UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLÁN"

PROGRAMACIÓN CGI CON PERL Y UNA APLICACIÓN A LAS BASES DE DATOS RELACIONALES

T E S I N A

QUE PARA OBTENER EL TÍTULO DE:

LIC. EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

P R E S E N T A

JUAN CARLOS LÓPEZ PEÑAFORT

ASESOR: ING. RUBÉN ROMERO RUÍZ

JUNIO 2003

A

Tus desvelos, tus preocupaciones, tus anhelos, tu amor, tu amistad, tu cariño...
Todo valió la pena, he llegado al logro de una de nuestras metas;
Mamá, gracias por darme la vida y por darme una vida.

A ustedes hermanos: Martín, Santa y Flavio, Andrea;
que sin su apoyo, comprensión, motivación y orientación
no hubiese llegado a este logro.

A mis sobrinos: Alaric, Martín e Itzel;
nunca se den por vencidos, siempre existe un camino para lograr sus metas.

A mis tíos y tías: Abel e Inés; Felicitas y Bonfilio; Jaime y Juana; Josefina y Ciro;
Carlos y Raquel; Martha; Ernesto y Maria.
Gracias por su apoyo.

A mi asesor y sinodales; y a todos los profesores
que han influido en mí durante estos 25 años de mi vida

Y finalmente, gracias a todos los que hicieron posible la realización de este trabajo.

Marisol
La primera vez que te vi, quise estar cerca de ti
Cuando te conocí, me enamoré de ti
Ahora que estamos juntos, no me pienso separar de ti
porque somos uno solo,
porque en la eternidad solo existe amor...
amor que yo siento por ti.

Gracias por tu apoyo.

INDICE

INTRODUCCIÓN

El acceso a las bases de datos en Internet ha llevado a la generación de nuevas tecnologías que permiten explotar la información desde cualquier parte del mundo.

Existen varios lenguajes de programación que facilitan el acceso a bases de datos, algunos de ellos dependen de una plataforma de sistema operativo, como los ASP's de Microsoft, sin embargo existen otros que trabajan en la mayoría de las plataformas de sistemas operativos como: Perl, JAVA y PHP, entre otros.

Perl cuyas siglas significan: Lenguaje Práctico de Extracción e Informes (Practical Extraction and Report Language), en un principio tuvo como intención, llevar a cabo la manipulación de documentos de texto (parseo, extracción...) pero rápidamente comenzó a ser algo más que eso.

Con Perl se puede hacer todo, desde tareas administrativas, hasta scripts cgi para la creación de aplicaciones complejas y realización de interfaces de bases de datos. La filosofía de Perl es: "Hay más de una forma de hacerlo".

Así como existen varios lenguajes de programación, también existen numerosos manejadores de bases de datos, como MySQL, Oracle, DB2, Ingress, Progress, etc., la mayoría de ellos es multiplataforma, es decir, existe el manejador para cada sistema operativo, pero la principal característica de cada uno de ellos, es su costo. El manejador MySQL es gratuito y esta liberado bajo la GNU (es un acrónimo recursivo para "GNU No es Unix") GPL (General Public Licence) para ser libre, dependiendo para lo que se le utilice, por lo que cuantiosas empresas están migrando sus bases de datos a este manejador.

La conjunción del lenguaje de programación Perl y el manejador de base de datos MySQL, genera un matrimonio por conveniencia, ya que cada uno de ellos aporta características particulares para el desarrollo y crecimiento de una institución ó una empresa en el ambito tecnológico.

De lo anterior surge el interés del desarrollo de esta tesina, que tiene como objetivo: Explicar los conceptos básicos de la programación de scripts CGI, así como su construcción y aplicación al acceso a bases de datos relacionales en el ambiente Web, demostrando que el lenguaje de programación PERL es sencillo, robusto y confiable.

PAGINACION DISCONTINUA

Para comprender y entender el contexto de este trabajo se necesitan conocimientos básicos de bases de datos y lenguajes de programación. Por lo que la tesina se divide en los siguientes temas:

En la parte I se proveen los conocimientos básicos para entender el ambiente de los CGI's. Por lo que este capítulo aborda los principales conceptos para entender como funcionan los CGI's, además contiene una visión general del Lenguaje HTML enfocándose al uso de formularios y su relación con los CGI.

Parte II: En este apartado se proporcionan las características del leguaje de programación Perl, así como la escritura de scripts CGI's y sus aplicaciones. Este capitulo inicia con una descripción general del lenguaje Perl, además contiene algunos ejemplos que muestran la facilidad para crear programas. Explica la forma de escribir scripts CGI's y como utilizar la librería CGI.pm para perfeccionarlos. Además muestra algunas aplicaciones que son utilizadas en la Web para interactuar con los usuarios, es decir, aquellas páginas Web cuyo contenido depende de las peticiones que el usuario realiza a partir de formularios HTML.

