UNIVERSIDAD NACIONAL AUTONOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGON

SISTEMA DE CONTROL DE REPORTES PARA ATENCIÓN DE USUARIOS DE LAS SUPERCOMPUTADORAS

T E S I S

QUE PARA OBTENER EL TITULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

REYNA ELIZABETH CABALLERO CRUZ

DIRECTOR DE TESIS: ING. JUAN GASTALDI PÉREZ

MÉXICO **2003**

PAGINACION

DISCONTINUA

A mis padres y hermanos
por su amor y confianza.

A Cesar por el cariño
y la compresión.
A la UNAM.

Índice general

Índice de figuras

Índice de Tablas

Introducción

El Departamento de Supercómputo se destaca por la utilización de los instrumentos más avanzados con que cuenta el país, para el desarrollo de las investigaciones de alto rendimiento, esto es llamado tecnología de Supercómputo. Basada en sistemas con alta capacidad de procesamiento que posibilitan la ejecución de programas de cómputo muy complejos, así como modelos y simulaciones que permiten resolver diversos problemas científicos y tecnológicos a través de las Supercomputadoras.

Las Supercomputadoras se distinguen de cualquier otra computadora, fundamentalmente por dos aspectos: la velocidad y la capacidad, la velocidad de cálculo se encuentra entre las más altas en existencia y la capacidad de memoria y almacenamiento secundario permite resolver problemas complejos en tiempos menores que una computadora convencional, cuenta con canales de alta velocidad que permitan el movimiento de grandes cantidades de datos entre los diferentes dispositivos periféricos con la memoria central, y ésta con los procesadores de modo que las unidades de proceso central se mantienen ocupadas.

El Supercómputo en México tuvo su origen en noviembre de 1991 con la llegada de la CRAY YMP 4/464, que se caracterizaba por contar con 4 procesadores, cada uno de los cuales podía alcanzar una velocidad de hasta 333 millones de operaciones de punto flotante por segundo (Mflops), 64 millones de palabras en memoria central donde cada palabra es de 8 bytes y 128 millones de palabras en memoria secundaria de estado sólido.

Más tarde se vio reforzado con la llegada de la Origin 2000, adquirida a finales de 1996. Esta máquina que cuenta con 40 procesadores MIPS R10000, con un rendimiento máximo de 390 Mflops por procesador, siendo los procesadores escalares con caché secundario de 4 Mbytes.

Cabe hacer mención que este equipo es de los más poderosos dentro de las universidades latinoamericanas, y ha sido una pieza importante en el aumento de la investigación que realiza el país.

El Departamento de Supercómputo, dividido en Administración y Atención a Usuarios, tiene como prioridades resolver los requerimientos, problemas y la óptima utilización de los recursos. El trabajo de estas dos áreas se ha visto rebasado con la creciente demanda en el uso de esta tecnología. Creandose la necesidad de realizar mecanismos que permitan aumentar la eficiencia en los tiempos de respuesta a las necesidades de los usuarios.

La parte fundamental de este mecanismo es desarrollar un sistema de información de la supercomputadora, para mejorar la administración de los recursos. Éste debe comprender los reportes de uso de cpu, unidades de asignación de recursos (UCRS) y reportes de fallas. Debe ser un sistema que procese la información mensual y semestral, con las medidas de seguridad requeridas por el departamento.

Capítulo 1

Conceptos Básicos

1.1. Base de datos

La importancia de la información en la mayoría de las organizaciones, y por tanto el valor de la base de datos, ha llevado al desarrollo de una gran cantidad de conceptos y técnicas para la gestión eficiente de los datos.

Una base de datos es una colección de datos que tienen como objetivo primordial proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información.[1]

Las bases de datos poseen las siguientes características

- Conjunto de datos relacionados que tienen un fin común.
- Estructuración que permite la manipulación de datos.
- Colección de piezas de información interrelacionadas o independientes, almacenadas juntas sin necesidad de redundancia.
- Colección de datos relacionados cuya finalidad es la de compartir información.

1.2. Tipos de bases de datos

Ha sido tal la aceptación que han tenido las bases de datos que se han creado diferentes tipos, cada uno de ellos proporciona una metodología y técnica de programación diferente, ajustándose así a las necesidades del mundo moderno. A continuación se describirá brevemente cada uno de ellos.

1.2.1. Sistemas de bases de datos jerárquicas

Organizadas bajo una estructura de árbol, el problema es que la relación sólo se presenta en un sentido y dirección.

La estructura básica de este modelo de datos es el tipo de interrelación padre-hijo entre pares de tipos de registros. En la Figura 1.1 se define una base de datos con cinco tipos de registros estos son:*Departamento, Personal Administrativo, Profesor, Estudiante y Curso.*

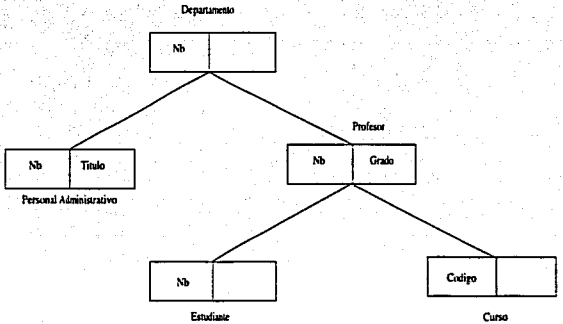


Figura 1.1: Esquema jerárquico o árbol de definición.

En el modelo jerárquico se tiene la conexión padre-hijo, y en consecuencia no puede haber un hijo sin un padre (no puede estar desconectado).

Una jerarquía debe obedecer a la estructura de un árbol. Así, el único nodo sin padre es el nodo raíz. Esto trae otra consecuencia, si se borra el padre también desaparecerán los hijos en la base de datos.

Además hay problemas con las relaciones muchos a muchos: un hijo no puede tener muchos padres, por lo que se debe repetir (duplicación de datos).

1.2.2. Sistemas de bases de datos de red

Una base de datos en red se construye a partir de **diagramas de estructura de datos,** (introducidos por Bachman en 1969).

En estos diagramas representaremos un tipo de registro por medio de unos cuadros que corresponden a tipos de registro y una línea que corresponden a ligas.

Para ilustrar esto, pensemos en una base de datos que represente la relación *Cuenta_habiente-cuenta* en un sistema bancario. Supongamos que deseamos una base de datos de red que almacene los datos de los cuenta habientes que pueden tener, cada uno de ellos, varias cuentas. Las cuentas a su vez pueden pertenecer a uno

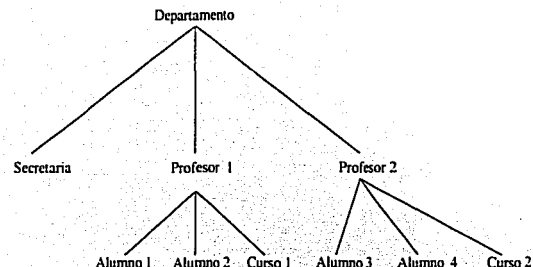


Figura 1.2: Ejemplo de una base de base de datos jerárquica.

o varios cuenta-habientes. Los datos del Cuenta_habiente que se quieren almacenar son: *nombre, domicilio y ciudad.* Los datos de la cuenta son: *número y saldo.* La relación es muchos a muchos y su diagrama de estructura de datos se muestra en la Figura 1.3.

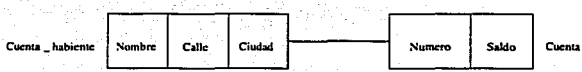


Figura 1.3: Diagrama de estructura de datos con relación muchos a muchos.

Para ver como se representa una relación uno a muchos supongamos que un cuenta habiente puede tener muchas cuentas, pero una cuenta podrá pertenecer a un solo cuenta habiente. El diagrama de estructura de datos de la Figura 1.4 corresponde a lo anteriormente expresado.

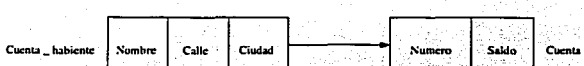


Figura 1.4: Diagrama de estructura de datos con relación de muchos a uno.

Si fuera el caso de una relación uno a uno, la línea tendría flecha en ambos lados de la línea Figura 1.5.

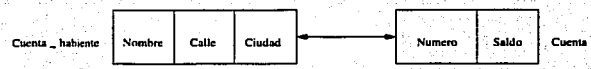


Figura 1.5: Diagrama de estructura de datos con relación uno a uno.

1.2.3. Sistemas de base de datos relacionales

El modelo de datos relacional representa la base de datos como un conjunto de tablas. Aunque las tablas son un concepto simple e intuitivo, existe una correspondencia directa entre el concepto de una tabla y el concepto matemático de una relación. Basados en el cálculo relacional y el álgebra relacional. Permiten extraer cualquier tipo de información en la base de datos.

El álgebra relacional permite que tenga mayor flexibilidad que otros modelos, por lo que extrae fácilmente la información que esté almacenada en la base de datos utilizando el lenguaje de consultas estructuradas (SQL[1] que esta basado en el álgebra relacional)

La Figura 1.6 es un ejemplo de base de datos relacional que muestra clientes y las cuentas que ellos tienen. Muestra, por ejemplo, que el cliente Hodges vive en Sidehill, en Brooklyn, y tienen dos cuentas, una con número 647, con un saldo de 105 366 dólares, y la otra con número 801, con un saldo de 10 533 dólares. Así como también que los clientes Shiver y Hodges comparten número de cuenta 647 debido a que son socios de la misma empresa.

Nombre	Calle	Ciudad	Numero
Lowery	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	647
Hodges	Sidehill	Brooklyn	801
Hodges	Sidehill	Brooklyn	647

Numero	Saldo
900	55
556	100 000
647	105 366
801	10 533

Figura 1.6: Ejemplo de una base de datos relacional.

[1]Structured Query Languaje.

1.2.4. Sistemas de bases de datos orientadas a objetos

Al igual que el modelo E-R, el modelo orientado a objetos se basa en una colección de datos de objetos. Un objeto contiene valores almacenados en *variables instancia* dentro del objeto. A diferencia de los modelos orientados a registros, estos valores son objetos por sí mismos. Así, los objetos a un nivel de anidamiento de profundidad arbitraria. Un objeto también contiene partes de código que operan sobre el objeto. Estas partes se llaman *métodos*.

Los objetos contienen los mismo tipos de valores y los mismos métodos se agrupan en *clases*. Una clase puede ser vista como una definición de tipo para objetos. Esta combinación de datos y código en una definición de tipo parecida al concepto de tipos abstractos en lenguajes de programación.

1.3. Manejadores de base de datos

Generalmente las bases de datos requieren una gran cantidad de información. Las bases de datos de las empresas comúnmente se miden en términos de gigabytes de información. Puesto que la memoria principal de la computadora no pueden almacenar esta información se guarda en discos. Los datos se transfieren entre el almacenamiento en disco y al disco es lento comparado con la velocidad de la unidad central de procesamiento de las computadoras, es imperativo que el sistema de base de datos estructure la información de tal manera que se reduzca la necesidad de transferir datos entre el disco y la memoria principal.

El objetivo de un sistema de base de datos es simplificar y facilitar el acceso a los datos. Las vistas de alto nivel ayudan a lograrlo. No debe abrumarse innecesariamente a los usuarios con los detalles físicos de la implantación del sistema. Sin embargo, uno de los factores primordiales para la satisfacción o insatisfacción del usuario con el sistema de base de datos es su funcionamiento.

Si el tiempo de respuesta para una consulta es demasiado largo, el valor del sistema se reduce. El funcionamiento del sistema depende de la eficiencia de las estructuras de datos utilizados para representar los datos en la base de datos y sucede en muchos otros aspectos de los sistemas de cómputo, deben hacerse concesiones, no sólo entre el espacio y el tiempo, sino también entre la eficiencia de un tipo de operación y la de otro.

Un manejador de base de datos es un módulo de un programa que constituye la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicaciones y las consultas hechas al sistema.

El manejador de base de datos es responsable de las siguientes tareas:

- **Interacción con el mensaje de archivos.** Los datos sin procesar se almacenan en el disco mediante el sistema de archivos proporcionado normalmente por un sistema operativo convencional. El manejador de base de datos tra-

duce las diferentes proposiciones en DML[2] a comandos de sistema de archivos de bajo nivel. Así, el manejador de base de datos se encarga realmente del almacenamiento, recuperación y actualización de los datos en la base de datos.

- **Implantación de la integridad.** Los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de limitantes de consistencia. Por ejemplo, el saldo de una cuenta bancaria no debe bajar de un mínimo previamente especificado (p. Ej., $25). de manera similar, el número de horas que puede trabajar un empleado en una semana no debe exceder un número específico (p. Ej., 80 horas). El administrador de la base de datos debe especificar estas limitantes, en forma explícita. Si se especifican estas limitantes, entonces el manejador de la base de datos puede verificar si las actualizaciones a la base de datos resulten en la violación de cualquiera de estas limitantes, y si así es, podrá realizar la acción apropiada.

- **Puesta en práctica de la seguridad.** Como se mencionó anteriormente, no es preciso que todos los usuarios de la base de datos tengan acceso a todo su contenido. Es labor del manejador de la base de datos hacer que se cumplan estos requisitos de seguridad.

- **Respaldo y recuperación.** Un sistema de cómputo, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallas. Existen muy diversas causas de estas fallas, entre ellas la caída de las cabezas lectoras de disco, la interrupción del suministro de energía y los errores de software. En cada uno de estos casos se pierde información de la base de datos. Es responsabilidad del manejador de la base de datos detectar estas fallas y restaurar la base de datos al estado que existía antes de presentarse la falla. Esto se logra normalmente iniciando diversos procedimientos de respaldo y recuperación.

- **Control de concurrencia.** Cuando varios usuarios actualizan la base de datos en forma concu-rrente, es posible que no se conserve la consistencia de los datos. Es necesario que el sistema controle la interacción entre los usuarios concurrentes; lograr dicho control es una de las tareas del manejador de la base de datos.

Algunos sistemas de base de datos, diseñados para utilizarse en computadoras personales pequeñas, no cuentan con varias de las funciones mencionadas. Esto da como resultado un manejador de datos de menor tamaño. Un manejador de datos pequeños requiere menos recursos físicos, sobre todo memoria principal, y su implantación es más económica.

1.3.1. MiniSQL

Mini SQL o mSQL, es un motor ligero para base de datos diseñado para pro-

[2] *Lenguaje de manipulación de datos.* Es un lenguaje que permite a los usuarios acceder o manipular los datos.

porcionar acceso rápido a datos almacenados con bajos requerimientos de memoria. Como su nombre lo indica, mSQL ofrece un subconjunto de SQL como su interfaz de consultas. Aunque sólo incluye un subconjunto de SQL (sin vistas, subconsultas, etc.) todas las características aceptadas cumplen con la especificación ANSI SQL. El paquete de mSQL incluye el motor de base de datos *(msqld)*, un programa terminal *(msql)*, un programa de administración de base de datos *(msqladmin)*, un visor de esquemas (relshow) y una API [3] de programación para lenguaje C.

La API y el motor de la base se han diseñado para trabajar en un ambiente cliente/servidor en una red TCP/IP.

La decisión de escribir otro paquete más de base de datos se debe al hueco en la gama de bases de datos. Cuando escribieron esto, no había otro paquete de bases de datos que aceptara a SQL como lenguaje de consultas. El paquete de base de datos más notable para trabajo de investigación, Postgres de la Universidad de California en Berkeley, ofrece como su lenguaje de consultas un superconjunto del Ingres QUEL original conocido como PostQUEL.

mSQL se ha desarrollado como el backend de una base de datos para el ambiente Minerva de Administración de redes de trabajo. Originalmente, Minerva utilizaba Postgres como su base de datos y generaba consultas PostQUEL para tener accesos a ella. Durante el período alfa inicial de pruebas de Minerva, se hizo el comentario que si Minerva generaba consultas SQL, los sitios con una instalación existente de base de datos, como Ingres u Oracle podrían utilizar sus bases de datos comerciales en vez de tener que soportar también Postgres. Para cumplir ese deseo, mSQL se escribió inicialmente como un traductor SQL a PostQUEL de manera que los sitios sin bases comerciales pudieran utilizar Postgres (en consideración a que no había motores SQL disponibles).

Conforme pasó el tiempo y Minerva continuó con su desarrollo, se hizo patente que Postgres consumía demasiados recursos para soportar los mecanismos en evolución proporcionados por Minerva. Para lograr velocidad, se amplio Minerva para efectuar monitoreo y adquisición de datos en paralelo. Desafortunadamente, cada proceso que se comunicaba con la base de datos forzaba la creación de una copia de la petición que se realizaba, ocupando casi 1.5 megabytes de memoria, esto detuvo las operaciones de adquisición de datos en paralelo.

Aunque Postgres es un paquete grande y bastante eficaz, sólo está soportado por un puñado de plataformas, lo cual resultó un problema cuando un par de examinadores alfa Minerva original deseaban correr en máquinas SGI y no pudieron participar en las pruebas. El hecho de que Minerva en sí utilizaba sólo una fracción de las características de Postgres y que requería ser más transportable, hizo patente que Postgres y Minerva no eran la mejor opción para combinar. A partir de esos hechos se desarrolló Mini SQL.

[3] *Interfaz para la programación de aplicaciones.* Es una biblioteca de funciones que usan los programadores para escribir programas de acceso de base de datos

1.4. Internet

La red Internet fue creada alrededor de 1970 por el departamento de defensa de los Estados Unidos de Norte América. En la actualidad, Internet interconecta a más de veinticinco mil redes. Por lo tanto Internet no es solamente una red, es una "Red de Redes".

El número total de computadoras en Internet, hasta el mes de agosto de 1994, era de 4 millones y el número de usuarios superaba los treinta millones. Además, el índice de crecimiento, tanto en redes como en usuarios finales, es del doce por ciento mensual.

Son muchas las ventajas que ofrece una red de computadoras, como es el compartir información; por lo tanto, si hablamos de Internet, que tiene más de cuatro millones de computadoras conectadas, nos damos cuenta de que el volumen de información al que uno puede acceder y ofrecer es muy grande.

La red Internet es conocida en el mundo, como la red de redes. Es un término usado para entender que se refiere a la interconexión de redes a través de todos los continentes y teniendo la cultura basada en la simplicidad, la investigación y estandarización. Hoy por hoy los que lideran las tecnologías de las redes provienen de la comunidad internet.

