memoria de sólo lectura. Memoria incorporada que contiene datos que no pueden ser modificados. Permite a la computadora arrancar. A diferencia de la RAM, los datos de la memoria ROM no se pierden al *-apagar el equipo.

S

Servidor: Computadora central de un sistema de red que provee servicios y programas a otras computadoras conectadas.

Sistema operativo: Programa que administra los demás programas en una computadora.

Software: Término general que designa los diversos tipos de programas usados en computación.

SQL (Structured Query Language): Lenguaje de programación que se utliza para recuperar y actualizar la información contenida en una base de datos. Fue desarrollado en los años 70 por IBM. Se ha convertido en un estándar ISO y ANSI.

Suite: Serie, conjunto. Conjunto de programas que se comercializan en un solo paquete.

T

TCP/IP: Transfer Control Protocol / Internet Protocol. Es el protocolo que se utiliza en Internet.

U

Unix: Sistema operativo multiusuario, fue muy importante en el desarrollo de Internet.

URL: Uniform Resource Locator.

USB (Universal Serial Bus): es una interfase de tipo plug & play entre una computadora y ciertos dispositivos, por ejemplo, teclados, teléfonos, escáners e impresoras.

V

Virus: Pequeño programa que "infecta" una computadora; puede causar efectos indeseables y hasta daños irreparables.

W

Windows 2000: Versión del sistema operativo Windows, cuyo lanzamiento ha sido anunciado por Microsoft para el año 1999.

Windows 95: Sistema operativo lanzado por Microsoft en agosto de 1995.

Windows 98: Sistema operativo lanzado por Microsoft en 1998, como sucesor de Windows 95. Una de las más visibles diferencias con el anterior consiste en la integración del sistema operativo con el navegador Internet Explorer. Esta característica dio pie a un juicio por monopolio.

Windows NT Server: Windows NT diseñado para máquinas que proveen servicios a computadoras conectadas a una LAN.

Windows NT Workstation: Windows NT diseñado especialmente para empresas, se lo considera más seguro y estable que Windows 95 y 98.

Windows NT: Sistema operativo Windows de Microsoft diseñado para usuarios avanzados y empresas. En realidad se trata de dos productos: Windows NT Workstation y Windows NT Server.

Workstation: Estación de trabajo.Computadora personal conectada a una LAN. Puede ser usada independientemente de la mainframe, dado que tiene sus propias aplicaciones y su propio disco rígido.

World Wide Web: Red mundial; telaraña mundial. Es la parte multimedia de Internet. Es decir, los recursos creados en HTML y sus derivados. Sistema de información global desarrollado en 1990 por Robert Cailliau y Tim Berners-Lee en el CERN (Consejo Europeo para la Investigación Nuclear). Con la incorporación de recursos gráficos e hipertextos, fue la base para la explosiva popularización de Internet a partir de 1993.

WWW: World Wide Web.

Bibliografía

Libros

1. Adoración de Miguel, Piattini Mario, Marcos Esperanza. Diseño de bases de datos relacionales. Páginas 3, 289-298, 298-301, 393-400, 418-428. Ed. Alfaomega. Colombia. 2000.

2. Adoración de Miguel, Piattini Mario. Concepción y diseño de base de datos: del modelo E/R al modelo relacional. Madrid. Ed. Ra-Ma. 1999.

3. Cobb. In praise of 4GLs. En: Datamation, nro.15 de julio. Páginas 90-96. 1985.

4. Coll-Vinent, Robert. Información y poder. El futuro de las bases de datos documentales. Páginas 13-17. Ed. Herder. Barcelona. 1998.

5. Colmeraure A., Roussel P., "The Birth of PROLOG", ACM SlgPlan Notices, Vol. 28, No. 3, March 1993.

6. Connolly, Begg y Strachan (1996)

7. Fernández Ballesteros, Carlos. El Derecho de Autor y los Derechos Conexos en los Umbrales del año 2.000. Libro Memoria del Primer Congreso Iberoamericano de Propiedad Intelectual. Tomo I. Página 126.

8. Freedman, Alan. Diccionario de Computación. Software de The Computer Language Company Inc.Versión 5.2 (1981-1993).

9. Gardarin, Georges. Bases de Datos. Paginas 57-62. Ed. Paraninfo. Madrid. 1987.

10. Gillenson, Mark L..Introducción a las Bases de Datos. Páginas 98 y ss., 109 y ss., 155 y ss., 185 y ss., 241 y ss. Ed. McGraw-Hill. México, S.A. de C.V.1988; Villanueva Lara, Julio E. Ob.citada. Páginas 78 y ss.; y Kovacevic B., Antonio y González S., Alfredo.Sistemas de Información. Conceptos e implicancias para la empresa. Página 105 y ss. Ediciones Universidad Católica de Chile.1990.