En la parte III se muestran las características de una base de datos relacional, así como su teoría y su aplicación. En este tema se muestra la teoría del Dr. Codd sobre las bases de datos relacionales, así como sus operaciones y como estas operaciones se pueden llevar a acabo sobre la base de datos con el lenguaje de cuarta generación SQL. Al final de este capítulo se ejemplifica esta teoría.

Finalmente, en la Parte IV se demuestra la facilidad de acceso a las bases de datos relaciones y la presentación de datos en el Navegador Web utilizando el lenguaje de programación Perl. Este capítulo concluye la tesina, ejemplificando el uso conjunto de Perl con HTML, Apache Web Server y MySQL, primeramente instalando los productos y posteriormente su integración.

El Protocolo de Transferencia de Hipertexto (HTTP) es el lenguaje común de los navegadores y de los servidores Web y estos lo utilizan para comunicarse con cualquier otro protocolo en el Internet.

CGI (Common Gateway Interface) esta construido en la parte superior del HTTP, pero para entender completamente al CGI, necesitamos primero entender HTTP. Una de las razones de la eficacia del CGI, es que puede manipular el intercambio de metadatos entre el navegador Web y el servidor, además tiene numerosas ventajas, incluyendo:

- Servir de contenido a lenguajes, o cualquier código según las necesidades del cliente.
- Comprueba previamente la localización del usuario.
- Comprueba el tipo de navegador, la versión y se adapta a su respuesta.
- Especifica cuánto tiempo el cliente puede tener en caché una página antes de ser tratada como antigua y deba recargarse.

No se cubrirán todo los detalles de HTTP, solo lo que es importante para explicar y entender el CGI. Específicamente, hay que enfocarse en el proceso sobre la petición y la respuesta: cómo los navegadores piden y reciben páginas Web.

1.1 INTRODUCCIÓN AL CGI

El CGI es una interfaz ligera, es esencialmente lo mínimo que el servidor Web necesita para proveer el orden y permitir procesos externos para crear páginas Web. Tradicionalmente cuando un servidor Web recibe una petición para una página estática, busca el archivo HTML correspondiente en el sistema de archivos. Cuando un servidor Web recibe una petición para un script CGI entonces ejecuta el script CGI como otro proceso; el servidor pasa el proceso con algunos parámetros y recoge su salida que devuelve entonces al cliente así como fue obtenido hacia un archivo estático.

1.1.1 Definición

CGI es un estándar para comunicar aplicaciones externas con los servidores Web. Un script CGI es ejecutado en tiempo real, así que genera información dinámica.

Primeramente hay que diferenciar entre programas y scripts, los programas se consideran escritos en algún lenguaje compilado, como "C", mientras que los scripts son escritos en un lenguaje interpretado, como Perl.

En general, se necesita la presencia de dos elementos: una página Web en formato HTML con un formulario donde el usuario introduce sus datos y un programa CGI en el servidor que recibe y procesa los datos del usuario.

CGI se utiliza comúnmente para contadores, bases de datos, motores de búsqueda, formularios, generadores de e-mail automático, foros de discusión, chats, comercio electrónico, mapas de imágenes, juegos en línea, entre otros.

Esta tecnología tiene la ventaja de ejecutarse en el servidor cuando el usuario lo solicita por lo que es dependiente del servidor y no de la computadora del usuario.

Los programas que utiliza el CGI pueden estar compilados en diferentes lenguajes de programación. El funcionamiento de esta tecnología es muy sencilla. Los scripts residen en el servidor donde son llamados y ejecutados y regresan información de vuelta al usuario. Para una mejor comprensión ver la Figura 1.1:

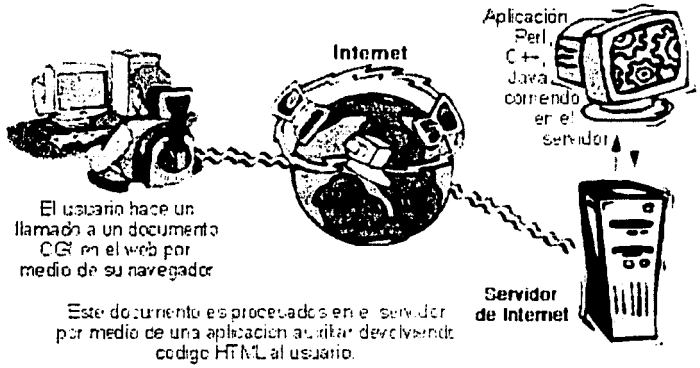


Fig 1.1 Proceso de funcionamiento del CGI

De acuerdo a la traducción de la NCSA: *"Un documento HTML es estático, lo que significa que existe en un estado constante; es un archivo de texto que no cambia. Un script CGI por otro lado, es ejecutado en tiempo real, lo que permite que regrese información dinámica. Por ejemplo, digamos que quieres conectar tus bases de datos de Unix al World Wide Web para permitir que las personas de todo el mundo la manipulen. Básicamente, lo que debes hacer es crear un script CGI que será ejecutado por el servidor para transmitir información al motor de la base de datos, recibir los resultados y mostrárselos al cliente. Este es un ejemplo sencillo que muestra donde el CGI tiene sus orígenes".* [1]

1.1.2 Tecnologías Alternativas

En años recientes han aparecido varias tecnologías. Todas ellas se construyen en el legado del CGI y proporcionan sus propios acercamientos a la misma meta subyacente: responder a consultas y presentar el contenido dinámicamente vía HTTP. La mayoría de ellos también intenta evitar la desventaja principal de los scripts CGI: Crear un proceso separado para ejecutar el script cada vez que sea solicitado. Otros también intentan hacer menos distinción entre páginas HTML y el código que se pasa a las páginas HTML.