1.5. Servicios de internet

Existen más de sesenta y cinco mil herramientas en Internet, por lo que trataremos sólamente los más importantes.

1.5.1. Correo electrónico

El correo electrónico es una herramienta que permite intercambiar mensajes con cualquiera de los treinta millones de usuarios de Internet. Estos mensajes pueden ser desde simples textos ASCII hasta complejos archivos que incluyan hojas de cálculo, gráficos, fotografías etc.

Cada usuario de Internet tiene una dirección electrónica única que lo representa. Dicha dirección es para las computadoras de Internet una indicación clara y precisa de cuál es el camino que debe tomar el mensaje para llegar exactamente a su destino. Ejemplo de una dirección electrónica:

reyna@super.unam.mx

En donde se trata del usuario llamado **reyna**, de la red de nombre **super** del subdominio **unam** que indica la institución a la que pertenece, que en este caso es la **Universidad Nacional Autónoma de México**, por último el dominio **mx** indica que dicha institución se encuentra en México.

El correo electrónico permite al usuario acceder al conocimiento de las personas, solicitándoles información que no necesariamente se encuentra en el disco duro de una computadora.

Asimismo, le permite dirigirse a bases de datos automáticas, que interpretan su mensaje y le envían la información requerida, o intercambiar correspondencia con familiares y amigos en una forma mucho más rápida, económica y eficiente que el correo tradicional.

1.5.2. Listas de interés, servidores de listas

Una lista de interés está compuesta por un conjunto de direcciones electrónicas de personas que tienen un interés común. En Internet existen miles de listas de interés. Si el usuario desea recibir por ejemplo información sobre medicina, puede inscribirse en una lista de interés 'salud'. de esta forma toda la información que circule por la lista llegará vía correo electrónico al usuario.

Los servidores de listas son programas que permiten al usuario suscribirse o desuscribirse de cualquier lista de interés. Estos programas tienen una dirección electrónica y son accesibles por mail (correo electrónico).

1.5.3. Servidores automáticos de información

Los servidores automáticos de información son programas que permiten al usuario retirar archivos (información) de bases de datos con sólo enviar un mail (correo electrónico) a la dirección electrónica del servidor.

Estos servidores trabajan con comandos como Index y Help y envían automáticamente la información solicitada al usuario vía correo electrónico.

1.5.4. Servidores automáticos de usuarios (whois)

Los servidores automáticos de usuarios son programas que se encuentran en Internet y que permiten al usuario averiguar direcciones electrónicas de personas o instituciones a las que desee escribir. Son accesibles por correo y su respuesta es automática. Las herramientas que describiremos a continuación son sólo utilizables por usuarios TCP/IP.

1.5.5. Telnet

Permite al usuario hacer un salto a cualquier computador del mundo para ingresar a trabajar en éste de un modo interactivo. Una vez hecho el salto, el usuario podrá trabajar como si estuviera sentado frente a dicha computadora.

1.5.6. Gopher

Son miles de bases de datos que se encuentran en todo el mundo y a las que el usuario ingresa de modo interactivo. Una vez que el usuario entre a un gopher, comenzará a 'navegar' a través de menúes, entrando en una serie de directorios y subdirectorios hasta encontrar la información que requiera.

1.5.7. Talk

El Talk es una herramienta muy poderosa que permite al usuario conversar con cualquier otro usuario de Internet, sin importar en qué parte del mundo se encuentre. Una vez establecida la comunicación, la pantalla de ambos usuarios se dividirá en dos. Uno de ellos escribirá en la parte superior y el otro en la parte inferior, lo que permitirá una conversación en tiempo real.

1.5.8. FTP

Es otra herramienta poderosa que permite al usuario hacer un salto a un directorio del disco de cualquier computadora Internet, con la finalidad de copiar en su propio disco uno o varios archivos que le interesen.

1.5.9. WAIS, Archie, Verónica

Son herramientas que permiten al usuario hacer una búsqueda indexada de archivos o programas que desee conseguir. Por ejemplo si ingresamos la palabra 'Perú' en el Verónica, el programa nos devolverá automáticamente la lista de los archivos que se encuentran en Internet y que contienen la palabra 'Perú'.

1.6. World wide web

El World Wide Web, W3 o simplemente WWW es una de las herramientas más potentes en Internet y trabaja con miles de bases de datos en formato hipertexto y multimedia, lo que quiere decir que a través de ellas el usuario podrá encontrar gran cantidad de información que incluye fotografías, audio y vídeo en línea. Se describirla como un sistema global, distribuido interactivo dinámico, gráfico basado hipertexto, con plataforma de enlaces cruzados, que se ejecutan en Internet.

El Web es un sistema de información basado en el hipertexto. La idea en la que se basa el hipertexto es que en lugar de leer un texto siguiendo una estructura rígida y lineal (como un libro), es posible avanzar de un punto a otro fácilmente, obtener mas información, regresar el primer punto, brincar hacia otros temas y desplazarse (navegar) por el texto según los intereses que tenga en determinado momento. Por ejemplo el hipertexto para presentar información se utiliza en los sistemas de ayuda en línea proporcionados por la ayuda (help) de Microsoft Windows. Para obtener mas información sobre un tema en específico, solo se hace un clic, y aparecerá una nueva pantalla (u otra ventana, un cuadro de diálogo o lo que tenga definido el programa) con mas información. Tal vez esa ventana contenga vínculos que lo transportaran a alguna aplicación particular, así como referencias en esta última que lo remitirán todavía mas allá del tema original.

1.6.1. Servidor de web

Para publicar documentos en el Web, se necesita un servidor que proporcione documentos y medios de información al visualizador que se los solicite. El visualizador se comunica con el servidor para llegar al documento deseado. Como el Web lee documentos de servicios como FTP o Ghopher, puede utilizar tales herramientas para proveer documentos propios (La mayoría de los visualizadores deducen que leen documentos de hipertexto aún cuando éstos no lleguen por el cable utilizando específicamente HTTP) de cualquier manera, el mejor medio para publicar páginas de Web es a través del uso de un servidor de Web con enlace permanente.

Un servidor de Web utiliza el protocolo HTTP para atender las solicitudes de documentos que hacen los visualizadores. Luego, el servidor envía los archivos y las imágenes con las que dichos archivos tengan alguna referencia incorporada. Los servidores de Web también se configuran para interpretar comandos enviados desde el visualizador; por ejemplo, en el caso de los formularios u otras páginas de Web interactivas. Pero no puede hacer esto mismo con FTP ni con Ghopher.

Hay muchos servidores de Web disponibles, pero gran parte de la tecnología de los servidores está orientada hacia el lado de UNIX.

1.6.2. Funcionamiento del servidor web

Un servidor de Web es un programa asentado en una máquina conectada a Internet, en espera de que se enlace a él un visualizador para Web y se le solicite, por lo general, un archivo. Una vez que la solicitud le llega a través del cable, el servidor localiza y envía el archivo.

Los servidores y los visualizadores se comunican mediante el Protocolo de Transferencia Hipertexto (HTTP), un lenguaje especial creado especialmente para transmitir documentos de hipertexto a través de Web. A raíz de esto, los servidores Web suelen ser llamados servidores HTTPD. Donde la letra *D* representa la palabra "demonio"(daemon). Demonio es el término de UNIX con el que se nombra a un programa que espera en un segundo plano a que llegue una solicitud. Cuando recibe una, despierta, procesa la solicitud y se vuelve a dormir. No es necesario que esté en UNIX para que un programa actúe como un demonio, por lo que los servidores de Web de cualquier plataforma se siguen llamando HTTPD.

Cuando un servidor Web envía un archivo hacia un visualizador, también incluye información referente al tipo de tal archivo (por ejemplo, un archivo gif o una película Quicktime), de manera que el visualizador deduzca si puede leerlo por sí mismo o necesita recurrir a una aplicación auxiliar. También se puede ampliar el comportamiento del servidor para incluir archivos que tal vez no formen parte del conjunto por omisión.

A final de cuentas, los servidores de Web también se configuran para ejecutar scripts y/o programas basados en información que los lectores proporcionen desde sus respectivos visualizadores. Por ejemplo, se puede configurar una página de Web que le solicite al lector una cadena de búsqueda. Cuando el visualizador envía la

cadena a un programa del lado del servidor; este programa realiza la búsqueda y envía la cadena por buscar (insertada por el lector mediante su visualizador), el servidor pasa esa misma cadena a un programa del lado del servidor; este programa realiza la búsqueda y envía el resultado al servidor, que a su vez lo envía al visualizador.

A estos programas especialmente se les llama de compuerta (gateway) o guiones de compuerta, y son la base para crear formularios interactivos y mapas de imágenes propicios para utilizar el ratón.

1.6.3. Apache

Apache es un proyecto, un esfuerzo de colaboración para el desarrollo de software, creado de código robusto, poderoso y libre. El proyecto es manejado en común por un grupo de voluntarios situados alrededor del mundo, usando el Internet y el Web para comunicar, planear y desarrollar el servidor y la documentación relacionada con él. Estos voluntarios son conocidos como el grupo de Apache. Además, existen centenares de utilizadores que han contribuido con ideas, código, y documentación al proyecto.

En febrero de 1995, el software más popular para el servidor Web era el demonio del HTTPD del dominio público desarrollado por Rob McCool en el National Center for Supercomputing Applications (*NCSA*), de la Universidad de Illinois, Urbana. Sin embargo, el desarrollo del proyecto se detuvo debido a que el creador Rob McCool dejara el NCSA a mediados de 1994; para entonces muchos webmasters habían desarrollado sus propias extensiones y arreglos al código original (parches), en ese momento se percatarón que tenían una necesidad común. Así fúe como se creó un foro con la misión de administrar los parches, con el fin de coordinar sus cambios (en la forma de "correcciones"), comunicandose a través del correo electronico. Brian Behlendorf y Cliff Skolnick pusierón una lista de correo, un espacio compartido de información y cuentas para los desarrolladores del proyecto en una máquina en California, y el ancho de banda fue donado por HotWired. Para el final de febrero de 1995, los ocho contribuidores formaron el grupo de Apache:

- Brian Behlendorf, Roy T. Fielding, Rob Hartill,
- David Robinson, Cliff Skolnick, Randy Terbush,
- Roberto S. Thau y Andrew Wilson.

Con contribuciones adicionales de

- Eric Hagberg, Peters Franco y Nicolas Pioch.

Utilizarón httpd version 1.3 del *NCSA* como base, agregaron modificaciones al código, ya sea para arreglar problemas ó enriquecerlo, realizando pruebas en sus servidores, hasta llegar a la primera versión oficial de *Apache (0,6,2)*, en abril de 1995. Por coincidencia, el *NCSA* retoma su desarrollo durante el mismo período,

Brandon Long y Beth Frank de la NCSA se unierón a la lista del grupo de *Apache* como miembros honorarios, para compartir ideas y contribuir con programación.

El servidor de *Apache* era un golpe grande, pero el grupo sabia que el código necesitaba un reacondicionamiento y un reajuste general. Durante mayo y junio de 1995, mientras que Rob Hartill y el resto del grupo se enfocaba en implementar las nuevas características para la versión 0.7.x, apoyado por la comunidad de Apache creció rápidamente, Roberto Thau diseñó una nueva arquitectura del servidor, basada en el código llamado *Shambhala* que incluye una estructura modular y un API para un mejor uso de extensiones, así como una mejor asignación de memoria, y un modelo de proceso adaptable. El grupo cambió a esta nueva base del servidor en julio y agregó las características de 0.7.x, dando por resultado *Apache* 0.8.8 en agosto. Después de realizar pruebas con versión beta, muchos puertos fuerón cubiertos en las plataformas, y una nueva documentación se generó por David Robinson, y agregarón muchas características en sus módulos estándares, *Apache 1.0* fué la versión de diciembre de 1995.

1.7. Modelo cliente/servidor

Las tecnologías computacionales modernas buscan responder a las necesidades de los usuarios propios de la época, para ello se plantean nuevas formas de trabajo con las computadoras. Algunas de estas tecnologías son el uso de la red y modelos de software que lo permitan.

Uno de los modelos de software más utilizados para redes es el denominado cliente/servidor. Este esquema es un modelo de computación en el que el procesamiento requerido para ejecutar una aplicación o conjunto de aplicaciones relacionadas se divide en dos o más procesos que cooperan entre sí. La clave para comprender el concepto de cliente/servidor es entender la relación lógica entre una entidad que demanda un servicio (denominado cliente) a otra entidad que responde la petición a favor del cliente (denominado servidor). Usualmente la mayoría del trabajo pesado se hace en el proceso llamado servidor y el(los) proceso(s) cliente(s) sólo se ocupan de la interacción con el usuario (aunque esto puede variar).

Como su nombre lo indica, el proceso servidor provee los servicios al proceso cliente, normalmente por una vía de procesos específico que sólo ellos pueden entender. El proceso cliente se libra de la complejidad, solo envía su petición y puede realizar otro trabajo útil.

1.7.1. Servidor

Los servidores proporcionan un servicio al cliente y devuelve los resultados. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer el pedido, recibir su respuesta y enviarla al cliente. Además deben manejar los interbloques, la recuperación ante fallas y otros aspectos afines. Por las razones anteriores la plataforma computacional asociada con los servidores es más poderosa que la de los clientes.

Por esta razón se utilizan PC's poderosas, estaciones de trabajo, minicomputadoras o sistemas de cómputo grandes. Por otra parte deben manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema *(login)*, auditoría y recuperación y contabilidad. Usualmente en los servicios existe algún tipo de servicio de base de datos.

Que los clientes y los servicios puedan comunicarse se requiere una infraestructura de comunicaciones, la cual proporciona los mecanismos básicos de direccionamiento y transporte. La mayoría de los sistemas cliente/servidor actales se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo cual implica que las aplicaciones deben hacer las verificaciones. La red debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración.

1.7.2. Cliente

Los clientes interactúan con el usuario, usualmente en forma gráfica. Frecuentemente se comunican con procesos auxiliares que se encargan de establecer conexión con el servidor, enviar el pedido, recibir la respuesta, manejar las fallas y realizar actividades de sincronización y de seguridad.

Cualquier estación de trabajo puede usarse como cliente, si ésta puede compartir simultáneamente memoria a varios usuarios, entonces también puede usarse como servidor. Cuando una estación de trabajo cliente se encuentra conectada a una red de área local, ésta puede accesar a los servicios proporcionados por el servidor.

1.7.3. Modelo cliente/servidor para base de datos

La arquitectura cliente/servidor consiste en un conjunto de programas que pueden correr en la misma máquina ó en máquinas distintas. El proceso servidor se encarga de proporcionar un servicio, en el caso de bases de datos es el manejador ó motor de bases de datos. El proceso cliente solicita un servicio al proceso servidor, para bases de datos el cliente está asociado a la aplicación a través de la cual el usuario final interactua con el motor de bases de datos. Como lo ilustra la Figura 1.7.

1.8. Creación de interfaces gráficas.

Para la mayoría de los usuarios es necesaria una interfaz gráfica para accesar algún sistema; para nuestro caso en particular una base de datos. Las bases de datos pueden ser complejas que confunden de cuando en cuando, incluso a los usuarios más experimentados. Una interfaz gráfica de usuario (GUI), puede facilitar las cosas para el usuario y prevenir errores de datos o, por lo menos, reducirlos

1.8.1. Lenguaje de Marcación de Hipertexto

El lenguaje de Marcación de Hipertexto mejor conocido por las siglas *HTML* contiene tres elementos básicos, cada uno de los cuales se ha utilizado por separado

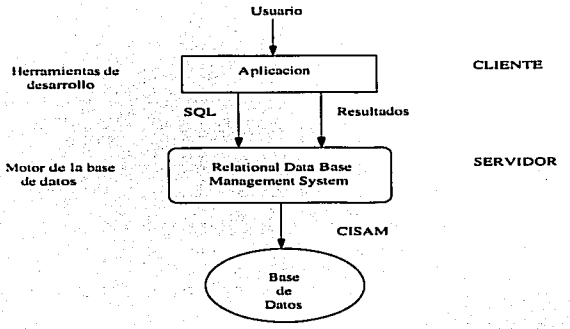


Figura 1.7: Representación del modelo cliente/servidor para bases de datos.

durante algún tiempo. Reuniendo las características de esos elementos se ha creado una nueva entidad mucho más útil para la programación de computadoras. Los tres elementos que componen *HTML* los indica su nombre.

- **Hipertexto**

  Es sólo un nombre complejo para un vínculo. Por lo general cualquier persona que ha utilizado algún tipo de ayuda, es casi seguro que utilizó un vínculo de hipertexto. Por ejemplo en un catálogo de biblioteca en línea, después de que ha buscado un libro en particular, generalmente se encuentra con el mensaje al final de la página, "véase también" seguido por una lista de otros libros y autores. Haciendo clic en alguno de los vínculos de hipertexto, podrá cambiar a otra sinopsis bibliográfica. Si eso no es lo que usted está buscando, puede regresar a la lista del libro original e intentar con otro vínculo.

- **Marcación.**

  Es simplemente lo que indica, un sistema de símbolos escritos en o dentro de un documento que permite al procesador, humano o electrónico, reconocer la forma de desplegar y formatear un documento.

- **Lenguaje.**

  Un lenguaje es un conjunto de símbolos o sonidos que permite la comunicación entre entidades. Si alguien habla francés y otra persona habla alemán, no podríamos comunicarnos muy bien. Pero si ambos hablamos español como

segunda lengua, tendríamos un medio común de comunicación y podríamos intercambiar ideas.

Reuniendo los tres elementos se obtiene un lenguaje de marcación para hipertexto, un conjunto común de símbolos para formatear documentos, crear vínculos con otras páginas Web y tener comunicaciones funcionales entre navegadores Web de todo el mundo. Un documento HTML bien escrito se ve exactamente igual en cualquier tipo de navegador Web

Los documentos HTML se componen exclusivamente de texto. Parte de éste es el conjunto de instrucciones para que el navegador haga algo especial con el texto asociado sencillo. Esas instrucciones se llaman etiquetas. Las etiquetas se distinguen del resto porque utilizan símbolos de "mayor que"(>) y "menor que"(<). Un < inicia la etiqueta y un > la termina. La mayoría de las etiquetas se utilizan en pares, para activar y desactivar una función de formato en torno de un bloque de texto. La instrucción de desactivo es igual que la de activado sólo que tiene una diagonal (/) después del signo de inicio de la etiqueta.