11. Leber, J.B. 4GLs and 3GLs: Leading Dual Lives. En: Database Programming & Design 4:1 enero. 1991.

12. Lewis, T. The big software chill. En: IEEE Computer, marzo. Páginas 12-14. 1996.

13. Mcarthy, J. et al., LISP 1.5 "Programming Manual Cambridge" MA. MIT Press, 1965.

14. Nuñez Valencia, Oscar "Bancos de Datos y Propiedad Intelectual en Informática" (seminario realizado en Santiago en Septiembre de 1987) citado por Mayo, Marie Claude en "Informática Jurídica" y los comentarios de esta autora al respecto. Página 65. Editorial Jurídica de Chile. 1991.

15. Pérez V., Victor L. y Pino U., José A. Curso de Computación e Informática. Volumen II. Estructuras de datos y organizaciones de archivos. Página 17. Editorial Universitaria. 1992.

16. Piattini, M., García, F., Calidad en el desarrollo y mantenimiento del software. Página 15, 113-114. Editorial Alfaomega. México. 2003.

17. Piattini, M. Control interno y auditoria en un entorno de base de datos relacionales. En: ALIBase nro. 21 y 22 noviembre/diciembre. 1992.

18. Real Academia Española. Diccionario de la Lengua Española. Vigésima primera edición. Editorial Espasa Calpe S.A. Madrid, España. 1995.

19. Rochefeld, A. (1986). MERISE, an Information System Design and Development Methodology. En 5th International Conference on Entity-Relationship Approach, Dijon, Francia, 17-19 de noviembre.

20. Rolland, C., Foucaut, O y Bencl, G. (1988). Conception des systemes d'information. La méthole REMORA. París, Francia. Eyrolles.

21. Ruble, David A. Análisis y diseño practico de sistemas cliente/servidor con GUI. Pagina XXII, XXIII, 9, 15-17, 162, 163, 304-307. Ed. Prentice Hall. México. 1997.

22. Shan y Shixuan (1984). "Normal Entity-Relationship Model, A New Method for Enterprice Schema Design". En: IEEE.

23. Sommerville, I. (1988). Ingeniería del Software. Addison-Wesley Iberoamericana. México.

24. Stallings, William. Sistemas operativos. Prentice Hall, 2da edición

25. Tanguy, Jacques. Elección y compra del software de gestión. Paginas 7-15, 69-84. Ed. Deusto. Barraicúa, Bilbao. 1984.

26. Teorey, T. J. y Fry, J. P. (1982). Design of Database Structures. Englewood Cliffs, N. J.: Prentice-Hall.

27. Villanueva Lara, Julio E. Computadoras y Procesamiento de datos. Página 73, 127. Serie de Matemática. Monografía n° 28.Secretaría General de la O.E.A. Programa Regional de Desarrollo Científico y Tecnológico. Washington, D.C. 1987.

28. Wasserman, A. I. (1979). The Data Management Facilities of PLAIN. Proc. 1979 ACM SIGMOD International Conference on the Management of Data. Boston, Mass., mayo.

Referencias WEB

http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf
Metodologías de Desarrollo de Software

http://arraquis.dif.um.es/~rafa/bd1.htm
Rafael Menéndez Barzanallana

http://comunidad.ciudad.com.ar/argentina/entre_rios/dbinternet/metodo3.htm
Javier Alberto Besso, Cristian Fabian Borghello
Metodología del diseño distribuido. La metodología DATAID-D

http://elizabethpeguero.8m.com/Eliza.htm
Elizabeth Peguero

http://www3.uji.es/~mmarques/f47/apun/node1.html
Apuntes de Ficheros y Bases de Datos
María Mercedes Marqués Andrés
2001

http://www3.uji.es/~mmarques/f47/apun/node81.html
Metodología de diseño de bases de datos
María Mercedes Marqués Andrés
2001

http://www.berzal.freeservers.com/freeware/dbms/spanish.html

http://www.dbinternet.com.ar/metodo.htm

http://www.itlp.edu.mx/publica/tutoriales/basedat1/tema1_4.htm
http://www.itlp.edu.mx/publica/tutoriales/basedat1/tema7.htm
Base de datos orientados a objetos, Instituto Tecnológico de La Paz
Departamento de sistemas y computación
Yadira Hernández Ramírez
1999

http://www.lafacu.com/apuntes/informatica/base_datos/default.htm#Introducción

http://www.monografias.com/Computacion/Programacion/more2.shtml
Programación
http://www.monografias.com/trabajos11/basda/basda.shtml
Bases de datos
http://www.monografias.com/trabajos13/trsqlinf/trsqlinf.shtml
Manejadores de Bases de Datos SQL-ORACLE-INFORMIX
Monografías.com

http://www.programacionfacil.com/basic/cuatro4.htm
ProgramacionFacil.com

http://www.sindominio.net/biblioweb/pensamiento/softlibre/softlibre007.html
La definición de software libre
1996

http://www.uas.mx/cursoswebct/Progsist/material.htm
Universidad Anáhuac del Sur
México, Cd. De México. México

http://www.ur.mx/ur/faciya/carreras/cursos/sis/mod-dat1/graph.HTM
Facultad de Ingeniería y Arquitectura Universidad Regiomontana
Monterrey, Nuevo León. México

http://www.yudy.8m.com/Sistemasmanejador.htm