ASP

ASP responden al nombre Active Server Pages, que en español significa Páginas de Servidor Activas.

Agrupadas en la categoría de lenguajes de script las páginas ASP contienen además de los tags (etiquetas) de HTML habituales en las páginas Web, fragmentos de código que el servidor resolverá antes de enviarlo al navegador.

La facilidad para conectar con una base de datos y extraer datos de la misma dinámicamente visualizándolos en el navegador es la utilidad más utilizada de las páginas ASP. Puede conectarse a manejadores de base de datos SQL, Access, Oracle, o cualquier otro motor que disponga de un driver o manejador ODBC, además tiene distintas aplicaciones dentro del comercio electrónico, portales y todas aquellas aplicaciones en las que el protagonista es la información dinámica.

---

[1] *http://hoohoo.ncsa.uiuc.edu/cgi/* National Center for Supercomputing Applications

Para procesar una página ASP no existe ninguna restricción especial en el lado del cliente, por lo que es indiferente la utilización del navegador Internet Explorer o Netscape Comunicator sin embargo, en el lado del servidor es necesario un servidor Web de Microsoft. Se utiliza el archivo ASP.DLL para interpretar el código, siendo el servidor más extendido Internet Information Server (más conocido como IIS).

Estos son los servidores de contenidos ASP posibles para plataformas Microsoft:

- Internet Information Server 3.0 o superior (para sistema operativo NT)
- Personal Web Server (para Windows 95 y Windows 98)
- Para plataformas Unix es necesario añadir un software que actué de intérprete siendo algunos de los más conocidos: Chilisoft, Instant ASP.

Java Servlets

Los Servlets son módulos que extienden los servidores orientados a petición-respuesta, como los servidores Web compatibles con Java.

Estos proporcionan una forma de generar documentos dinámicos que son fáciles de escribir y rápidos en ejecutarse. También solucionan el problema de hacer la programación del lado del servidor con APIs específicos de la plataforma: están desarrollados con el API Java Servlet, una extensión estándar de Java.

Se deben compilar como clases antes de que sean ejecutadas, además están dinámicamente cargados como clases en el servidor Web.

Los Servlets pueden reenviar peticiones a otros servidores y servlets. Con esta ventaja pueden ser utilizados para cargar balances desde varios servidores que reflejan el mismo contenido; y para particionar un único servicio lógico en varios servidores de acuerdo con los tipos de tareas ó la organización compartida.

PHP

PHP es un lenguaje de programación el cual se ejecuta en los servidores Web y permite crear contenidos dinámicos en páginas HTML.

Dispone de múltiples herramientas que permiten acceder a bases de datos de forma sencilla, por lo que es ideal para crear aplicaciones para Internet.

Es multiplataforma ya que funciona tanto para Unix (Apache), como para Windows (Microsoft Internet Information Server) de forma que el código que se haya creado para una de ellas, no tiene porqué modificarse al pasar a la otra. La sintaxis que utiliza, la toma de otros lenguajes muy extendidos como "C" y Perl.

Al ser PHP un lenguaje que se ejecuta en el servidor, no es necesario que el navegador lo soporte, es independiente del navegador, pero sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

Para entender complemente el funcionamiento del CGI, se tienen que mencionar algunos aspectos importantes del nivel inferior: el protocolo HTTP.

1.2 URL's

URL significa Uniform Resource Locators (Localizador Uniforme de Recursos). En los términos de la Web, un recurso representa algo disponible: una página HTML, una imagen, un script CGI, etc., por lo que los URL's proporcionan un camino estándar para localizar estos recursos en la Web.

Los URLs no son realmente específicos para HTTP, sin embargo, estos pueden referirse a recursos en varios protocolos.

1.2.1 Elementos de un URL

Los URLs están formados de un método, nombre de host, número de puerto, ruta, cadena de consulta y referencia, cualquiera puede omitirse bajo ciertas circunstancias (Figura 1.2)

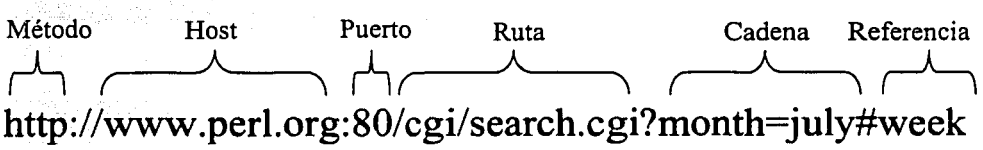


Fig 1.2 Elementos de un URL

Método

El método representa el protocolo como por ejemplo http, ftp, news, telnet, etc., y para el este propósito serán los HTTP ó HTTPs los que representan una conexión a un servidor Web seguro.