1.8.2. Common Gateway Interface

Uno de los elementos más potentes de HTML es la Common Gateway Interface (CGI). HTML contiene etiquetas que le permiten al usuario la elaboración rápida de documentos de aspecto profesional. Los diseñadores de HTML crearon una entrada en el estándar HTML que permite a los programadores ejecutar un programa escrito en cualquier lenguaje que deseen usar. Esto se llama Common Gateway Interface. Un gateaway es un conexión con el sistema operativo externo. CGI proporciona a los programadores un modo de que las páginas Web en HTML pueden ejecutar programas externos y presentar los resultados.

1.8.3. Pasos del CGI

La acción de llamar un programa CGI desde un navegador Web es muy sencilla para el usuario, lo cual es uno de los principales atractivos del Common Gateway Interface. Sin embargo, desde la perspectiva del programador, el proceso es un poco más complicado. Existen varios pasos a seguir para que un programa CGI funcione adecuadamente:

1. El usuario llama un programa CGI haciendo clic sobre un vinculo u oprimiendo un botón.

2. El navegador solicita autorización al servidor de Web para ejecutar el programa CGI.

3. El servidor Web revisa la configuración y los archivos de acceso para asegurarse que el solicitante tiene permitido el acceso al programa CGI.

4. El servidor web se asegura de que exista el programa CGI.

5. Si existe el programa, éste se ejecuta.

6. Cualquier resultado producido por el programa CGI se devuelve al navegador Web.

7. El navegador Web despliega el resultado.

El navegador web puede transferir información al programa CGI de varias maneras, y el programa puede devolver los resultados con etiquetas HTML incluidas, como texto sencillo o como una imagen. El navegador Web interpreta los resultados de la misma manera como lo hace con cualquier otro documento. Con un clic del ratón, este proporciona una herramienta muy poderosa para ejecutar prácticamente cualquier programa, y permite a los programadores el acceso a cualquier base de datos externa, que proporcione una interfaz de programación.

1.8.4. PHP

PHP *acrónimo de PHP Hipertext Preprocessor.* Es un lenguaje de programación de páginas web que funciona en el lado del servidor. Técnicamente es un lenguaje interpretado de alto nivel, similar en construcciones léxicas y sintácticas a Perl, C e incluso Java y embebido en páginas *HTML*.

PHP es multiplataforma, ya que es un lenguaje interpretado. Esto permite que soporte casi todas las variantes de Unix y, ahora, también existe una versión para Windows 95 y Windows NT. Puede correr como un CGI o como un módulo. Tratado como un CGI puede correr sobre cualquier servidor que soporte CGI's. Cuando es compilado como un módulo, debe haber uno por cada servidor Web. Hay módulos escritos para el Servidor Apache de UNIX y muy pronto los habrá para Microsoft Internet Información Server, para Enterprise de Netscape, y para Apache para Windows. Gran integración a Bases de Datos tiene interfaces nativos para Bases de Datos tan populares como DBMS, Oracle o Sybase. Además, la versión 3.0 incluye soporte ODBC para conectar con la mayoría de las Bases de Datos bajo NT.

También permite el manejo de "pipes" y "sockets", haciendo muy fácil el poder enviar un mail, conectarse con servidores remotos (mediante el soporte de URLs, nuevo en la versión 3.0), o communicarse con aplicaciones externas. Soporta autenticación HTTP, aunque sólo si funciona como módulo en un servidor Apache.

Permite actualizar archivos que, por defecto, son guardados en un subdirectorio del servido. Esto es posible introduciendo la variabiable de entorno de nombre TMPDIR en donde PHP esté corriendo, o editando el fichero php.h y definiendo la variable de ambiente de nombre UPLOAD_TMPDIR.

Soporta la librería de imágenes GD escrita por Thomas Boutell, que concediendo generar imágenes en formato GIF dinámicamente.

Permite "Safe Mode". es decir, un mecanismo de seguridad que permite que varios usuarios estén corriendo scripts PHP sobre el mismo servidor. Este mecanismo está basado en un esquema de permisos de archivos, concediendo el acceso a aquellos archivos que son apropiados por el mismo identificador de usuario que el del script que

está intentando acceder a ese fichero, o bien cuando el fichero está en el directorio que es propiedad del mismo identificador de usuario que el del script que está intentando acceder.

Otras características importantes son el soporte de cookies HTTP de forma transparente, o la posibilidad de trabajar sobre configuraciones de máquinas virtuales, soportadas por algún dominio HTTP.

Y terminamos con lo que sin duda es atractivo para todo el mundo, *PHP es totalmente gratis*. Es un producto distribuido bajo licencia GPL [4] de GNU, por lo que permite usar el software para cualquier propósito. Este software se puede obtener en las direcciones **http://www.php.net o en http://php.iquest net**

1.8.5. Inicios de PHP

Todo comenzó con un CGI escrito en Perl que programó el danés, ahora residente en Canadá, Rasmus Lerdorf, para su uso personal, que más tarde reescribió al lenguaje C, para disminuir el número de recursos utilizados y aumentar la velocidad. Al mismo tiempo, desarrolló la herramienta conocida como FI (Form Interpreter), para embeber consultas SQL en páginas HTML y manejarlas mediante formularios. Cuando la gente que visitaba su página comenzó a solicitar su código, Lerdorf agrupó estas dos herramientas en la primera versión de PHP/FI, que significaba Personal Home Pages/Form Interpreter.

A medida que los usuarios se familiarizaron con este paquete, comenzaron a solicitar nuevo código, con facilidades para macros, bucles, arrays, etc. Este fue el motivo de que Lerdorf desarrollara un sencillo analizador sintáctico, surgiendo así la versión 2.0 de PHP/FI.

A medida que la popularidad de este lenguaje crecía, la gente comenzó a contribuir elaborando su propio código y depurando el existente. Pero la mayor contribución llegó de manos de Zeev Suraski y Audi Gutmans, que desarrollaron un nuevo analizador sintáctico y escribieron un API para facilitar las extensiones del PHP mediante la introducción de nuevos módulos. Este fue el comienzo de la colaboración de grandes desarrolladores y profesionales; otros fueron Stig Bakken, responsable del soporte para Oracle; Shane Caraveo, encargado de portar el PHP a Windows 95/NT y Tim Winstead, quien capturó todos los errores. Esta fue la principal razón por la que el nombre de "Personal Home Pages"fuera sustituido por el que actualmente define a la última versión de este lenguaje.

[4] General Public License

Capítulo 2

Análisis y diseño del sistema

2.1. Estudio preliminar

El Departamento de Supercómputo, compuesto de las áreas de Administración de Supercómputo y Atención a Usuarios, tiene como misión proporcionar servicio de alta calidad a la comunidad de usuarios del país que requieren la utilización de computo científico numérico intensivo.

Esto se logra manteniendo el buen funcionamiento de las Supercomputadoras, dando soluciones pertinentes a los problemas y ofreciendo a la comunidad de usuarios de supercómputo, las diversas alternativas y mecanismos para la mejor utilización de los recursos de las supercomputadoras.

Debido al crecimiento en el uso de las supercomputadoras, estos servicios se han visto afectados en la solución de los conflictos ya que el mecanismo actual es insuficiente por diversos motivos, entre ellos destacamos la falta de comunicación entre el personal debido a la separación física y logística de las áreas, que el número de problemas excede al número de personas que puede resolverlo, la poca disposición por parte del personal de soporte SGI y la falta de datos oportunos que permitan evaluar mejor a los usuarios y dar un mejor servicio a los mismos, proporcionándoles los recursos óptimos para realizar sus proyectos.

Cabe destacar que la sintonización del equipo con los usuarios es de gran valía, debido a que muchos proyectos importantes que realiza el país dependen del buen funcionamiento de éste.

El departamento contaba con un sistema, pero su funcionamiento es lento en la generación de reportes, lo que provoca demoras e ineficacia; esta es la razón principal por la cual, se sometio a una evaluación para lograr mejoras en el mismo. De la evaluación se concluyo los siguientes puntos:

1. Sistema realizado en un servidor *SUN, SPARCstation 4*, con sistema operativo *Solaris 2.7* con el siguiente software:
   - MiniSQL
   - Apache

- Perl
- Mod_Perl

2. La causa de la demora en el funcionamiento del sistema, es el modo de programación de los CGI's realizados en *Perl*, y a pesar de contar con el modulo *Mod_Perl* para *Apache*, los tiempo en la emisión de reportes oscilaban desde 1 minuto hasta 12 horas, esto dependendia del tipo de reporte.

3. Las pruebas al manejador de la base datos *mSQL* resultarón positivas, ya que a pesar de ser muy ligero, para las necesidades del departamento las cubría.

4. La modificación de programas resultaba complicada, por la estructura que estos tenian.

5. Se requerian nuevas tablas, y la insercion de estas derivaba en tranformar los programas.

6. En el caso del servidor se dedujo que su rendimiento es *aceptable*, ya que en la compilación y ejecución de diversos programas los resultados se encontraban en la media de los equipos del departamento. Más sin en cambio el problema hallado es el tiempo de vida del equipo.

Lo anteriormente mencionado ha creado la necesidad de desarrollar un nuevo sistema poseedor de información de las supercomputadoras, para mejorar la administración de los recursos. Éste debe comprender los reportes de uso de cpu, memoria y disco; unidades de asignación de recursos (UCRS), uso del sistema de colas (NQE, MISER) y reportes de fallas. Así mismo debe ser un sistema automatizado que permita procesar rápidamente la información emitida mensualmente y contar con las medidas de seguridad en su utilización, donde los objetivos buscados son:

- Controlar la administración de usuarios, consumo de recursos y emisión de reportes.
- Poseer una interfaz gráfica capaz de ingresar, actualizar, borrar y controlar la información del sistema, con las medidas de seguridad pertinentes.
- Servir como banco de información que contenga la solución de problemas para retro-alimentar, y dar las debidas soluciones de manera rápida.
- Proporcionar la información necesaria para realizar análisis de comportamiento de problemas, para solucionar los cuellos de botella.
- Proporcionar la información necesaria para la toma de decisiones de los miembros del Comité Académico de Supercómputo con respecto a la Asignación de Recursos de Supercómputo.
- Realizar un sistema en el WWW que goce de privacía del departamento de Supercómputo.

2.2. Estudio de factibilidad

En esta etapa se debe entender y describir los procedimientos de la posible solución. Dentro de este estudio se llevará a cabo una serie de evaluaciones con las cuales sabremos si la propuesta es valida o no lo es.

2.2.1. Tipos de factibilidad

Los recursos serán analizados en relación con el tipo de factibilidad: *técnica, económica y operativa.*

Factibilidad técnica. Consiste en determinar si existen las herramientas técnicas; por lo general la respuesta a si una tecnología se encuentra disponible y si llegaría a satisfacer las necesidades de los usuarios.

Factibilidad económica. Identifica los recursos básicos que deben considerarse como es el tiempo, y el tiempo del equipo de análisis de sistemas, el costo de la realización integral de un estudio de sistemas, el costo del tiempo del empleado para la empresa, el costo estimado del equipo y el costo estimado del software comercial o de su desarrollo.

Factibilidad operativa. La factibilidad operativa depende de los recursos humanos que participan durante la operación del proyecto. Mediante esta se puede conocer que tan bien un sistema automatizado de información trabaja en un entorno dado, aunque normalmente en los sistemas este es el tipo de factibilidad que con más frecuencia se da por sentado que funcionará o que a pesar de no elaborarlo el proyecto se desarrollará correctamente, de manera general podemos decir que este tipo de estudio pretende determinar si el nuevo sistema se utilizará tal y como esta planteado.

2.2.2. Evaluación de factibilidad

1. *Evaluación técnica.*
   Dado que el Sistema de Información del Departamento de Supercómputo se está desarrollando en una área que no se dedica explícitamente al desarrollo de sistemas sino a su administración, el equipo de desarrollo debe trabajar con el equipo y software disponible en el mismo departamento.

   El Departamento Supercómputo por ser un área de alto nivel tecnológico, cuenta con Redes ya instaladas (LAN), así como la tecnología de respaldos y recuperación de datos (Almacenamiento en cintas DAT 4mm DDS3 12 Gbytes c/u). Así mismo cuenta con conexión a RedUNAM y por ende a Internet (WAN), con un equipo de Administradores tanto de sistemas como de Red que constantemente monitorean todas las operaciones de entrada y salida de datos, para detectar posibles problemas de conectividad.

2. *Evaluación económica.*
   La situación económica en que se encuentra la UNAM es difícil, posee pocos recursos económicos y deben ser administrados de la mejor forma, esta es la razón de adquirir hardware sobre el software; en la actualidad en sofware libre es variado, robusto y consistente en la creación de aplicaciones. Para nuestro caso particular el sistema se concebio con software perteneciente a la ideología OpenSource (Software Libre) que implica bajo costo, amplias posibilidades de adecuación a nuestras necesidades y un gran esfuerzo para implementar los mecanismos de seguridad adecuados.

3. *Evaluación operativa.*
   Los usuarios del sistema será el personal del Departamento de Supercómputo que está formado por *académicos y becarios*, estos cuentan con un nivel de estudios que varia desde licenciatura con un avanze de credito del 50 % hasta posgrados, con carreras ó especialidades afines a temas de computación. Debido a que dentro de las actividades del personal, la utilización de diferentes aplicaciones gráficas es una tarea que comúnmente se enfrenta en la realización del trabajo, esta no representará ningún problema para la interacción del sistema.

2.3. Planeación y control de actividades.

Dos puntos importantes dentro del desarrollo de un sistema es la estimación de costos y tiempo. Por medio de esta valoración se podrán controlar los avances de nuestro proyecto, para así coordinar de forma efectiva las actividades necesarias y asegurar que estas se terminen dentro de lo estimado y que no rebasen los costos programados.

Esta tarea la realizaremos con el método denominado Diagramas de Gantt, comenzaremos por listar las actividades a realizar:

1. Análisis de datos.
2. Diseño del sistema.
3. Análisis de recursos.
4. Desarrollo del sistema.
5. Diseño de archivos.
6. Desarrollo de datos de prueba
7. Pruebas
8. Implantación del sistema.

A continuación en la Figura 2.1 se presenta el diagrama de Gantt, donde se describe la relación de las actividades contra el tiempo de desarrollo de cada una.

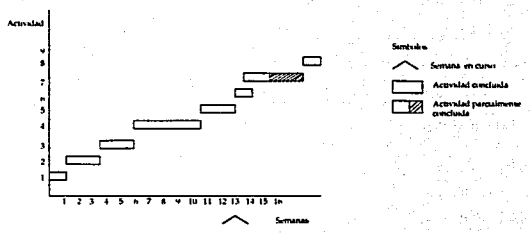


Figura 2.1: Diagrama de Gantt.

2.4. Análisis de datos.

Con el propósito de determinar los requerimientos del sistema, se debe conocer como trabaja, para esto se utilizarían técnicas que permitan establecer el flujo de la información.

2.4.1. Análisis de flujo de datos.

Como primera parte en el desarrollo del flujo de datos realizamos un diagrama de contexto Figura 2.2, en él se presenta un esbozo del sistema.

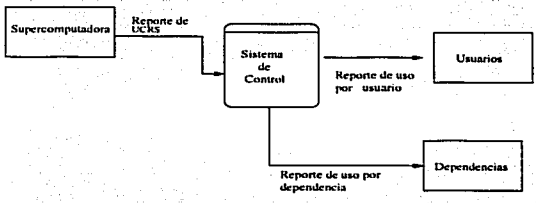


Figura 2.2: Diagrama de contexto.

En la Figura 2.3 se muestra el diagrama de datos de nivel cero, en el se representa el movimiento de los datos a lo largo del sistema de control, comenzando en la parte superior izquierda, dirigiéndose hacia abajo y a la derecha.

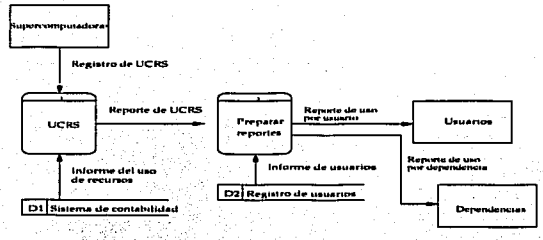


Figura 2.3: Diagrama de flujo nivel cero.

Para el proceso 2 (sistema de control) se realizó un diagrama de flujo de datos nivel 1 Figura 2.4, para este caso se desgloso dicho proceso.

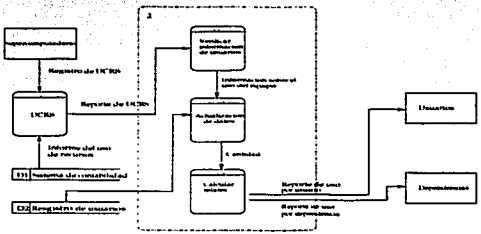


Figura 2.4: Diagrama de flujo nivel uno.

Cabe mencionar que no se describe el proceso de UCRS, debido a que este ya es un sistema, que actualmente se encuentra en operación.

2.4.2. Diccionario de datos

El diccionario de datos se caracteriza por poseer "información acerca de los datos" con la finalidad de recopilar, coordinar y confirmar la transición de los datos. Para esto la información esencial de cada dato, se plasmará en las formas y notaciones

UCRS

Descripción: *Unidades de control de recursos del sistema.*

Entradas	Descripción	Salidas
Reporte de UCRS	Asignación, verificación y control de los recursos.	Reporte de uso.
Información de uso		

Tabla 2.1: Tabla de diccionario de datos para el proceso UCRS.

SISTEMA DE CONTROL

Descripción: *Verificar, actualizar, generar reportes del consumo de recursos, durante algún periódo para los usuarios.*

Entradas	Descripción	Salidas
Reporte de uso	Verificar información de usuarios	
Registro de usuarios	Actualizar información Calcular totales	Usuarios

Tabla 2.2: Tabla de diccionario de datos para el proceso Sistema de control.

de C. Gane y T. Sarson [1]

Comenzaremos en el nivel superior del diagrama de flujo de datos, catalogando los procesos esenciales en la tabla 2.1, se tiene un registro de las entradas al proceso UCRS, mostrando el resumen y las salidas del mismo. En la tabla 2.2 se realiza la misma operación para el siguiente proceso. Obteniendo un nuevo flujo de datos, que será el resultado de los procesos anteriores (UCRS y Sistema de Información) mostrados en las tablas 2.3, 2.4 y 2.5 siguiendo este movimiento de los datos, toca el turno para el almacén de los datos al que se tiene acceso, esta tarea se representa en las tablas 2.6 y 2.7.

Por último realizamos la estrutura de datos tabla 2.8, esta se compone de los datos elementales relacionados. Para esto se requiere el nombre de la estructura de los datos. así como la representación simbólica en la esquina superior derecha; como en las tablas anteriores.

[1] Structured Systems Analysis and Design Tools and Tecniques, Englewood Cliffs, N.J. Prentice-Hall, Inc. 1979

REPORTE_DE_UCRS

Descripción: *Información de las unidades de control de recursos de las supercomputadoras durante algún periodo.*

Fuente	Destino
UCRS	Actualizar datos de uso.

Tabla 2.3: Tabla de diccionario de datos para el flujo de datos denominado Reporte_de_UCRS.

REPORTE_DE_USO_POR_USUARIO

Descripción: *Información del consumo de las supercomputadoras durante algún periodo por usuario.*

Fuente	Destino
Sistema de control	Usuarios

Tabla 2.4: Tabla de diccionario de datos para el flujo de datos denominado Reporte_de_uso_por_usuario.

REPORTE_DE_USO_POR_DEPENDENCIAS

Descripción: *Información del consumo de las supercomputadoras durante algún periodo por dependencias.*

Fuente	Destino
Sistema de control	Dependencias

Tabla 2.5: Tabla de diccionario de datos para el flujo de datos denominado Reporte_de_uso_por_dependencias.

SISTEMA_DE_CONTABILIDAD

Descripción: *Información de la utilización del sistema.*
Contenido:

- Información del consumo de CPU, memoria y disco
- Comportamiento del sistema
- Calendarización de procesos

Flujo de datos entrantes	Flujo de datos salientes
Supercomputadoras	Datos de uso de recursos.

Tabla 2.6: Tabla de diccionario de datos para el almacen de datos denominado Sistema_de_contabilidad.

RESGISTRO_DE_USUARIOS

Descripción: *Información de usuarios.*
Contenido:

- Datos del usuario
  - Máquina donde trabaja
  - Nombre
  - Apellido Paterno
  - Apellido Materno
  - Dependencia
  - Telefono
  - Email
  - Estatus
  - Reportes: Consumo de UCRS (login,mes, año)

Flujo de datos entrantes	Flujo de datos salientes
Reporte de UCRS	Datos de uso de recursos por usuario.

Tabla 2.7: Tabla de diccionario de datos para el almacen de datos denominado Registro_de_usuarios.

REPORTE_DE_USO_USUARIOS

Descripción breve: *Información de la utilización del sistema para cada usuario*
Descripción:

- Información del consumo de CPU, memoria y disco
- Reportes mensuales, semestrales, y/o extraordinarios
- Información del usuario

Flujo relacionados	Volumen
Sistema de control	4 mensuales 500 semestrales n extraordinarios

Tabla 2.8: Tabla de diccionario de datos para el almacen de datos denominado Reporte_de_uso_usarios.

2.5. Actividades clave

Enseguida se mencionan las tareas, con la finalidad de aplicar criterios que apoyen a la realización del sistema, pero sobre todo al análisis de recursos y el diseño del mismo.

Previsiones Administrativas

- Contratos de mantenimiento preventivo y correctivo de equipos.
- Establecer condiciones de garantía de equipos nuevos.
- Seminarios de capacitación y adiestramiento del personal involucrado.
- Actualización del sitio Web del Departamento de Supercómputo y de la Intranet del Área de Administración de sistemas Unix del mismo.

Creación de Documentación

- Establecer los mecanismos adecuados para documentar el Inventario de Software y Hardware utilizado en el desarrollo del sistema.
- Establecer los mecanismos de comunicación de resultados de las actividades de monitoreo de la red para establecer horas pico y determinar posibles soluciones a problemas.
- Plan de Recuperación de Desastres. Establecer los mecanismos de recuperación en caso de incidentes con el fin de garantizar la continuidad del servicio en caso de contingencia.

Mecanismos de soporte para sistemas OpenSource

- Ligas a páginas oficiales
- Listas de correo
- Grupos de discusión
- Sitios FTP

Infraestructura de Administración

- Dado que el área donde se desarrolla el sistema es un área de administración de sistemas, todos los involucrados son en esencia administradores de sistemas, por lo cual se cuenta con todos los conocimientos necesarios para una correcta administración tanto de los sistemas, servidor, red y seguridad de los mismos.

Soporte a Usuarios

- Usuarios de supercómputo que soliciten conocer su perfil y comportamiento de uso de recursos.

- Personal de alto rango directivo que requiera conocer estado de sus subordinados, como por ejemplo: Directores de Escuelas, Centros e Institutos.
- Miembros del Subcomité Académico de Supercómputo que requieren tener las bases adecuadas de información para asignar de manera correcta los recursos de supercómputo a los usuarios.
- Miembros del Comité Académico de Supercómputo que requieres establecer políticas y lineamientos que seguirá la UNAM en materia de cómputo numérico intensivo.
- Miembros de la Junta de gobierno y Rectoría que deben aprobar los planes de desarrollo y presupuestos en materia de supercómputo para la UNAM y el país en general.
- Personal de Administración del Departamento de Supercómputo que realiza el seguimiento y contabilización de recursos de manera continua.

Telecomunicaciones

- Reconfiguración parcial de la red interna de supercómputo con el fin de incrementar el ancho de banda en las comunicaciones.

2.6. Análisis de recursos

Es importante determinar los recursos (hardware y software) con los que se cuentan, ya que son esenciales para desarrollar un sistema eficiente. Debido a que la tecnología es determinante en la productividad de cualquier proyecto. En la tabla 2.9 se evalua los recursos esenciales en la elaboración del sistema, el proposito buscado, para cada uno de ellos y la posibilidad con que cuenta el departamento, para realizarlo.

2.7. Diseño Físico

Se propone la renovación del equipo de cómputo, específicamente los clientes del sistema, ya que actualmente se cuenta con los siguientes equipos:

1. SGI Indy 32Mb RAM, Disco Duro de 2Gbytes, Sistema Operativo IRIX 6.5.10f Tarjeta Ethernet 10 Mb/s.
2. SUN Sparc4 64 Mb RAM, Disco Duro de 3Gbytes, Sistema Operativo Solaris 8 Tarjeta Ethernet 10 Mb/s.

Por equipos más nuevos con la siguiente especificación:

1. 2 Pentium 4, 128 Mb RAM, Disco Duro IDE 40 Gbytes, Sistema Operativo Linux RedHat 7.0, Tarjeta FastEthernet 100 Mb/s.

Con el fin de mejorar el rendimiento en las comunicaciones en la red de área local (LAN), o nuestro propio segmento de red, se propone segmentar la red con un switch que permita aislar el tráfico de la red propia del departamento, del tráfico de la red interna del sistema.

En cuanto a la administración se aplicarán las políticas que sigue el departamento de administración.

Para el software, se desea una interfaz gráfica. Para este caso se ha decidido que sea cualquier programa que soporte el servicio WWW (World Wide Web), ya sea netscape, opera, mozilla, explorer etc, por la facilidad con que se puede interactuar con dichos programas. Es por ello que este sistema se podra utilizar a través de una página, este desarrollo contará con las medidas de seguridad pertinentes. Donde solamente las direcciones permitidas y usuarios podrán hacer uso de este servicio. Se podrán realizar consultas a los usuarios, reportes de uso de recursos y problemas, donde se podrá especificar el formato de salida.

Las consultas se deberán hacer por medio del web. Para esto se pensó en utilizar CGI, pero se realizaron pruebas de análisis de rendimiento y estos no eran tan rápidos como se requería. Entonces se optó por utilizar PHP (Personal Hipertext Preprocessor), el cual arrojó mejores resultados.

2.8. Diseño de la base de datos

Se desarrolló un sistema, con la finalidad de controlar la información de usuarios, uso de recursos y control de reportes. Para así poder llevar una mejor administración de los recursos, todo esto por medio de una base de datos de tipo relacional.

2.8.1. Identificación de entidades

Las entidades son los objetos de interés principal o importante de un sistema, para el caso particular son:

- Usuarios
- Reportes
  - Supercomputadora Berenice8
  - Supercomputadora Berenice32
- Dependencias

2.8.2. Identificación de atributos de cada entidad

Los atributos son características o calificadores que proporcionan información detallada de una entidad. A continuación se citan los atributos para las entidades anteriormente mencionadas.

- USUARIOS

- Login Berenice8
- Login Berenice32
- Login Clusters
- Nombre
- Apellido Paterno
- Apellido Materno
- Dependencia
- Telefono
- Email
- Estatus

- REPORTES: CONSUMO DE UCRS LA MÁQUINA BERENICE8
  - Login8
  - Usuario (Nombre, Apellido Paterno, Apellido Materno)
  - Mes
  - Año
  - tcpu
  - Cpupm
  - Ucrcpup
  - Cpupnm
  - Ucrcpunp
  - Memp
  - Ucrmemp
  - Memnp
  - Ucrmemnp
  - Disco
  - Ucrdisco
  - Ucrtotal

- REPORTES: CONSUMO DE UCRS PARA LA MÁQUINA BERENICE32
  - Login
  - Usuario (Nombre, Apellido Paterno, Apellido Materno)
  - Nombre de la dependencia del usuario
  - Mes
  - Año

- Tcpu
- Cpupm
- Ucrcpup
- Cpupnm
- Ucrcpunp
- Memp
- Ucrmemp
- Memnp
- Ucrmemnp
- Disco
- Ucrdisco
- Ucrtotal

- REPORTES: CONSUMO DE UCRS PARA LA MÁQUINA CLUSTER
  - Login
  - Usuario (Nombre, Apellido Paterno, Apellido Materno)
  - Nombre de la dependencia del usuario
  - Mes
  - Año
  - Tcpu
  - Cpupm
  - Ucrcpup
  - Cpupnm
  - Ucrcpunp
  - Memp
  - Ucrmemp
  - Memnp
  - Ucrmemnp
  - Disco
  - Ucrdisco
  - Ucrtotal

- DEPENDENCIAS
  - Nombre de la dependencia
  - Clave de la dependencia

- Nombre del usuario

- ESTADO
  - Clave de estatus
  - Descripción de estatus

2.8.3. Relación de entidades

La relación representa asociaciones del mundo real entre entidades. Una relación esta asociado normalmente a un verbo o una preposición que establece la conexión entre dos entidades.

Las relaciones de entidades mayores se muestra la Figura 2.5.

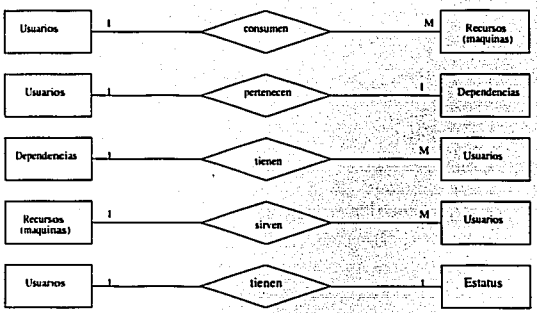


Figura 2.5: Relación de entidades.

2.8.4. Normalización

La normalización es una formalización de reglas aplicadas para asociar atributos con entidades. Los Beneficios de realizarla son:

- Mayor flexibilidad
- Asegurar que los atributos estén en las entidades apropiadas
- Disminuir la redundancia de datos
- Minimizar cambios estructurales al desarrollar nuevas aplicaciones
- Mantenimiento sencillo de la información

2.8.5. Primera forma normal

Un entidad cumple con la 1era. forma normal cuando se eliminan la repetición de grupos o dominios y columnas.

La Figura 2.6 contiene las entidades antes declaradas con los datos repetidos ya eliminados.

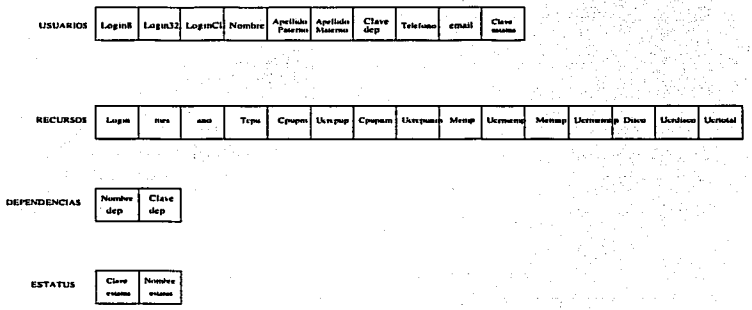


Figura 2.6: Primera forma normal.

2.8.6. Segunda forma normal

Para que se cumpla la segunda forma normal se necesita que:

- Cumple con la 1era. forma normal.
- Todos los atributos dependen de una llave primaria.

Esto obliga a que cada atributo tenga una *Dependencia Funcional* con la llave primaria. La Figura 2.7 muestra las llaves primarias y/o secundarias de cada una de las tablas, en ellas se pueden demostrar su dependencia ya que para conocer el consumo de recursos de un usuario forzosamente los datos se determinan por la entidad de recursos.

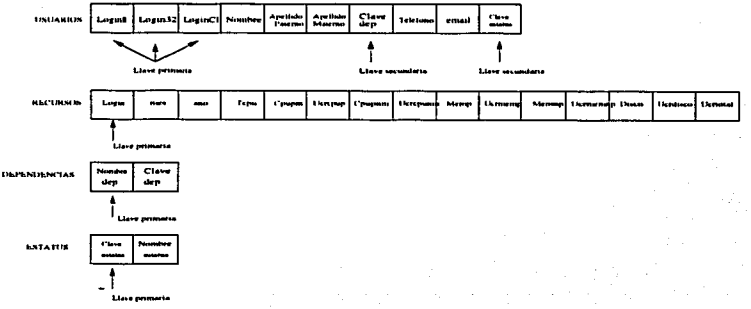


Figura 2.7: Segunda forma normal.

2.8.7. Tercera forma normal

Una entidad cumple con la tercera forma normal si:

- Cumple con la segunda forma normal.
- Ningún atributo de la entidad debe tener *Dependencia Transitiva* con la llave primaria.

La dependencia transitiva existe si **A** es funcionalmente dependiente de **B** y si **B** es funcionalmente dependiente de **C**, entonces **A** tiene una dependencia transitiva con **C**.

En este caso particular no existe una dependencia debido a que ningún atributo de una entidad depende de otro atributo que no sea la llave primaria.

En la Figura 2.8 se observa la comunicación de las tablas, las lineas negras representan las llaves primarias y las líneas punteadas son las llaves foráneas.

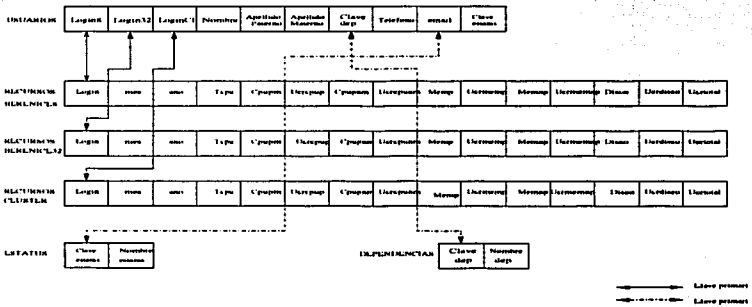


Figura 2.8: Tercera forma normal.

2.9. Pruebas

La estrategia de prueba de software empleada en la presente aplicación fue mixta, ya que para probar que el mismo satisficiese los requerimientos se empleo una técnica de arriba hacia abajo, es decir se probo que de manera general por cada uno de los módulos el sistema respondiese a lo especificado en los requerimientos, mientras que las pruebas tratando de causar un error en la aplicación se dieron de abajo hacia arriba, tomando cada función o dato en particular y extendiéndose hacia aspectos más generales. Por último las pruebas de desempeño se realizaron en componentes, de hardware, software, comunicaciones y personal.

Verificación de satisfacción de especificación de requerimientos.

Para ello se empleo una lista de chequeo, donde apareciesen los requerimientos determinados en la etapa de análisis y por otra tenemos al producto de software, en la misma se registraba si el requerimiento era cumplido y en caso de no hacerlos los motivos.

Pruebas tratando de causar un error en la aplicación.

Estas se desarrollaron tratando de introducir tipos de datos distintos a los correspondientes en las solicitudes de los mismos, tratando de bloquear la ejecución de la aplicación, desconectando dispositivos periféricos, como impresoras, nodos de red, etcétera, así como corriendo otras aplicaciones en los equipos cliente.

Pruebas de desempeño en componentes.

Las pruebas de desempeño del hardware se realizarón en el equipo propuesto, la aplicación presento un buen desempeño, por otra parte se pudo comprobar un óptimo comportamiento en la red de comunicaciones, respaldos y recuperación de la informacion.

Con los usuarios ser realizaron pruebas de las interfaces, así como de las salidas, para verificar que estas fuesen amigables con los mismos, sin complicaciones ó fuera de contexto, el resultado de esta prueba provocó correcciones menores en la interfaz gráfica.

2.10. Mantenimiento

Las actividades del mantenimiento integran una buena parte de la rutina en la puesta en marcha de un sistema, pues forman parte sustancial del buen funcionamiento de este y pueden repercutir en la productividad del mismo. En cambio un sistema bien diseñado puede requerir de pocas horas hombre en la protección del sistema.

Para este sistema en particular se han considerado las siguientes tareas tabla 2.10 que deben realizarse periodicamente.

Se creó un sistema cuyo diseño es comprensible y duradero que esta sirviendo para las necesidades actuales y las proyectadas, por lo cual se le contempla una vida útil. El sistema esta basado en módulos los cuales permiten que cualquier modificación al sistema no implica cambios sustanciales en el diseño total.