Host

El Host identifica la máquina en ejecución en el servidor Web. Puede ser un nombre de un dominio o una dirección IP, aunque es una mala idea usar direcciones IP en URL's. El problema es que direcciones IP cambian muy a menudo, sin embargo un sitio Web puede moverse de una máquina a otra o puede trasladarse a otra red sin tener ningún problema. Los nombres del dominio pueden permanecer constantes en estos casos, mientras esos cambios permanezcan ocultos al usuario.

Número de Puerto

El número del puerto es opcional y sólo puede aparecer en URL's si el host también se incluye. El Host y el puerto están separados por dos puntos. Si el puerto no se especifica, el puerto 80 se usa por default para http y el puerto 443 para https.

Información de la Ruta

La información de la ruta representa la localización del recurso solicitado como un archivo HTML o un script CGI. Dependiendo de cómo este configurado el servidor Web puede o no tener alguna ruta real en el sistema. La ruta del URL para los scripts CGI generalmente empieza con */cgi/* ó */cgi-bin/* y estas rutas se localizan en un directorio semejante nombrado en el servidor Web, como por ejemplo: */usr/local/apache/cgi-bin* en sistema UNIX o *C:\Program Files\Apache Group\Apache\cgi-bin* en Win32.

El URL para un script puede incluir la información de la ruta más allá de la ubicación del script. Por ejemplo, un CGI puede estar en:

http://localhost/cgi/browse_docs.cgi

Cadena de consulta

Una cadena de consulta pasa parámetros adicionales al script. A veces está referido a una búsqueda de una cadena o índice. Puede contener pares de nombre-valor y

cada par está separado por un ampersand (&) del siguiente par; y el nombre del valor está separado por un signo de igual (=).

Una cadena de consulta también puede incluir datos que no se estructuran como los pares de nombre-valor. Si una cadena de consulta no contiene signos iguales, está a menudo referido hacia un índice. Entonces, cada argumento debe separarse del siguiente por un espacio codificado (puesto en código como + o %20).

Referencia

Una referencia es una sección específica en un recurso. Las referencias no son enviadas a los servidores Web por lo que no se puede tener acceso a este componente de los URLs en los scripts CGI. En cambio, los navegadores muestran un recurso y entonces aplican la referencia para localizar la sección apropiada. Para documentos HTML la referencia es el tag (etiqueta) ancla dentro del documento.

* Tema 6.Teoría de Gráficas ... *

El siguiente URL pide el documento completo y entonces se desplazará a la sección marcada por la etiqueta de ancla:

http://localhost/documento.html#Tema_6

Los navegadores Web generalmente saltan al fondo del documento si no se encuentra ninguna ancla de referencia.

1.2.2 URL's Absolutos y Relativos

Algunos de los elementos dentro de los URL son opcionales. Se puede omitir el método, host y número de puerto, si el URL se usa en un contexto donde estos elementos se suponen. Por ejemplo, si se incluye una liga sobre una página HTML y se omiten estos elementos, el navegador asumirá la liga aplicada a un recurso sobre la misma máquina de esa liga. El navegador al buscar la ruta del recurso y puede encontrarse con dos clases de URLs:

URL Absolutos

Los URLs que incluyen el nombre del host son llamados URLs absolutos. Un ejemplo es *http://www.localhost/cgi/script.cgi.*

URL Relativos

Los URLs sin método, host o puerto son llamados URL relativos. Estos pueden separarse en rutas completas y relativas.

Rutas completas: Los URL relativos con una ruta absoluta son algunas veces referidas como rutas completas (sin embargo, algunas veces se puede incluir una cadena de consulta y una referencia). Las rutas completas pueden ser distinguidas para un URL con una ruta relativa ya que estas siempre empiezan con un slash (barra inclinada hacia delante "/"). En esos casos las rutas son rutas virtuales, y no necesariamente corresponden a una ruta en el sistema de archivos en el servidor Web. Un ejemplo de una ruta absoluta es */index.html*

Rutas relativas: Los URLs relativos que empiezan con otro carácter que no es un slash son rutas relativas. Ejemplos: *script.cgi* e *../images/photo.jpg*

1.3 HTTP

El Protocolo de Transferencia de HiperTexto es un sencillo protocolo cliente-servidor que permite los intercambios de información entre los clientes y los servidores Web.

HTTP se basa en sencillas operaciones de petición/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la petición. El servidor responde con un mensaje similar que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto (documento HTML, archivo multimedia o aplicación CGI) es conocido por su URL como se mencionó en el subtema anterior.