RECURSO	PROPÓSITO	POSIBILIDAD
Renovación del equipo de cómputo	Mayor rendimiento en los sistemas	Alta
Mejorar la integración de la red	Proveer niveles más confiafles de accesibilidad al servidor	Media
Actualización del servidor de impresión	Soportar el crecimiento y la demanda de peticiones	Baja
Nueva configuración clientes_ligeros -servidor_robusto	Mejor control de la información Facilidad en las labores de respaldo y recuperación	Alta
Uso de equipos con procesadores Intel	Proporcionar buen rendimiento a costos reducidos	Alta
Medios de almacenamiento digital de alta calidad.	Proporcionar niveles altos de confiabilidad	Alta
Medios de almacenamiento digital de calidad media.	Manipular información de forma sencilla y confiable a costos reducidos.	Alta
Uso de un sistema de impresión de alta resolución	Satisfacer las demandas de reportes al sistema de información.	Media
Uso de un impresión de resolución media.	Satisfacer las demandas de reportes al sistema de información.	Alta
Herramientas tipo OpenSource	No incurrir en gastos innecesarios en la compra de software y adquisición de licencias	Alta
Probar herramientas: • Linux • OpenBSD • Solaris	Proveer de funcionamiento y portabilidad en diversas plataformas	Alta
Uso de herramientas con Licencias de Software Libre	Garantizar el uso, estudio y distribución libre de las herramientas de software	Alta
Uso de licencias GFDL para documentación	Garantizar la libre distribución de la documentación generado sin perder los derechos de autor	Alta

Tabla 2.9: Tabla de evaluación de recursos.

Actividad	Período	Observación
Revisar procesos del sistema		
Verificar datos emitidos por el sistema		
Examinar la ejecución de las herramientas implicadas en el funcionamiento del sistema		
Evaluar el desempeño		
Verificar información - usuario		
Examinar los recursos físicos		
Detectar eventos anormales		

Tabla 2.10: Actividades de mantenimiento.

Capítulo 3

Liberación del sistema

3.1. Introducción

El sistema de control de reportes para atención de usuarios de las supercomputadoras fue creado para atender una necesidad latente del departamento de Supercómputo. Además de colaborar con las diferentes instituciones en la utilización de este recurso.

En la realización del sistema se efectuaron diferentes pruebas que consistian en el manejo de la información, siendo la prueba principal la interacción con los usuarios, a los cuales se les entregó varias versiones del programa para su evaluación, de donde se obtuvo una retroalimentación esencial en la realización del sistema.

En este capítulo se describe el funcionamiento del sistema, con la finalidad de ser un herramienta de apoyo para los administradores ó usuarios del mismo.

3.2. Instalación del software

Se necesitó instalar y configurar el sofware, para que este pudiese trabajar en conjunto.

A continuación se lista el software requerido:

- Msql
- PHP
- Apache

3.2.1. Instalación de Mini SQL

Comparado con otros manejadores de bases de datos, Mini SQL es seguro para configurar e instalar.

Proceso de instalación se describe en la siguiente lista:

1. Obtener la versión de mSQL del sitio *(www.hughes.com.au/products/msql/)*, para poder conseguirlo se debe llenar una forma de datos, indicando el uso que la aplicación tendrá.

2. Desempaquetar y descomprimir el código fuente del manejador de la base de datos.
   [root@localhost]# tar -xvzf msql-2.0.11.tarz

3. Introducirse al directorio.
   [root@localhost /root]# cd msql-2.0.11

4. Ejecutar la siguiente línea:
   [root@localhost msql-2.0.11]# make target
   Está crea un directorio con el nombre del sistema operativo, la versión del kernel y la arquitectura; en este existirá un archivo que contendra el software existente dentro del servidor.

```
Making target directory for Linux-2.4.2-2-i686

Building directory tree.
Adding common
Adding conf
Adding lang-common
Adding lite
Adding makedepend
Adding makegen
Adding msql
Adding regexp
Adding tests
Adding tests/rtest.src
Adding w3-msql
Adding w3-msql/tests

Adding sym-links
.....................................................................
.....................................................................
.................................
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking return type of signal handlers... void
```

```
...
...
...
checking for HP-UX ranlib. Nope, it's OK to use ranlib.
checking for Linux. Yup, it's Linux.  Adding -rdynamic to the link flags.
Also forcing msync for Linux.

Ready to build mSQL.


You may wish to check "site.mm" although the defaults should be
fine.  When you're ready, type  "make all" to build the software
```

5. Realizar la compilación y ejecución de los programas.
   [root@localhost]# make all

```
Regenerating Makefile.
.....................................................
Done.
make[1]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686'

Starting make for mSQL-2


--> [common] directory
make[2]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/common

Regenerating Makefile.
..................................
Done.
Done.
make[3]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/common
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
```

```
make[3]: Leaving directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/common'
make[2]: Leaving directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/common'
<-- [common] done

--> [regexp] directory
make[2]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/regexp

Regenerating Makefile.
.....................................
Done.
Done.
make[3]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2
-2-i686/regexp'
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
ar rc libregexp.a  regexp.o regsub.o
ranlib libregexp.a
rm -f ../lib/libregexp.a
ln -s ../regexp/libregexp.a ../lib/libregexp.a
...
...
...
gcc  -O -I../ -DHAVE_CONFIG_H -DHAVE_SSIZE_T -DHAVE_U_INT -DHAVE_BIT_TYPES
 -D_OS_UNIX  -DHAVE_DIRENT_H -DHAVE_DIRENT -DHAVE_MMAP -DMSYNC_3    -DINST_I
'"/usr/local/Hughes"' -DTARGET='"Linux-2.4.2-2-i686"' -DHAVE_RLIMIT_NOFILE
 acl.o -c ../msql/acl.c
```

6. Realizar la instalación del software.
[root@localhost]# make install

```
Starting install for mSQL-2

mkdir /usr/local/msql; chmod 0755 /usr/local/msql
mkdir /usr/local/msql/bin; chmod 0755 /usr/local/msql/bin
mkdir /usr/local/msql/include; chmod 0755 /usr/local/msql/include
mkdir /usr/local/msql/include/common; chmod 0755 /usr/local/msql/include/c
mkdir /usr/local/msql/lib; chmod 0755 /usr/local/msql/lib
mkdir /usr/local/msql/msqldb; chmod 0755 /usr/local/msql/msqldb
mkdir /usr/local/msql/msqldb/.tmp; chmod 0755 /usr/local/msql/msqldb/.tmp
mkdir /usr/local/msql/doc; chmod 0755 /usr/local/msql/doc
mkdir /usr/local/msql/www; chmod 0755 /usr/local/msql/www
mkdir /usr/local/msql/misc; chmod 0755 /usr/local/msql/misc
mkdir /usr/local/msql/makegen; chmod 0755 /usr/local/msql/makegen
```

```
mkdir /usr/local/msql/modules; chmod 0755 /usr/local/msql/modules

--> [common] directory
make[2]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/common'
make[3]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/common'
if test -f /usr/local/msql/include/common/portability.h;\
then\
rm -f /usr/local/msql/include/common/portability.h.old;\
mv /usr/local/msql/include/common/portability.h /usr/local/msql
/include/common/portability.h.old;\
fi;\
cp portability.h /usr/local/msql/include/common/portability.h
chmod 0755 /usr/local/msql/include/common/portability.h
if test -f /usr/local/msql/include/common/config.h;\
then\
rm -f /usr/local/msql/include/common/config.h.old;\
mv /usr/local/msql/include/common/config.h /usr/local/msql/
include/common/config.h.old;\
fi;\
make[2]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/lang-cc
make[3]: Entering directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/lang-cc
if test -f /usr/local/msql/lib/liblite.a;\
then\
rm -f /usr/local/msql/lib/liblite.a.old;\
mv /usr/local/msql/lib/liblite.a /usr/local/msql/lib/liblite.a.old;\
fi;\
...
...
...
cp mod_lite.o /usr/local/msql/lib/mod_lite.o
chmod 0755 /usr/local/msql/lib/mod_lite.o
make[3]: Leaving directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/lite'
make[2]: Leaving directory 'msql-2.0.11/targets/Linux-2.4.2-2-i686/lite'
<-- [lite] done

Installation of mSQL-2 complete.

*********
**   This is the commercial, production release of mSQL 2.0
**   Please see the README file in the top directory of the
**   distribution for license information.
*********
```

7. Verificar que la instalación se llevó a cabo adecuadamente, se procederá de la siguiente manera:

   - Invocar al demonio del mSQL.
     [root@localhost]#/usr/local/msql/bin/msql2d &
   - Ejecutar la instrucción que permite conocer los atributos de las bases de datos.
     [root@localhost]#/usr/local/msql/bin/relshow

     Debido a que todavía no se ha creado alguna de las bases de datos, sólo aparecerá lo siguiente:

     ```
     [root@localhost]# /usr/local/msql/bin/relshow

     +----------------------------+
     |          Databases         |
     +----------------------------+
     +----------------------------+
     ```

3.2.2. Instalación de Apache

Estos pasos permiten la instalación del servidor web.

1. Adquirir de la dirección *www.apache.ory* el código de apache.

2. Hacer cesar la compresión del archivo fuente.
   root@localhost.localdomain:#gunzip apache_1.3.19.tar.gz

3. Eliminar la agrupación del archivo fuente.
   root@localhost.localdomain:# tar -xvf apache_1.3.19.tar

4. Colocarse en el directorio.
   root@localhost.localdomain:# cd apache_1.3.19

5. Ejecutar el programa configure, que genera la información necesaria para que pueda ser instalado el software. Es importante notar que si el sistema no presenta algún sofware, este programa abortará y la instalación continuará hasta que se cubra esta necesidad.
   root@localhost.localdomain:# ./configure

```
Configuring for Apache, Version 1.3.19
 + Warning: Configuring Apache with default settings.
 + This is probably not what you really want.
..
..
..
..
Creating Makefile in src/modules/standard
```

6. Se compilan y ejecutan los programas necesarios que conforman el software.
root@localhost.localdomain:# make

7. Se instala el software.
root@localhost.localdomain:# make install

```
Making all in Zend
make[1]: Entering directory '/home/reyna/php-4.0.5/Zend'
/bin/sh ../libtool --silent --mode=compile gcc -DHAVE_CONFIG_H -I. -I.
-I../main   -DSUPPORT_UTF8 -DXML_BYTE_ORDER=12  -g -O2
-c zend_language_scanner.c
<=== [config]
make[1]: Leaving directory '/home/reyna/apache_1.3.19'
+--------------------------------------------------------------+
| You now have successfully built and installed the            |
| Apache 1.3 HTTP server. To verify that Apache actually       |
| works correctly you now should first check the               |
| (initially created or preserved) configuration files         |
|                                                              |
|   /usr/local/apache/conf/httpd.conf                          |
|                                                              |
| and then you should be able to immediately fire up           |
| Apache the first time by running:                            |
|                                                              |
|   /usr/local/apache/bin/apachectl start                      |
|                                                              |
| Thanks for using Apache.        The Apache Group             |
|                                 http://www.apache.org/       |
+--------------------------------------------------------------+
```

8. Se debe editar este archivo, para indicarle los valores adecuados tales como el nombre del servidor, el puerto, los módulos que se utilizarán, etc.
root@localhost.localdomain: # vi /usr/local/apache/conf/httpd.conf

9. Se ejecuta por primera vez el servidor web.
root@localhost.localdomain: # /usr/local/apache/bin/apachectl start
Si hubiera algún problema con la herramienta ó la configuracion de la misma, está no se ejecutará. El error puede ser o no ser impreso en la salidad estandar, por tal motivo se recomienda la verificar la ejecución del mismo.

3.2.3. Instalación de PHP

1. Ir a la dirección *www.php.net* y extraer la versión deseada de php.

2. Descomprimir el archivo.
root@localhost.localdomain:# gzip -d tar xvf php4.0.5.tar.gz

3. Desempaquetar el archivo.
root@localhost.localdomain:# tar xvf php4.0.5.tar

4. Cambiarse al directorio padre de la aplicación.
root@localhost reyna# cd php-4.0.5

5. Ejecutar el configure para que almacene el ambiente del sistema, asegurandonos que está aplicación funcione con mSQL y apache. De no habilitar esta línea no se generaran los módulos que permiten la cominicación entre las aplicaciones mencionadas.
root@localhost php-4.0.5]# ./configure –with-msql=/usr/local/msql –with-apache=../apache_1.3.1

6. Compilar y ejecutar los programas que conforman php.
root@localhost php-4.0.5]#make

7. Si el paso anterior fue exitoso, se procede a instalar el software.
root@localhost php-4.0.5]#make install

```
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
...
...
...
...
...
creating number.c
creating number.h
creating main/internal\_functions.c
+--------------------------------------------------------------------+
|                        *** WARNING ***                             |
|                                                                    |
| You chose to compile PHP with the built-in MySQL support.  If you  |
```

```
| are compiling a server module, and intend to use other server      |
| modules that also use MySQL (e.g, mod_auth_mysql, PHP 3.0,         |
| mod_perl) you must NOT rely on PHP's built-in MySQL support, and   |
| instead build it with your local MySQL support files, by adding    |
| --with-mysql=/path/to/mysql to your configure line.                |
+--------------------------------------------------------------------+
| License:                                                           |
| This software is subject to the PHP License, available in this     |
| distribution in the file LICENSE.  By continuing this installation |
| process, you are bound by the terms of this license agreement.     |
| If you do not agree with the terms of this license, you must abort |
| the installation process at this point.                            |
+--------------------------------------------------------------------+

Thank you for using PHP.
```

8. root@localhost.localdomain:#cp php.ini-dist /usr/local/lib/php.ini
   Este paso es opcional, ya que las versiones actuales de apache soportan los modulos de PHP

3.3. Puesta en marcha

Toca el turno a la instalación del sistema, pasos son los siguientes:

1. Ubicarse en el directorio destinado para el área de trabajo del sistema:
   - root@localhost php-4.0.5]# cd DIR_SRC
2. Descomprimir y desagrupar el código fuente:
   - root@localhost php-4.0.5]#tar xvfz scr.tar.gz
3. Ejecutar el programa que controla la base datos:
   - root@localhost php-4.0.5]#/usr/local/msql/bin/msql2d&
4. Iniciar el servicio del web:
   - root@localhost php-4.0.5]#/usr/local/apache/bin/apachectl start
5. Consideraciones importantes:
   - Verificar los permisos de los archivos que conforman el sistema, los deben ser lectura,escritura y ejecución para el dueño, lectura, escritura y ejecución para el grupo y lectura y ejecución para otros.

```
[reyna@cafe bweb]$ ls -l
total 3204
drwxrwxr-x    9 reyna    bweb          4096 Sep 30 12:52 base
```

- En el archivo de configuración de apache, en la sección de virtual host se especifica: *dirección IP de la máquina, correo electrónico del administrador, ruta donde se encuentren los programas y el nombre de la página.*

```
<VirtualHost 132.248.159.25>
    ServerAdmin reyna@super.unam.mx
    DocumentRoot /web/bweb
    ServerName  cafe.super.unam.mx
</VirtualHost>
```

- Asegurarse que la comunicación entre el servidor web y el manejador de la base de datos se encuentre activa.

```
[reyna@cafe bweb]$ ps -fea |egrep "msql2d|httpd"
root       8260     1  0 Sep11 ?        00:00:05 /usr/local/Hughes/bin/
root      23648     1  0 Sep25 ?        00:00:00 /usr/sbin/httpd -DHAVE
apache     1371     1  0 Sep27 ?        00:00:00 httpd
apache    31378 23648  0 Sep29 ?        00:00:00 /usr/sbin/httpd -DHAVE
apache    31379 23648  0 Sep29 ?        00:00:00 /usr/sbin/httpd -DHAVE
apache    31380 23648  0 Sep29 ?        00:00:00 /usr/sbin/httpd -DHAVE
apache    31383 23648  0 Sep29 ?        00:00:00 /usr/sbin/httpd -DHAVE
apache    31384 23648  0 Sep29 ?        00:00:00 /usr/sbin/httpd -DHAVE
apache    31385 23648  0 Sep29 ?        00:00:00 /usr/sbin/httpd -DHAVE
apache    31386 23648  0 Sep29 ?        00:00:00 /usr/sbin/httpd -DHAVE
apache    31387 23648  0 Sep29 ?        00:00:00 /usr/sbin/httpd -DHAVE
```

3.3.1. Estructura del sistema

El sistema esta integrado por los siguientes archivos ó directorios:

- **base** Directorio principal.
  - style.css
  - top.html
  - principal.html
  - menú.html
  - index.php
  - contenido.php

- usu_todos
  - index.php
  - menú.html
  - todos_usuarios.php
- usu_alg
  - index.php
  - menú.html
  - consultas_parcial.php
- usu_bus
  - index.php
  - menú.html
  - busqueda_usuarios.php
- dep_todas
  - index.php
  - menú.html
  - dep_total.php
- dep_int
  - index.php
  - menú.html
  - dep_int.php
- dep_ext
  - index.php
  - menú.html
  - dep_ext.php
- uso_mensual8
  - index.php
  - menú.html
  - mensual_b8.php
- uso_mensual32
  - index.php
  - menú.html
  - mensual_b32.php
- uso_semestral
  - index.php
  - menú.html
  - semestral.php

3.4. Acceso al sistema

Para poder utilizar el sistema, se necesita; accesar a través de un navegador a la dirección *http://cafe.super.unam.mx/base*, esto bastará para ingresar al sistema, siempre y cuando la máquina de donde se esté realizando la petición está permitida en el sistema. Para este sistema las direcciones que pueden accesar pertenecen al dominio *super.unam.mx* así como algunas direcciones de la Dirección General y la Dirección de cómputo para la investigación de la DGSCA. Para estos casos donde el acceso fue permitido aparecerá en el browser un menú (Figura 3.1) la cual es pantalla principal del sistema; en caso contrario se presentara una ventana mostrandonos el acceso denegado Figura 3.2.

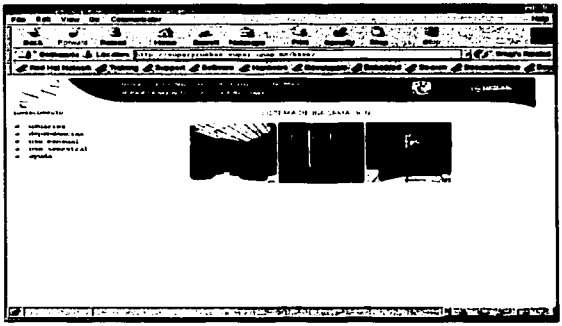


Figura 3.1: Inicio del sistema.

3.5. Realizar consultas

Una vez que se ingresó al sistema se podrá elegir entre diferentes tipos de consultas, estas son: *usuarios, dependencias, uso mensual, uso semestral y la ayuda*, para poder conocer a detalle información de quienes utilizan el equipo y como está siendo utilizado.

3.5.1. Menú usuarios

Cuando se hace referencia a la liga *usuarios* se presentará, un submenú donde se elegirá a detalle el tipo de consulta que se desea Figura 3.3, los opciones que presenta

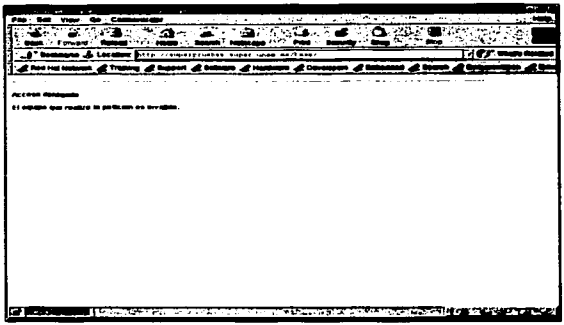


Figura 3.2: Acceso denegado al sistema.

este menú son:

- *Todos los usuarios*
  En esta opción se podrá elegir a los usuarios en general, es decir que no necesariamente estén en todas las máquinas, basta con que se encuentren en cualquiera de ellas. Esta consulta no es excluyente.

- *Algunos usuarios*
  Esta opción muestra a los usuarios de una forma excluyente, donde los usuarios listados deben pertenecer a todas las máquinas elegidas en la consulta.

- *Búsqueda*
  Es una búsqueda específica de cualquier usuario a través de su nombre o el login asignado.

Submenú todos los usuarios

Si se elegió todos los usuarios, aparacerá un formato (Figura 3.4) donde se especifican los campos que se desean incluir.

Los campos mínimos a ingresar para que se realize la consulta son:

- Berenice8
- Berenice32
- Cluster

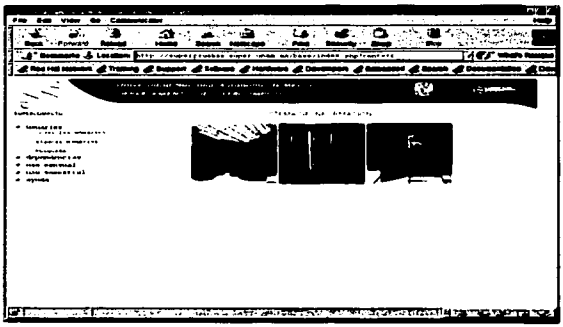


Figura 3.3: Menú usuarios.

- Nombre

En caso de no especificar algún campo, aparecerá la Figura 3.5 indicando que no se pudo realizar la consulta.

Los campos que se pueden excluir para una consulta son:

- Teléfono
- Email
- Dependencia
- Estatus de la cuenta

Submenú algunos usuarios

La Figura 3.6 muestra un formato similar al anterior donde se deben selecionar los campos que desean incluir en la consulta.

Esta consulta se puede realizar con la inclusión de cualquiera de los siguientes campos:

- Berenice8
- Berenice32
- Cluster

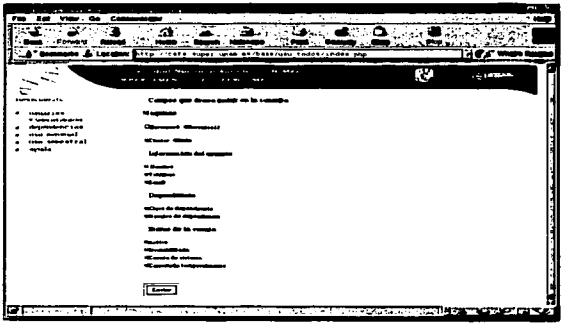


Figura 3.4: Submenú todos los usuarios.

- Nombre

Campos opcionales:

- Teléfono
- Email
- Dependencia
- Estatus de la cuenta

Submenú búsqueda
En este menú (Figura 3.7) se indica si la consulta será por nombre o login, ingresando la palabra que se desea encontrar.

3.5.2. Menú dependencias

En este menú (Figura 3.8) pueden ser mostradas todas las dependencias, dependencias internas o externas.

En los submenús referidos a dependencias (Figura 3.9, Figura 3.10 y Figura 3.11) se puede elegir el formato de la consulta deseada, por clave, nombre o ambas.

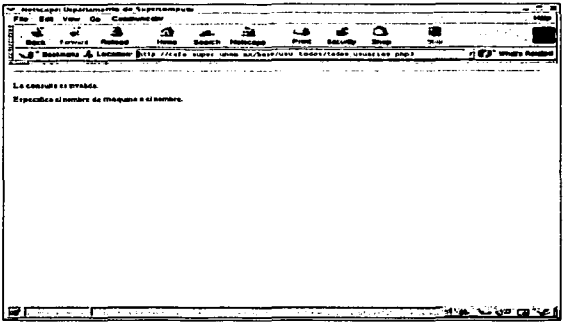


Figura 3.5: Consulta invalida de submenú todos los usuarios.

3.5.3. Menú uso mensual

En este menú de uso mensual podemos obtener información del uso de los equipos, berenice8 y berenice32 como se muestra en la Figura 3.12

En los submenús referidos al uso mensual (Figura 3.13 y Figura 3.14) se puede elegir el tipo de reporte a través de usuario o dependencia donde pueden ser consultados cualquier mes de cualquier año de los cinco años que conforman la base de datos.

3.5.4. Uso semestral

En este menú se pueden consultar el uso del equipo en un intervalo de seis meses y por máquina, como se muestra en la Figura 3.15

3.5.5. Ayuda

Esta opción nos permitirá conocer la forma de utilizar los distintos menús y submenús con los que cuenta este sistema.

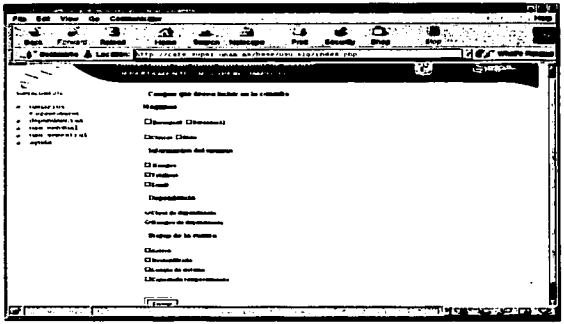


Figura 3.6: Submenú algunos usuarios.

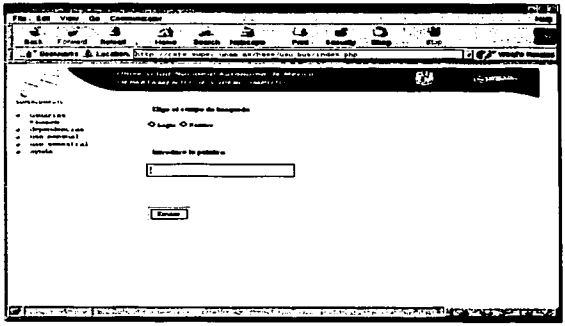


Figura 3.7: Submenú búsqueda.

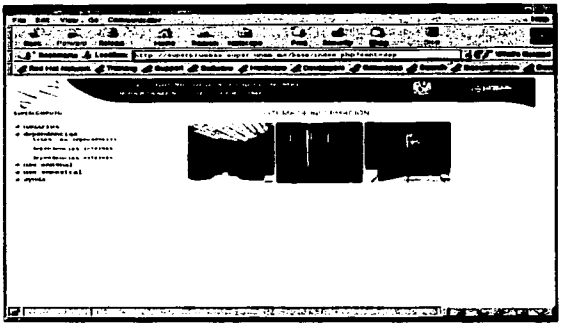


Figura 3.8: Menú dependencias.

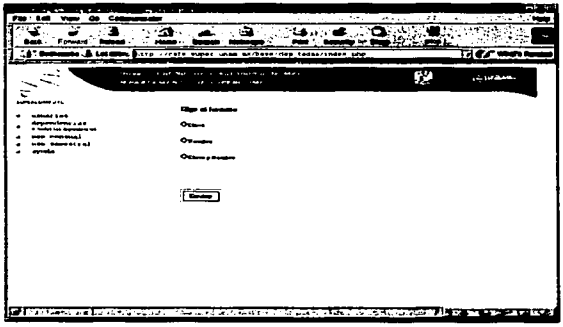


Figura 3.9: Submenú todas las dependencias.

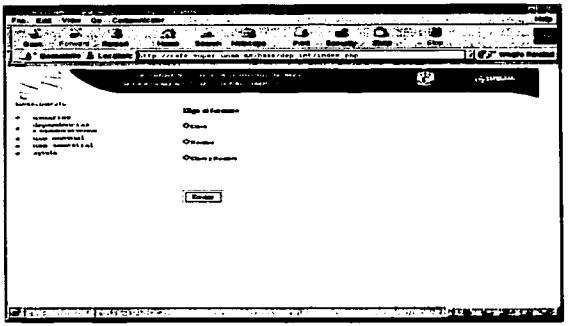


Figura 3.10: Submenú dependencias internas.

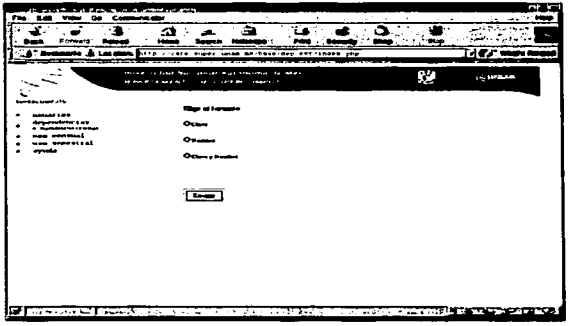


Figura 3.11: Submenú dependencias externas.

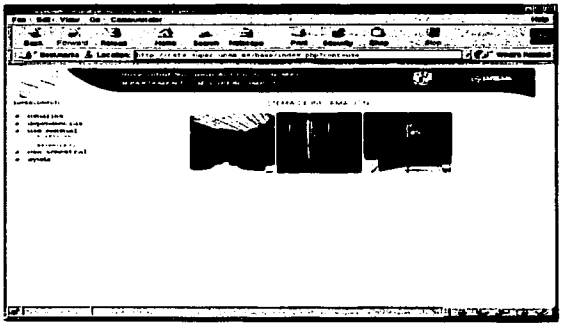


Figura 3.12: Menú uso mensual.

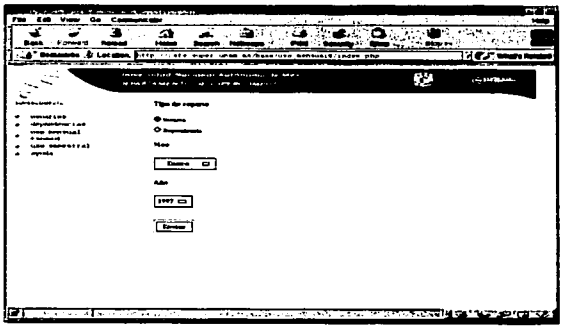


Figura 3.13: Submenú uso mensual berenice8.

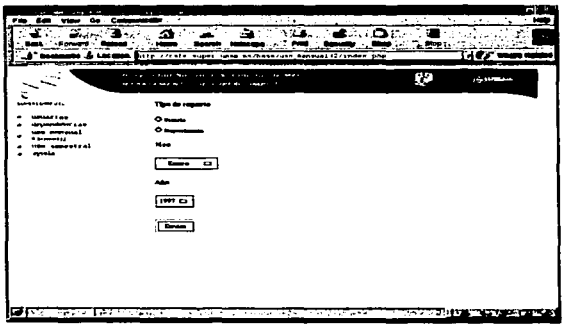


Figura 3.14: Submenú uso mensual berenice32.

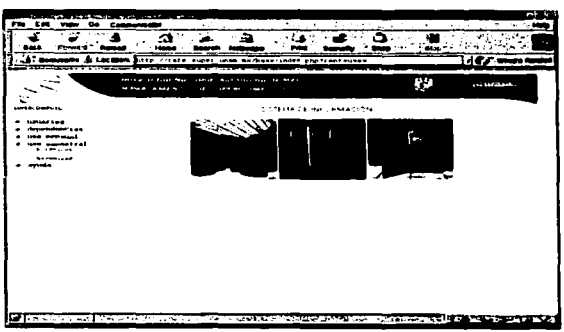


Figura 3.15: Menú uso semestral.

Conclusiones

En este trabajo se presentó un sistema de información, aplicado a supercomputadoras, con el fin de mejorar la administración de los recursos. En dicho sistema, se emplearón diversas herramientas que han sido evaluadas, instaladas, configuras y programadas, para poder trabajar en conjunto y lograr un óptimo funcionamiento, cubriendo las necesidades esenciales.

Durante el desarrollo del sistema se realizó un estudio de factibilidad técnica y económica, con estos paráinetros se estableció la utilización del software (*Msql, PHP, HTMIL*), de la cual se debe mencionar que es de dominio público y el uso del mismo no requiere pago alguno. Se ópto por usar una base de datos de tipo relacional, ya que me permite clasificar datos mediante tablas, que se comunican por medio de un dato común que pertenence a las tablas, denominado llave, este selecionará la información requerida; para lograr esta interacción se elegió utilizar el manejador de base de datos *mSQL*, ya que cuenta con las instrucciones básicas, que permiten la obtención de datos en las diferentes tablas, mediante las llaves asignadas. También se logro la integración de herramientas que permitieron consultas interactivas, mediante un canal que ofrece facilidad, accesibilidad y portabilidad en la realización de las consultas, en base a este criterio se decidió que estas consultas fueran a través de un navegador. Para obtener esta manipulación se programó con *PHP* que se encarga de la comunicación entre el manejador *mSQL* y lenguaje en el cual se hacen las páginas web *HTMIL*.

Una vez concluido el desarrollo del sistema, se comprobó que cuando se manejan grandes volumenes de información, el uso de la metodología de base de datos relacionales, presenta robustez, consistencia y estabilidad en la interacción de los datos.

Es importante señalar, que se pueden generar diferentes reportes, que se utilizan para construir datos estadistícos que informan a la comunidad científica como estan siendo utilizados los equipos de supercómputo, así como justificar la demanda y necesidad del cómputo de alto rendimiento.

Cabe mencionar que este sistema sienta las bases para un futuro crecimiento de esta herramienta, donde se contempla la implementación de un módulo que reporte los datos estadísticos de forma gráfica.

Apéndice A

Manual de MiniSQL

A.1. Introducción

El lenguaje MSQL ofrece un importante subconjunto de características proporsionadas por ANSI SQL. Esto permite a un usuario o programador almacenar, manipular y recuperar datos en tablas de estructuras. Sin embargo, no soporta algunas características relacionales, tales como vistas o consultas anidadas. Pero a pesar de que no soporta todas las operaciones relacionales definidas en las especificaciones ANSI, provee la característica de unir (joins) múltiples tablas.

La definiciones y ejemplos siguientes muestran las palabras reservadas de MSQL en mayúsculas, pero esto no es una restricción en las consultas actuales.

LA SENTENCIA CREATE

La sentencia Create que era soportada en MSQL 2 puede utilizarse para crear tablas, índices y secuencias. No puede utilizarse para crear otras definiciones como las vistas. La sintaxis correcta de la sentencia Create se muestra a continuación:

```
CREATE TABLE nombre_tabla(
nombre_col tipo_dato [ not null ]
[,nombre_col tipo_dato [ not null ] ])

CREATE [ UNIQUE ] INDEX nombre_indice ON nombre_tabla(
nombre_campo
[,nombre_campo]**)

CREATE SEQUENCE ON nombre_tabla [STEP step_val][VALUE val_inicial]
```

Un ejemplo de creación de tabla se muestra a continuación:

```
CREATE TABLE emp_detalles(
Nombre char(15) not null,
Apellido char(15) not null,
```

```
Comentario text(50),
Departamento char(20),
emp_id int)
```

Los tipos disponibles son:

Tipo de dato	Descripción
char (len)	Cadena de caracteres (u otros 8 bits de datos).
text (len)	Variable de longitud de cadena de caracteres (u otros 8 bits de datos). La longitud definida se utiliza para indicar la longitud promedio del dato. Cualquier dato más grande que la longitud especificada será dividido entre la tabla de datos y los buffers de sobreflujo externos. Nota: Los campos de texto son de acceso más lento que los campos char y no se pueden utilizar en un índice o en sentencias LIKE.
int	Valores enteros con signo.
real	Valores decimales o valores reales de notación científica.

La estructura de la tabla mostrada en el ejemplo es ideal para la creación de algunos índices. Se supone que el camo emp_id es un valor único que se usa para identificar a un empleado. Un campo como este debe ser definido como llave primaria. MSQL 2.0 ha eliminado el soporte a creación de llaves primarias dentro de la sintaxis de creación de la tabla, a menos que el mismo resultado pueda ser considerado como un índice. De forma similar, una consulta común puede utilizarse para acceder a un empleado basándose en la combinación de nombre y apellido. Un índice compuesto (por ejemplo, construido de más de un campo), daría más rendimiento. Pueden construirse dichos índices usando:

```
CREATE UNIQUE INDEX idx1 ON emp_details (emp_id)
CREATE INDEX idx2 ON emp_details (Nombre, Apellido)
```

Estos índices serán utilizados de forma automática, siempre que una consulta sea enviada al motor de base de datos que utilice esos campos en su cláusula WHERE. El usuario no es requerido para especificar cualquier valor especial en la consulta para asegurar que el índice se utilice para incrementar el rendimiento.

Las Secuencias proveen un mecanismo a través del cual un valor de secuencia puede ser mantenido por el servidor MSQL. Esto permite realizar operaciones atómicas (tales como obtener el siguiente valor de secuencia) y elimina los conceptos asociados con el rendimiento al ejecutar estas operaciones en las aplicaciones del cliente. Una secuencia se asocia con una tabla y una tabla puede contener más de una secuencia.

Una vez que ha sido creada una secuencia, esta puede ser accedida utilizando la sentencia SELECT para obtener la variable de sistema _seq de la tabla, en la cual la secuencia está definida, por ejemplo:

```
CREATE SEQUENCE ON TABLE test STEP 1 VALUE 5
SELECT _seq FROM test
```

La operación CREATE indicada arriba definirá una secuencia en la tabla llamada test que tiene un valor inicial de 5 y que será incrementado en cada vez que es accedido (con un incremento de 1 en 1). La sentencia SELECT retornará un valor de 5. Si el SELECT se ejecuta de nuevo, retornara un valor de 6. Cada vez que se llama al valor de _seq se retorna su valor actual y se incrementa en uno.

Usando las opciones STEP y VALUE puede crearse una secuencia que inicia en un número específico y se incrementa o decrementa por un valor especificado. El valor de la secuencia decrementará en 5 cada vez que sea accedido si se define un STEP 5.