1.3.1 El ciclo de Petición y Respuesta

Cuando un navegador solicita una página, este envía un mensaje de petición al servidor Web. El mensaje siempre incluye una cabecera y algunas veces también incluye un cuerpo. El servidor Web en turno contesta con un mensaje de respuesta.

Hay dos características que son importantes para comprender HTTP.

- Es un protocolo de petición/respuesta: Cada respuesta proviene de una petición.

- Aunque cada petición y respuesta contiene información diferente, la estructura cabecera/cuerpo es la misma para ambos mensajes. La cabecera contiene meta-información (información acerca del mensaje) y el cuerpo comprende el contenido del mensaje.

La Figura 1.3 muestra un ejemplo de una transacción HTTP. Si se pide al navegador que busque el documento *http://localhost/index.html* el navegador se conectará a la máquina *localhost* por el puerto 80 y enviará el siguiente mensaje:

GET /index.html HTTP/1.1
Host: localhost
*Accept: image/gif, image/x-xbitmap, image/jpeg, image/xbm, */**
Accept-Lenguaje: en
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 4.5.; Mac_PowerPc)

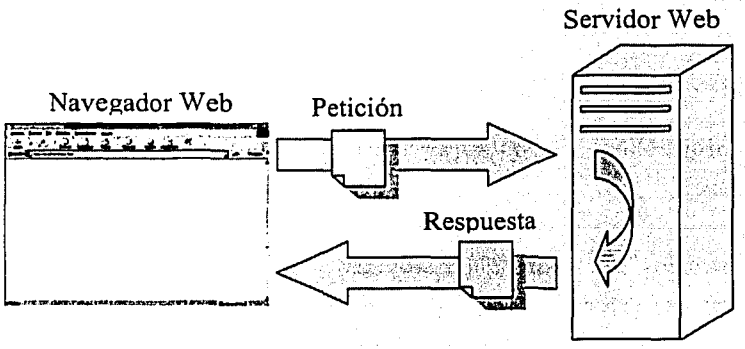


Fig 1.3 Mensaje de envío para buscar un documento HTML

Pensando que el servidor Web esta ejecutándose y la ruta apunta a un documento válido, el servidor contestará con el siguiente mensaje:

```
HTTP/1.1 200 OK
Date: Sat, 18 Mar 2000 20:35:35 GMT
Server: Apache 1.3.9 (Unix)
Last-Modified: Wed, 20 May 1998 14:59:42 GMT
ETag: "74916-656-3562efde"
Content-Length: 141
Content-Type: text/html
<HTML>
<HEAD><TITLE>Ejemplo de un Documento</TITLE></HEAD>
<BODY>
        <H1>Ejemplo de un Documento</H1>
        <P>Este es un prueba de un documento HTML</P>
</BODY>
</HTML>
```

1.3.2 Cabeceras HTML

Históricamente, el formato de mensajes HTTP esta basado en varias de las convenciones usadas por el correo de Internet, es establecido por MIME (Multipurpose Internet Mail Extensions). Sin embargo, las cabeceras HTTP y MIME no son iguales. La similitud sólo se extiende a ciertos campos y algunas similitudes han cambiado en las últimas versiones de HTTP.

Los siguientes puntos son importantes en la sintaxis de la cabecera:

- La primera línea de la cabecera tiene un formato único y un significado especial.

- El resto de las líneas de la cabecera contiene pares de nombre-valor. El nombre y el valor son separados por dos puntos, además de una combinación de espacios y/o tabuladores. Las líneas son llamadas campos de la cabecera.

- Algunos campos de la cabecera tienen múltiples valores. Estos pueden ser representados por campos de la cabecera conteniendo el mismo nombre de campo y diferentes valores ó incluyendo todos los valores en el campo de la cabecera separados por una coma.

- Los nombres de campo no son sensibles al contexto, por ejemplo Content-Type es lo mismo que Content-type.

- Los campos de las cabeceras no tienen que aparecer en un orden especial.

- Cada línea en la cabecera debe ser terminada por un retorno de carro y una línea de sucesión que a menudo se abrevia como CRLF y es representada en Perl sobre los sistemas ASCII como \015\012.

- La cabecera debe ser separada del contenido por una línea blanca. En otras palabras, la última línea de la cabecera debe terminar con dos CRLFs.

1.4 PETICIÓN DEL NAVEGADOR

Cada interacción HTTP empieza con una petición de un cliente, por ejemplo: Un navegador Web. Un usuario proporciona un URL al navegador dando un clic a un hipervínculo ó seleccionando un bookmark, y el navegador busca el documento correspondiente. Para hacer eso, debe crear una petición HTTP (Figura 1.4).

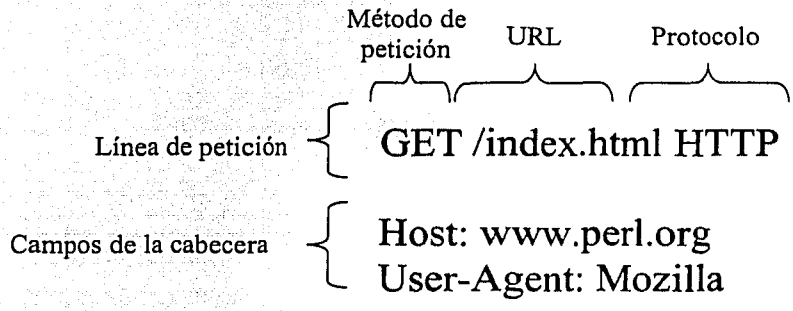


Fig 1.4 Petición HTTP realizada por el navegador

Del ejemplo anterior, un navegador Web generó la siguiente petición cuando se buscó el URL *http://www.localhost/index.html*:

```
GET /index.html HTTP/1.1
Host: localhost
Accept: image/gif, image/x-xbitmap, image/jpeg, image/xbm, */*
Accept-Languaje: en
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 4.5.; Mac_PowerPc)
```

El navegador crea una conexión de red usando el nombre de host y el número de puerto. El método dice al navegador que esta usando el protocolo HTTP, así la conexión es establecida, enviando una petición HTTP para el recurso. La primera línea de la petición incluye una ruta virtual y (si esta presente) una cadena de consulta (Figura 1.4).

1.4.1 La Línea de Petición

La primera línea de una petición HTTP incluye un método, un URL para el recurso solicitado y una sección de la versión del protocolo (Figura 1.4). Los métodos de petición son sensibles al contexto y deben ir en mayúsculas. Son varios los métodos definidos por HTTP (Tabla 1) aunque el servidor no pueda hacer que todos ellos estén disponibles para cada recurso. El nombre del protocolo y su versión van separadas por un slash, así que HTTP 1.0 y HTTP 1.1 son representados como HTTP/1.0 y HTTP/1.1. Las peticiones https sólo producen una de estas dos cadenas de protocolos HTTP.

TABLA 1. MÉTODOS DE PETICIÓN HTTP

Método	Descripción
GET	Pregunta al servidor por el recurso dado.
HEAD	Usado en los mismo casos que GET, solo que regresa cabeceras HTTP y no contenidos.
POST	Pregunta al servidor para modificar información guardada en el servidor.
PUT	Pregunta al servidor para crear o reemplazar un recurso en el servidor.
DELETE	Pregunta al servidor para borrar un recurso en el servidor.
CONNECT	Usado para permitir enlaces seguros SSL para profundizar en conexiones http.
OPTIONS	Pregunta al servidor la lista de métodos de petición disponibles para el recurso solicitado.
TRACE	Pregunta al servidor para hacer eco de las cabeceras de las peticiones así como el los recibió.

El estándar HTTP/1.0 recoge únicamente tres comandos: GET, HEAD y POST, que representan las operaciones de recepción, envío de información y chequeo de estado.

La última versión de HTTP denominada 1.1 recoge otras novedades como los comandos: PUT, DELETE, LINK, UNLINK.

Para los métodos de petición listados en la Tabla 1, los tres más importantes para escribir scripts CGI son GET, HEAD y POST y a continuación se describen:

1.4.2 GET

El método GET, es el método de petición estándar para recibir documentos vía HTTP. Cuando se da un clic sobre un hipervínculo, se escribe una dirección en el navegador ó se da un clic en un bookmark, el navegador generalmente crea una petición GET para ese URL. Las peticiones GET tienen la intención de recuperar los recursos. Estos no deben de alterar la información sino deben mantenerla en el servidor Web. Las peticiones GET no tienen un cuerpo de contenido.

Los navegadores asumen que las peticiones GET no tienen efectos laterales, pero se debe tener cuidado en hacer múltiples peticiones para el mismo documento. Por ejemplo, si el usuario presiona el botón regresar del navegador Web para volver a la página que originalmente se solicito vía GET y la página no tiene mucho tiempo en el cache del navegador, el navegador puede obtener una nueva copia. En cambio, si la petición original fuese vía POST, el usuario recibiría un mensaje indicando que el documento no esta disponible en el cache del navegador. Si el usuario decide recargar la petición, este recibiría un mensaje de confirmación indicando el reenvió de la petición POST. Estas características ayudan al usuario a evitar el envió de múltiples peticiones cuando el reenvió de esa petición podría modificar la información guardada ó enviada al servidor.

1.4.3 HEAD

Como se mencionó anteriormente, el navegador Web crea una petición GET para buscar recursos que han sido solicitados. Si el navegador fue previamente alimentado con el recurso, este puede estar guardado en el cache del navegador. Para saber si el navegador despliega la copia guardada en el cache o si solicita una nueva copia, el navegador envía una petición de HEAD. Dicha petición HEAD es en un formato

exactamente igual como la petición GET, entonces el servidor responde a él como si fuese una petición GET pero con una excepción: El navegador envía únicamente la cabecera HTTP y no hace el envío del contenido. El navegador puede entonces verificar la meta-información contenida en las cabeceras, como la modificación de la fecha del recurso para ver si este ha cambiado y si puede reemplazar la versión guardada en el cache con la nueva versión. Las peticiones HEAD no tienen un contenido.