A.2. La sentencia Drop

La sentencia Drop se utiliza para eliminar una definición de la base de datos. Es comúnmente utilizada para eliminar una tabla de una base de datos pero también puede usarse para remover algunos constructores. En 2.0 puede utilizarse para eliminar definiciones de índice, secuencias o tablas. Es importante resaltar que al eliminar una tabla o un índice se eliminan los datos asociados con el objeto junto con la definición.

La sintaxis de la sentencia DROP y su uso se muestran a continuación.

```
DROP TABLE nombre_tabla
DROP INDEX nombre_indice FROM nombre_tabla
DROP SEQUENCE FROM nombre_tabla
```

Ejemplo:

```
DROP TABLE emp_detalles
DROP INDEX idx1 FROM emp_detalles
DROP SEQUENCE FROM emp_detalles
```

A.3. La cláusula Insert

A diferencia del ANSI SQL, no se puede anidar un SELECT dentro de un INSERT (no se puede insertar un dato retornado por un select). Si no se especifica el nombre de los campos, estos se utilizarán en el orden en que fueron definidos, se debe especificar un valor para cada campo si se hace de esta forma.

```
INSERT INTO nombre_tabla columna columna VALUES (value [, value])
```

Ejemplo:

```
INSERT INTO emp_detalles (Nombre, Apellido, Depto, Salario) VALUES ('David', 'H
INSERT INTO emp_detalles VALUES ('David', 'Hughes','Development','12345')
```

El número de valores proporcionados debe coincidir con el número de columnas.

A.4. La cláusula SELECT

El SELECT incluido en MSQL no tiene algunas de las características del SQL estándar. El desarrollo de MSQL 2 continúa, por eso algunas de las funciones faltantes estarán disponibles en la siguiente versión beta. Por el momento el SELECT de MSQL no incluye:

- Selects anidados
- Funciones implícitas (como count(), avg(), etc.)

Sin embargo, ya soporta lo siguiente:

- Joins - incluyendo alias de tablas
- DISTINCT para la seleccion de renglones
- ORDER BY
- Coincidencia de expresiones regulares
- Comparaciones Columna a Columna en cláusulas WHERE
- Condiciones complejas

La definición formal de la sintaxis de la cláusula SELECT de MSQL es:

```
SELECT [tabla.]columna [,[tabla.]columna ]**
       FROM tabla[=alias][,tabla[=alias] ]**
       [WHERE[tabla.]columna OPERATOR VALUE
              [AND|OR [tabla.]columna OPERATOR VALUE]**]
       [ ORDER BY [tabla.]column [DESC] [, [tabla.]column [DESC] ]
```

Los operadores pueden ser <, >, =, <=, >=, <>, LIKE, RLIKE o CLIKE VALUE que puede ser un valor literal o un nombre de columna. Las sentencias WHERE pueden contener las siguientes condiciones anidadas: where (edad < 20 or edad > 30) and sexo Masculino

Un SELECT simple puede ser:

```
SELECT Nombre, Apellido FROM emp_detalles
   WHERE depto = 'finanzas'
```

Para ordenar los datos en orden ascendente por Apellido y descendente por Nombre se vería de la siguiente forma:

```
SELECT Nombre, Apellido FROM emp_detalles
   WHERE depto = 'finanzas'
   ORDER BY Apellido, Nombre DESC
```

Para eliminar renglones duplicados del resultado del select se puede usar el operador DISTINCT

```
SELECT DISTINCT Nombre, Apellido FROM emp_detalles
    WHERE depto = 'finanzas'
    ORDER BY Apellido, Nombre DESC
```

MSQL incluye tres operadores de expresión regulares para utilizarse en comparaciones WHERE. La sintaxis estándar de SQL provee características de expresiones regulares muy simples que no da la flexibilidad o el poder al que los programadores de UNIX están acostumbrados. MSQL soporta la sintaxis de las expresiones regulares de SQL a través del operador LIKE, pero tambien ofrece más funcionalidad si es requerida. Los operadores de expresiones regulares son:

- LIKE - El operador de expresión regular estándar de SQL.
- CLIKE - Un operador LIKE estándarque ignora los Case
- RLIKE - operador de expresión regular completo para UNIX.

Nota: CLIKE y RLIKE no son estándar de SQL y pueden no estar disponibles en otras implementaciones del lenguaje al portar la aplicación. Sin embargo, son características muy poderosas y convenientes para MSQL.

La sintaxis de las expresiones regulares soportadas por los operadores LIKE y CLIKE es la misma que la del SQL estándar y se mencionan en seguida:

- '_' Coincide con algún caracter simple
- '' coincide con 0 o más caracteres de cualquier valor
- '\' Ignora caracteres especiales (por ejemplo, '\' coincide, y '\\' coincide con otros caracteres y consigo mismo)

Como ejemplo del operador LIKE, es posible buscar a alguien del departamento de finanzas cuyo apellido comience con cualquier letra seguido por 'ughes', como Hughes. La consulta para realizar esta operación quedaría como sigue:

```
SELECT Nombre, Apellido FROM emp_detalles
    WHERE depto = 'finanzas' and Apellido like '_ughes'
```

El operador RLIKE da acceso al poder de las expresiones regulares de UNIX. La sintaxis de expresiones regulares de UNIX provee una mayor funcionalidad que la sintaxis LIKE de SQL. La sintaxis de UNIX no usa el ' - ' o'' de la forma que SQL lo hace (como se menciono arriba). La sintaxis del operador RLIKE es:

- '.' Coincide con cualquier caracter simple
- '^' Cuando se usa como el primer caracter en una expresión regular, forza a que la comparación comience con el primer caracter de la cadena.
- '$' Cuando se usa como último caracter, el signo de moneda forza a la comparación a comenzar con el último caracter de la cadena
- '[ ]' Agrupando un grupo de caracteres simples dentro de corchetes cuadrados, la expresión regular hará coincidir un caracter simple con un grupo de caracteres. si el caracter ']' es uno de los caracteres que se desea que coincida, se debe especificar como el primer caracter en un grupo sin cerrar el grupo (por ejemplo '[]abc]', hará coincidir cualquier caracter simple que coincida con ']', 'a', 'b', o 'c'). Los rangos de caracteres pueden ser especificados dentro del grupo usando la sintaxis 'primero-último' (por ejemplo '[a-zO-9]' haría coincidir cualquier letra minúscula o un dígito). Si el primer caracter de un grupo es el caracter 'A', el regex hará coincidir cualquier caracter simple que no esté contenido en el grupo.
- '*' Si algún elemento de la expresión regular es seguido por un '*', coincidirá con cero o más instancias de la expresión regular.

El poder de un lenguaje de consulta relacional comienza a ser visible cuando se efectúan uniones entre tablas durante una operación select. Digamos que se tienen 2 tablas definidas, una conteniendo los detalles del staff y otra listando los proyectos que serán llevados a cabo por cada miembro del staff, y a cada miembro del staff se le ha asignado un número que es único para esa persona. Se puede generar una lista ordenada de quién está trabajando en qué proyecto con una consulta como:

```
SELECT emp_detalles.Nombre, emp_detalles.Apellido,
proyecto_detalles.proyecto
FROM emp_detalles, proyecto_detalles
WHERE emp_detalles.emp-id = proyecto_detalles.emp-id
ORDER BY emp_detalles.Apellido, emp_detalles.Nombre
```

MSQL no pone restricción en cuanto al numero de tablas unidas (joined) durante una consulta, asi que si fueran 15 tablas, todas conteniendo información relativa a una clave de empleado de alguna manera, los datos de cada una de esas tablas pueden ser obtenidos por una simple consulta. Algo a destacar al hacer uniones es que se debe hacer referencia al nombre de la tabla en la llamada de cada columna. MSQL no soporta el concepto de columnas con nombre unico en multiples tablas, asi que esto obliga a hacer mencion de la tabla de la cual se esta obteniendo la columna en un SELECT simple.

MSQL tambien soporta alias de tablas, asi se pueden efectuar uniones de tablas en si mismas. Esto puede parecer algo carente de utilidad, pero es una característica muy poderosa si hay renglones en una tabla simple relacionados con otros en alguna

forma. Un ejemplo de una tabla así pudiera ser una persona incluyendo los nombres de sus padres. En tal tabla habría múltiples renglones con una relacion padre/hijo. Usando un alias de tabla se puede averiguar cualquier padre que este contenido en la tabla usando algo como esto:

```
SELECT t1.padre, t2.hijo from padre-datos--t1,
padre-datos--t2 where t1.hijo = t2.padre
```

Los alias de tabla t1 y t2 apuntan ambos a la misma tabla (padre-datos en este caso) y son tratados como si fueran dos diferentes tablas que simplemente contienen los mismos datos.

A.5. La sentencia Delete

La sentencia SQL Delete se utiliza para eliminar una o mas entradas de una tabla de una base de datos. La selección de los renglones a ser eliminados de la tabla se basa en la misma forma en la que fue construida al usar la sentencia SELECT. La sintaxis para la sentencia Delete es:

```
DELETE FROM nombre_tabla
WHERE columna OPERATOR valor
AND 1 OR columna OPERATOR valor ]**
```

El operador puede ser <, >, <>, LIKE, RLIKE, o CLIKE.
Ejemplo:

```
DELETE FROM emp_detalles WHERE emp_id = 12345
```

A.6. La sentencia Update

La sentencia SQL Update se usa para modificar los datos que estan ya en la base de datos. La operacion puede ejecutarse en varios renglones definidos con la sentencia WHERE. El valor de cualquier numero de campos en los renglones que coinciden con lo especificado en el WHERE pueden ser actualizados. MSQL limita una operacion en la clausula update, no se puede usar un nombre de columna como un valor a actualizar (por ejemplo: no se puede establecer el valor de un campo al valor actual de otro campo). Solo los valores literales pueden ser utilizados como valor de actualizacion. La sintaxis soportada por MSQL es:

```
UPDATE nombre_tabla SET columna=valor[,columna=valor]**
WHERE columna OPERATOR valor
[ AND 1 OR columna OPERATOR valor]**
```

El operador puede ser <,>,<>, LIKE, RLIKE o CLIKE.
Ejemplo:

```
UPDATE emp_detalles SET salario = 30 WHERE emp_id = 1234
```


A.7. Programación de la API de Mini SQL 2.0

A.7.1. Introducción

Incluida en la distribución, se encuentra la librería API de MSQL, libmsql.a. La API permite a cualquier programa en C comunicarse con el motor de base de datos. Las funciones API quedan disponibles al incluir el archivo de cabecera msql.h en el programa y enlazando la librería MSQL (utilizando -lmsql como argumento en el compilador de C). La librería y el archivo de cabecera serán instalados en forma predeterminada dentro de /usr/locallHughes/lib y usr/local/Hughes/include respectivamente.
Como el motor MSQL, la API soporta depuración a través de la variable de ambiente MSQL-DEBUG. Tres modulos de depuración son actualmente soportados por la API: query, api y malloc. Habilitar la depuración "query"hará que la API muestre el contenido de las consultas a medida que son enviadas al servidor. El módulo de depuración .ªpi"hace que la información interna, tal como los detalles de conexión sean mostrados. Los detalles sobre la memoria utilizada por la librería API pueden obtenerse a través del módulo de depuración "malloc". La información tal como la ubicación y el tamaño de los bloques de memoria y sus direcciones será generada. Múltiples modulos de depuración pueden habilitarse estableciendo MSQL-DEBUG como una lista de nombres de módulo separada por dos puntos. Por ejemplo setenv MSQL-DEBUG api:query

A.8. Funciones relativas a consultas

A.8.1. msqlConnect()

```
int msqlConnect ( host )
char * host;
```

msqlConnect() ejecuta una interconexión con el motor MSQL. Toma como único argumento el nombre de la dirección IP del servidor en el que corre MSQL server. Si se especifica un NULL como argumento del host, la conexión se establece con el servidor que corre en la máquina local utilizando el socket de dominio de UNIX /dev/msqld. Si ocurre un error, se retorna un valor de -1 y la variable externa msqlErrMsg contendrá el mensaje de error correspondiente. Esta variable está definida en "msql.h".

Si la conexión se realiza con un servidor, un identificador entero es retornado a la función que la llama. Este valor se utiliza como manejador para todas las demas llamadas a la API de MSQL. La variable retornada es de hecho el socket descriptor para la conexión. Al llamar a msqlconnecto mas de una vez y asignar las variables retornadas a variables separadas, pueden mantenerse múltiples conexiones simultaneas a servidores de base de datos.

En versiones anteriores de MSQL, la variable de ambiente MSQ@HOST podía utilizarse para especificar una máquina destino si el parámetro del host era NULL. Esto ya no funciona en nuevas versiones.

A.8.2. msqlSelectDB()

```
    int msqlSelectDB ( sock , dbName)
int sock ;
char * dbName;
```

Antes de realizar cualquier consulta, una base de datos debe ser seleccionada. msqlSelectDB() le indica al motor que base de datos será accedida. msqlSelectDB() es llamado con el socket descriptor retornado por msqlConnect y el nombre de la base de datos deseada. Un valor de regreso -1 indica un error con msqlErrMsg establecido en una cadena que describe el error. msqlSelectDB() puede ser llamado múltiples veces durante la ejecucuíon de un programa. Cada vez que se llama, el servidor utilizará la base de datos especificada para accesos futuros. Al llamar a msqlSelectDB() múltiples veces, un programa puede alternar entre diferentes bases de datos durante su ejecución.

A.8.3. msqlQuery()

```
    int msqlq uery ( sock , query)
int sock ;
char * query;
```

Las consultas son enviadas al motor bajo la conexión asociada a un socket como cadenas de texto plano utilizando msqlQuery(). Como en comandos previos de MSQL, un valor retornado de -1 indica un error y msqlErrMsg será actualizada conteniendo el mensaje de error correspondiente. Si la consulta genera una salida para el motor, tal como una sentencia SELECT el dato es almacenado en la API esperando que la aplicación lo recupere. Si la aplicación envía otra consulta antes de recuperar el dato unsando msqlStoreResult(), el buffer se sobreescribirá por cualquier dato generado en la nueva consulta.

En versiones anteriores de MSQL, el valor de retorno de msqlQuery() era un -1 (indicando un error) o 0 (indicando éxito). MSQL 2 contiene agregados a esta semántica al incorporar más información de retorno para la aplicación mediante el código de retorno. Si el código de retorno es mayor a 0, no indica únicamente éxito, también indica el número de renglones afectados por la consulta (por ejemplo, el

número de renglones retornados por un SELECT, el número de datos modificados por un UPDATE, o el número de renglones eliminados por un DELETE).

A.8.4. msqlStoreResult()

```
m_result * msqlStoreResult()
```

Los datos regresados por una consulta SELECT deben ser almacenados antes de que otra consulta sea enviada o serán eliminados del buffer de la API. Los datos se almacenan utilizando la función msqlStoreResul(), la cual retorna un resultado a las rutinas que la llamaron. El resultado es un puntero a una estructura m-result y es pasado a otras rutinas API cuando se equiere acceso a los datos. Una vez que el manejaor de resultados es almacenado, otras consultas pueden ser enviadas. Un programa puede tener muchos manejadores de resultados activos simultáneamente.

A.8.5. msqlFreeResult()

```
void msqlFreeResult (result)
m_result * result;
```

Cuando un programa ya no requiere los datos asociados a un resultado de una consulta en particular, los datos deben ser liberados usando la función msqlFreeResult(). El resultado asociado con los datos, tal como fueron retornados por msqlStoreResult() se pasa a msqlFreeResult() para identificar el conjunto de datos a ser liberados.

A.8.6. msqlFetchRow()

```
m_row msqlFetchRow ( result )
m_result * result;
```

Los renglones individuales de la base de datos regresados por un SELECT son accedidos vía la función msqlFetchRow(). Los datos son retornados en una variable de tipo m_row la cual contiene un puntero char para cada campo en el renglón. Por ejemplo, si una sentencia SELECT selecciona 3 campos de cada renglón retornado, el valor de los 3 campos debe ser asignado a los elementos [0], [1], y [2] de la variable retornada por msqlFetchRow(). Un valor NULL es retornado cuando el fin de la base de datos ha sido alcanzado. Ver el ejemplo al final de estas secciones para mayores detalles. Note que un valor NULL es representado como un puntero NULL en el renglón.

A.8.7. msqlDataSeek()

```
void msqlDataSeek ( result , pos )
m_result * result;
int pos;
```

La estructura m_result contiene un çursor"del lado del cliente que almacena información sobre el siguiente renglón de datos que serán retornados al programa que lo llamó. msqlDataSeek() puede utilizarse para moverse a la posición del cursor de datos. Si es llamado con una posición de 0, la siguiente llamada a msqlFetchRow() retornará el primer renglón de datos retornados por el servidor. El valor de pos puede ser cualquiera, desde 0 (el primer renglón) hasta t número de renglones en la tabla. Si la búsqueda que se realiza sobrepasa el final de la tabla, la siguiente llamada a msqlFetchRow() retornará NULL.

A.8.8. msqlNumRows()

```
int msqlNumRows ( result )
m_result * result;
```

El número de renglones retornados por una consulta puede ser encontrado al llamar a msqlNumRows() y pasarle el resultado retornado por msqlStoreResult(). El número de renglones de datos enviados como resultado de la consulta es retornado como una variable de tipo entero.

Si una consulta SELECT no trae ningún dato, msqlNumRows() indicará que la tabla de resultados tiene 0 renglones (nota: las versiones anteriores de MSQL retornaban un resultado NULL si no se encotraba ningún dato. Esto se ha simplificado y hecho más intuitivo al retornar un resultado de 0 renglones como dato de resultado)

A.8.9. msqlFetchField()

```
m_field * msqlFetchField ( result )
m_result * result;
```

Junto con el actual renglón de datos, el servidor retornará información sobre los campos de datos seleccionados. Esta información queda disponible para el programa que la llamó a través de la función msqlFetchField(). Como msqlFetchRow(), esta función retorna un elemento de información a la vez y retorna NULL si no existe información disponible. El dato es retornado en una estructura m_field, la cual contiene la siguiente información:

```
typedef struct {
char* name ; /* nombre de la tabla */
int type ; /* tipo de datos de la tabla*/
int length /* longitud en bytes del campo */
int flags ; /* banderas de atributo*/
}m_field;
```

Posibles valores para el tipo de campo están definidos en msql.h as INT-TYPE, CHAR-TYPE and REAL-TYPE. Los atributos individuales de las banderas pueden ser accedidos usando las siguientes macros:

IS PRI KEY (flags /* El campo es la llave primaria */ IS-NOT-NULL (flags /* El campo no puede contener un valor NULL */

A.8.10. msqlFieldSeeko

```
void msqlFieldSeek ( result , pos )
m_result * result;
int pos;
```

La estructura de resultado incluye un cursor" para el campo de datos. Su posición puede ser cambiada utilizando la función msqlFieldSeek(). Ver msqlDataSeek() para más detalles.