1.4.4 POST

El método POST es usado en los formularios HTML para enviar datos que alteran la información guardada en el servidor. POST requiere siempre la inclusión de un cuerpo conteniendo la información estructurada que fue enviada con una cadena de consulta. Las peticiones POST requieren cabeceras adicionales especificando la longitud del contenido y su formato.

Aunque la petición POST debería ser únicamente usada para modificar los datos en el servidor, comúnmente los desarrolladores CGI utilizan las peticiones POST en scripts CGI para que simplemente regresan información, pero no para modificarla. Esta práctica es más común y menos peligrosa que la situación anterior (usando GET para modificar la información en el servidor).

Los desarrolladores usan POST para infinidad de situaciones:

- Algunos desarrolladores creen que enviando formularios vía POST ofrecen mayor seguridad encima de aquellos que envían vía GET, porque el usuario no puede modificar los valores en la URL del navegador cuando se envía con GET.

- La respuesta al recurso recibido vía POST no puede ser almacenado en el bookmark o hipervínculado. Aunque esto es generalmente un inconveniente para el usuario, pero algunas veces es lo correcto.

Los usuarios pueden encontrar mensajes de advertencia del navegador acerca de la expiración de las páginas que fueron obtenidas vía POST si ellos las revisan en el cache del navegador.

1.4.5 Los campos de la línea de cabecera

El cliente generalmente envía varios campos de la cabecera con una petición. Como se mencionó anteriormente, estos consisten en un nombre de campo, dos puntos, algunas combinaciones de espacios o tabuladores (aunque un espacio es más común) y un valor. Esos campos son usados para pasar información adicional acerca de la petición o acerca del cliente o para agregar condiciones a la petición. La tabla 2 contiene las cabeceras del navegador más comunes.

TABLA 2. CABECERAS HTTP MÁS COMUNES

Cabeceras	Descripción
Host	Especifica el nombre de host designado
Content-Length	Especifica la longitud (en bytes) de una petición de contenido
Content-Type	Especifica el tipo de media de la petición.
Authentication	Especifica el usuario y el password del usuario que pidió el recurso.
User-Agent	Especifica el nombre, versión y plataforma del cliente.
Referer	Especifica la URL que es referida al usuario para el actual recurso.
Cookie	Regresa un par de nombres y valores colocado por el servidor en una respuesta anterior.

1.5 *RESPUESTA DEL SERVIDOR*

El servidor responde tal como el cliente lo pidió; siempre contiene cabeceras HTTP y un cuerpo opcional. Así es como el servidor respondió en el ejemplo anterior:

```
HTTP/1.1 200 OK
Date: Sat, 18 Mar 2000 20:35:35 GMT
Server: Apache 1.3.9 (Unix)
Last-Modified: Wed, 20 May 1998 14:59:42 GMT
ETag: "74916-656-3562efde"
Content-Length: 141
Content-Type: text/html

<HTML>
<HEAD><TITLE>Ejemplo de Documento</TITLE></HEAD>
<BODY>
        <H1>Ejemplo de Documento</H1>
        <P>Este es un ejemplo de documento HTML</P>
</BODY>
</HTML>
```

La estructura de las cabeceras para las respuestas es la misma para la petición. La primera línea de la cabecera tiene un significado especial y es llamada la línea de estado. Las líneas restantes son líneas de campos de cabecera que tienen nombres y valores (Figura 1.5)

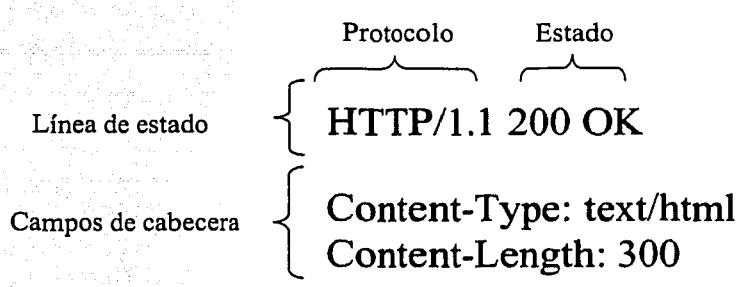


Fig 1.5 Línea de estado y campos de la cabecera

1.5.1 La Línea de Estado

La línea de estado es la primera línea de la respuesta y consiste en la versión de protocolo que se utiliza, seguida de una indicación de estado numérica a la que puede ir asociada una frase explicativa (Figura 1.5).