A.8.11. msqlNumFields()

```
int msqlNumFields ( result )
m_result * result ;
```

El número de campos retornados por una consulta puede ser averiguado al llamar a msqlNumFields() y pasarle el resultado. El valor retornado por msqlNumFields() indica el número de elementos en el vector de datos retornado por msqlFetchRow(). Es recomendable verificar el número de campos retornados anteriormente, como en todos los arreglos, acceder al elemento que se encuentra más allá del final del vector de datos puede resultar en una falla de segmentación.

A.8.12. msqlClose()

```
int msqlClose ( sock)
int sock;
```

La conexión al motor MSQL puede cerrarse usando msqlcloseo. La función debe ser llamada con el socket de conexión retornado por msqlconnecto cuando la conexión inicial fué establecida.

A.9. Funciones relativas a esquemas

A.9.1. msqlListDBs()

```
m_result * msqlListDBs ( sock)
int sock;
```

Una lista de bases de datos conocida como motor MSQL puede ser obtenida a través de la función msqlListDBs(). Un resultado es retornado al programa desde el que se llamó a la función que puede ser usado para acceder a los nombres de las bases de datos. Los nombres individuales son accedidos mediante el llamado de la

función msqlFetchRow() al pasarle el resultado. La estructura m_row retornada al interceptar la llamada contendrá un campo que es el nombre de una de las bases de datos disponibles. Tal como en otras funciones que retornan un resultado, los datos asociados con el resultado deben ser liberados cuando ya no se les necesite más utilizando msqlFreeResult().

A.9.2. msqlListTables()

```
    m_result * msqlListTables (sock)
int sock;
```

Una vez que se ha seleccionado una base de datos usando msqlInitDB(), una lista de las tablas definidas en esa base de datos puede ser revisada utilizando msqlListTables(). Como con msqlListDBs(), un resultado es retornado al programa que lo llama y los nombres de las tablas son almacenados en los rengones de datos, en donde el elemento 01 del renglón es el nombre de una tabla en la base de datos actual. El resultado debe ser liberado cuando ya no se le necesite más utilizando msqlFreeResult().

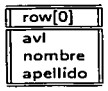
A.9.3. msqlListFields()

```
    m_result * msqlListFields ( sock , nombre_tabla );
int sock;
char * nombre_tabla;
```

La información sobre los campos de una tabla en particular puede obtenerse usando msqlListFields(). La función es llamada junto con el nombre de la tabla en la base de datos actual que se seleccionó con msqlSelectDB() y el resultado es retornado al programa que lo llamó. A diferencia de msqlListDBs() y msqlListTables(), la información del campo se encuentra contenida en estructuras de campo en lugar de renglones de datos. Esta es accedida utilizando msqlFetchField(). El resultado debe ser liberado cuando ya no se le necesite más utilizando msqlFreeResult().

```
    msqlListIndex()
    m_result * msqlListIndex ( sock, nombre_tabla , index )
int sock ;
char * nombre_tabla;
char * index;
```

La estructura de una tabla de índices puede ser obtenida del servidor utilizando la función msqlListIndex(). La tabla resultante retornada contiene un campo. El primer renglón del resultado contiene los nombres de los campos que comprenden el índice. Por ejemplo, si un índice compuesto fué definido como un Arbol AVL de índices y se basó en los valores de los campos Nombre y Apellido, entonces el resultado se verá como sigue:



Actualmente el único tipo de indice válido es 'avl', y significa un Arbol AVL de memoria mapeado.

Ultima actualización 23 Aug 1999.

A.10. Variables del Sistema Mini SQL 2.0

A.10.1. Introducción

Mini SQL 2.0 incluye soporte interno para variables de sistema (comúnmente conocidas como pseudo campos o pseudo columnas).

Estas variables pueden ser accedidas en la misma forma que campos de tabla normales son accesados, a menos que la información provenga del motor de base de datos y no cargado de alguna tabla. Las variables de sistema se utilizan para dar acceso al servidor manteniendo la información o los meta datos relacionados a las bases de datos.

Las variables de sistema pueden identificarse por un guión bajo que precede al nombre de la variable. Tal identificador no es válido en MSQL para tablas o campos de nombre. Ejemplos de las variables soportadas por el sistema y sus usos son resumidos a continuación.

A.11. Variables de sistema disponibles

El motor de MSQL 2 actualmente soporta las siguientes variables de sistema:

A.11.1. _rowid

La variable rowid, provee un identificador de renglón único para cualquier renglón en una tabla. El valor contenido en esta variable es un número de registro interno utilizado por el motor MSQL para acceder al renglón de la tabla. Puede ser incluido en alguna consulta para identificar de forma única a un renglón dentro de una tabla. Un ejemplo de dicha consulta podría ser:

```
select _rowid, nombre, apellido from empleados
where apellido = 'Smith'

update emp_detalles set title = 'Gerente'
where  _rowid = 57
```

Se pueden utilizar valores _rowid para optimizar el rendimiento de la base de datos. En el segundo ejemplo de arriba, solo un renglón (el renglón con el ID de registro interno 57) será accedido. En una búsqueda secuencial, se busca a través de toda la base de datos hasta encontrar el renglón en el que está el dato que se busca para ser modificado, sin embargo cada renglón es accedido.

Usar el valor _rowid para construir búsquedas es el método de acceso más rápido disponible en MSQL 2.0. Como en todas las decisiones de acceso interno, la decisión de basar los accesos a la tabla en el valor _rowid es automática y no requiere más acción por parte del programador o usuario que el colocar la variable _rowid en la cláusula WHERE de la consulta.

A.11.2. _timestamp

La variable de sistema _timestamp contiene la hora en la cuál el renglón fué modificado. El valor está especificado en el formato estándar de UNIX, no está disponible para interpretación de software de aplicación. Se pretende que el valor se use como punto de referencia para que una aplicación pueda determinar si un renglón en particular fué modificado antes o después de algún otro renglón de la tabla. La aplicación no debe tratar de establecer una hora actual para este valor, pues como representación interna puede cambiar en futuras versiones de MSQL.

El principal uso para la variable de sistema _timestamp es interno para el motor de MSQL. Usando esta información, el motor puede derteminar si algún renglón ha sido modificado después de un punto en específico en el tiempo (el inicio de una transacción por ejemplo). Puede utilizar también esta variable para sincronizar una base de datos remota para replicación de bases de datos. Aunque ninguna de estas funciones se encuentran aún disponibles, la presencia de una variable _timestamp es el primer paso de la implementación.

Ejemplo de consultas podrían ser:

```
 select nombre, _timestamp from empleados
where nombre like 'fred'
order by _timestamp

select * from empleados where _timestamp $>$ 88880123
```

A.11.3. _sec

La variable de sistema _sec se usa para acceder al valor de la secuencia actual de la tabla desde la cual se hace la selección. El valor de la secuencia actual es retornado y la secuencia se actualiza al siguiente valor en secuencia (ver la sección CREATE de la Especificación del Lenguaje para mayor información sobre secuencias).

Un ejemplo de consulta utilizando _seq podría ser:

```
 select _seq from staff
```

A.11.4. _sysdate

El servidor puede contener algún estándar central para la fecha y hora actuales. si se selecciona de alguna tabla la variable _sysdate retornará la hora actual y la fecha en la máquina del servidor utilizando el formato de hora estándar de UNIX.

Un ejemplo de consulta utilizando _sysdate puede ser:

```
select _sysdate from staff
```

A.11.5. _user

Al seleccionar la variable de sistema _user de alguna tabla, el servidor retornará el username del usuario que realizó la consulta.

Un ejemplo de consulta utilizando _user puede ser:

```
select _user from staff
```


Última actualización 23 Aug 1999.

A.12. Utilerías Estándar de Mini SQL 2.0

A.12.1. El monitor - msql

Uso msql [-h host] [-f arch_conf] base_de_datos

Opciones -h especifica un nombre de servidor remoto o una dirección IP en el cual el servidor MSQL estácorriendo.
Por default se conecta al servidor local utilizando el socket de dominio de UNIX en lugar del TCP/IP (el cual da mejor rendimiento)
-f Especifica un archivo de configuración no-default para ser cargado. La acción por default es cargar el archivo estándar de configuración ubicado en INST-DIR/msql.conf (generalmente lusr/locaVHughes/msql.conf)

Descripción El monitor MSQL es una interfaz interactiva al servidor MSQL. Esto permite enviar comandos SQL directamente al servidor. Cualquier sintáxis válida de MSQL puede ser ingresada desde la línea de comandos del monitor MSQL.
El control del monitor en si mismo viene con 4 comandos internos. Cada comando consta de una diagonal invertida seguida por un caracter simple. Los comandos disponibles son:
\q Quit
\g Fo (Envía la consulta al servidor)
\e Edit (Edita la consulta previa)
\p Print (Imprime el buffer de la consulta)

Visor de Esquema - relshow

Uso relshow [-h host] [-f arch_conf] [base_de_datos [rel [idx]]]

Opciones -h especifica un nombre de servidor remoto o una dirección IP en el cual el servidor MSQL está corriendo.
Por default se conecta al servidor local utilizando el socket de dominio de UNIX en lugar del TCP/IP (el cual da mejor rendimiento)
-f Especifica un archivo de configuración no-default para ser cargado. La acción por default es cargar el archivo estándar de configuración ubicado en INST-DIR/msql.conf (generalmente lusr/locaVHughes/msql.conf)

Opciones -h especifica un nombre de servidor remoto o una dirección IP en el cual el servidor MSQL está corriendo.
Por default se conecta al servidor local utilizando el socket de dominio de UNIX en lugar del TCP/IP (el cual da mejor rendimiento)
-f Especifica un archivo de configuración no-default para ser cargado. La acción por default es cargar el archivo estándar de configuración ubicado en INST-DIR/msql.conf (generalmente lusr/locaVHughes/msql.conf)

Descripción Relshow se usa para desplegar la estructura de los contenidos de las bases de datos MSQL. Si no tiene argumentos, relshow listará los nombres de las bases de datos actualmente definidas. Si se le da un nombre de base de datos, listará las tablas definidas en esa base de datos. Si se le da también el nombre de una tabla, desplegará la estructura de la tabla (por ejemplo nombres de los campos, tipos, longitudes, etc.) Si se le da un nombre de índice, junto con la base de datos, y nombres de las tablas, relshow desplegará la estructura del índice especificado, incluyendo el tipo del índice y los campos que comprenden el índice

Administrador del programa - msqladmin

Uso msqladmin [-h host] [-f arch_conf] [-q] Comando

Opciones -h especifica un nombre de servidor remoto o una dirección IP en el cual el servidor MSQL está corriendo.
Por default se conecta al servidor local utilizando el socket de dominio de UNIX en lugar del TCP/IP (el cual da mejor rendimiento)
-f Especifica un archivo de configuración no-default para ser cargado. La acción por default es cargar el archivo estándar de configuración ubicado en INST-DIR/msql.conf (generalmente lusr/locaVHughes/msql.conf)
-q Pone al msqladmin en modo lento. Si esta bandera es activada, msqladmin no va a a pedir al usuario verificar acciones peligrosas (tales como eliminar una base de datos)

Descripción msqladmin se utiliza para ejecutar operaciones administrativas en un servidor de base de datos MSQL. Dichas operaciones incluyen la creación de bases de datos, dar de baja sistemas, etc. Los comandos disponibles para msqladmin son:
create db-name Crea una nueva base de datos llamada db-name
drop db_name Elimina la base de datos llamada db_name del servidor. Esta acción eliminará también todos los datos contenidos en la base
shutdown Finaliza el servidor MSQL
reload Forza al servidor a volver a cargar la información ACL
version Despliega la versión e información de la configuración sobre el servidor que actualmente corre
stats Despliega estadísiticas del servidor
Nota: Muchas funciones administrativas pueden ser ejecutadas únicamente por el usuario especificado en la configuración en tiempo de ejecución como administrador. También pueden ser ejecutadas del servidor en el cal el proceso estácorriendo (por ejemplo, no se puede dar de baja un proceso de un servidor remoto).

Vaciador de datos - msqldump

Uso msqldump [-h host] [-f arch_conf] [-c] [-v] base_de_datos [tabla]

Opciones -h especifica un nombre de servidor remoto o una dirección IP en el cual el servidor MSQL está corriendo.
Por default se conecta al servidor local utilizando el socket de dominio de UNIX en lugar del TCP/IP (el cual da mejor rendimiento)
-f Especifica un archivo de configuración no-default para ser cargado. La acción por default es cargar el archivo estándar de configuración ubicado en INST-DIR/msql.conf (generalmente lusr/locaVHughes/msql.conf)
-c Incluye los nombres de columnas en comandos INSERT generados por el dump
-v Corre en modo verboso. Esto significa que se desplegarán los detalles tales como los resultados de la conexión, etc.

Descripción msqldump produce un archivo de texto ASCII que contiene comandos válidos de SQL que recrean la tabla o la base de datos corrupta que fué arrojada por el programa de monitor de MSQL. La salida incluirátodos los comandos CREATE TABLE requeridos para recrear las estructuras de las tablas, comandos CREATE INDEX para recrear los índices, y comandos INSERT para llenar las tablas con los datos actualmente contenidos en las tablas.
Nota: msqldump no recrea secuencias hasta este momento.

Exportador de datos - msqlexport

Uso msqlexport [-h host] [-f confl [-v] [-s Charl [-q Char] [-e Char] base_de_datos tabla

Opciones -h especifica un nombre de servidor remoto o una dirección IP en el cual el servidor MSQL estácorriendo.
Por default se conecta al servidor local utilizando el socket de dominio de UNIX en lugar del TCP/IP (el cual da mejor rendimiento)
-f Especifica un archivo de configuración no-default a ser cargado. La acción por default es cargar el archivo estándar de configuración ubicado en INST-DIR/msql.conf (generalmente /usr/local/Hughes/msql.conf)
-v modo verboso
-s Utilice el caracter Char como el caracter de separación. El default es la coma. -q encomilla cada valor con el caracter especificado
-e Utiliza lo especificado en Char como el caracter de escape. El default es

Descripción msqlexport produce una exportación ASCII del dato de la tabla especificada. La salida producida puede utilizarse como entrada para otros programas como hojas de cálculo. Ha sido diseñado para ser tan flexible como sea posible, permitiendo al usuario especificar el caracter a usar para separar los campos, el caracter a usar para saltarse el caracter separador si es que aparece dentro del dato, si es que el dato debe ser entrecomillado y si es así que caracter se utilizará como comilla.
La salida se manda a stdout con un renglón de datos por linea.

Importador de datos - msqlimport

Uso msqlimport [-h host] [-f conf] [-v] [-s Charl [-e Char] [-c col,col ... 1 tabla de base_de_datos

Opciones -h especifica un nombre de servidor remoto o una dirección IP en el cual el servidor MSQL estácorriendo.
Por default se conecta al servidor local utilizando el socket de dominio de UNIX en lugar del TCP/IP (el cual da mejor rendimiento)
-f Especifica un archivo de configuración no-default a ser cargado. La acción por default es cargar el archivo estándar de configuración ubicado en INST-DIR/msql.conf (generalmente /usr/local/Hughes/msql.conf)
-v modo verboso
-s Utilice el caracter Char como el caracter de separación. El default es la coma. -e utiliza el caracter especificado como caracter de escape. El default es
-c Una coma que separa listas de nombres de columna dentro de las cuales los datos son insertados. NOta: puede no haber espacio en la lista.

Descripción msqlimport carga un archivo de datos ASCII en una tabla de la base de datos MSQL. El archivo puede formatearse utilizando algún caracter como separador de columnas. Cuando pasa a través de msqlimport, cada línea del archivo txt se carga como un renglón dentro de la tabla de la base de datos.
El caracter separador se especifica con la bandera -s, se utilizarápara dividir la linea de texto en columnas.
Si el dato utiliza algún caracter específico de escape para cualquier ocurrencia del caracter de separación en el dato, el caracter de escape puede ser especificado con la bandera -e y será eliminado del dato antes de ser insertado.

Bibliografía

[1] Korth, Henry F. *Fundamentos de bases de datos.* McGraw-Hill, México, 1987

[2] Kendall, Kendall. *Análisis y diseño de sistemas.* Prentice Hall, México, 1991

[3] Kabir, Mohammed J. *Servidor Apache.* Anaya, California, USA, 1999

[4] Yarger Jay, Randyi Reese, George and King Tim. *MySQL y MSQL.* O'Reilly, California, USA, 1999

[5] Atkinson, Leon. *Core PHP programming.* Prentice Hall, USA, 1999

[6] Cosentino, Christopher. *Essential PHP for web professionals.* Prentice Hall, USA, 2000

[7] Weinman, William E. *El libro de CGI.* Prentice Hall Hispanoamerica, 1996

[8] Raya, Jose Luis y Rodrigo Victor. *Domine TCP/IP.* RA-MA, 1997

[9] Rowe, Jeff. *Creación de servidores de base de datos para Internet con CGI.* Prentice Hall Hispanoamerica, 1996

[10] Murdick, Robert G. and Munson John. *Sistemas de información administrativa.* Prentice Hall Hispanoamerica, 1998

[11] Lerdorf, Ramus. *PHP Pocket reference.* O'Reilly, 2000

[12] Castagnetto, Jesus y otros. *Professional PHP programming.* Wrox , 1999

[13] Converse, Tim and Joyce Park. it PHP 4 bible. Foster City, California : IDG Books Worldwide, 1990

[14] Raya Cabrera, Jose Luis, otros. *HTML 4 : guía de referencia y tutorial* México, D. F. : Alfaomega, 1999

[15] Daines, Derrick. *Las bases de datos en la educación básica : Utilización y ejemplos* México, D. F. : Alfaomega, 1999

[16] *http//:www.super.unam.mx*

[17] *http//:www.apache.org*

[18] *http//:www.php.net*

[19] *http//:www.redhat.com*

[20] *http//:www.huhges.com.au*