El código de estado es un número de 3 dígitos que indica si la petición ha sido atendida satisfactoriamente o no, y en caso de no haber sido atendida, indica la causa. Los códigos se dividen en cinco clases definidas por el primer dígito del código de estado. Así se tiene:

- 1xx: Informativo. La petición se recibe y sigue el proceso. Esta familia de respuestas indican una respuesta provisional. Este tipo de respuesta está formada por la línea de estado y las cabeceras. Un servidor envía este tipo de respuesta en casos experimentales.

- 2xx: Éxito. La acción requerida por la petición ha sido recibida, entendida y aceptada.

- 3xx: Redirección. Para completar la petición se han de tomar más acciones.

- 4xx: Error del cliente. La petición no es sintácticamente correcta y no se puede llevar a cabo.

- 5xx: Error del servidor. El servidor falla al atender la petición que aparentemente es correcta.

Algunos de los códigos más comúnmente usados y las frases asociadas se muestran en la Tabla 3:

TABLA 3. CÓDIGOS COMUNES DE ESTADO

Código	Significado
100	continuar
101	cambio de protocolo
200	éxito
201	creado
202	aceptado
203	información no autorizada
204	sin contenido
205	contenido reestablecido
206	contenido parcial
300	Múltiples elecciones
301	movido permanentemente
302	movido temporalmente
303	ver otros
304	no modificado
305	usar proxy
400	petición errónea
401	no autorizado
402	pago requerido
403	prohibido
404	no encontrado
405	método no permitido
406	no se puede aceptar.
407	se requiere autentificación proxy
408	límite de tiempo de la petición
409	Conflicto
410	gone (movido)
411	tamaño requerido
412	falla una precondición
413	contenido de la petición muy largo
414	URL de la petición muy largo
415	campo media type requerido
500	error interno del servidor
501	no implementado
502	puerta de enlace errónea
503	servicio no disponible
504	tiempo límite de la puerta de enlace
505	versión de protocolo HTTP no soportada

1.5.2 Cabeceras del Servidor

Después de la línea de estado el servidor envía su cabecera HTTP. Algunas de estas son las mismas cabeceras que envían los navegadores con su petición. La siguiente tabla enumera las más comunes (Tabla 4).

TABLA 4. CABECERAS MÁS COMUNES PARA SERVIDORES HTTP

Cabecera	Descripción
Content-Base	Especifica la base URL para resolver todas las URLs referentes con el documento.
Content-Length	Especifica la longitud (en bytes) del cuerpo.
Content-Type	Especifica el tipo de media del cuerpo.
Date	Especifica la fecha y la hora cuando el servidor respondió a la petición.
Etag	Especifica una entidad Tag para el recurso pedido.
Last-Modified	Especifica la fecha y la hora cuando el recurso pedido tuvo su última modificación.
Location	Especifica la nueva localización para el recurso.
Server	Especifica el nombre y versión de el servidor Web.
Set-Cookie	Especifica un par de nombres y valores que el navegador debe proporcionar para futuras peticiones.
WWW-Authenticate	Especifica el método de autorización y dominio.

Sin embargo, el protocolo HTTP solo es la comunicación entre navegadores y servidores, mostrando el estado del recurso, el tipo de servidor, la longitud del recurso, etc. Por lo que para poner en marcha el CGI hay que conocer también el contenido de los recursos y en especial el contenido de las páginas HTML.

1.6 HTML

El HTML (Hyper Text Markup Language ó Lenguaje de Marcación de Hipertexto) no es más que una aplicación del SGML (Standard Generalized Markup

Language), un sistema para definir tipos de documentos estructurados y lenguajes de marcas para representar esos mismos documentos. El término HTML se suele referir a ambas cosas, tanto al tipo de documento como al lenguaje de etiquetas.

1.6.1 ¿Qué es el HTML?

El HTML es un lenguaje de marcas de texto utilizado normalmente en la WWW (World Wide Web). Fue creado en 1986 por el físico nuclear Tim Berners-Lee; el cual tomó dos herramientas preexistentes: El concepto de Hipertexto (conocido también como link o ancla) el cual permite conectar dos elementos entre si y el SGML el cual sirve para colocar etiquetas o marcas en un texto que indique como debe verse. HTML no es propiamente un lenguaje de programación como C++, Visual Basic, etc., sino un sistema de etiquetas. HTML no utiliza un compilador, por lo tanto algún error de sintaxis que se presente en su código, el navegador Web no lo detectará y se visualizara en la forma como éste lo entienda.

El entorno para trabajar HTML es simplemente un procesador de texto, como alguno incluido en los sistemas operativos Windows, UNIX o MACINTOSH. El conjunto de etiquetas que se creen se deben guardar en un archivo con la extensión .htm o .html

Estos documentos pueden ser mostrados por los navegadores o "browsers" de páginas Web en Internet, como Netscape Navigator, Mosaic, Opera y Microsoft Internet Explorer.

También existe el HTML Dinámico (DHTML) que es una mejora de Microsoft de la versión 4.0 de HTML que le permite crear efectos especiales como por ejemplo: Texto que vuela desde la página palabra por palabra o efectos de transición al estilo de anuncio publicitario giratorio entre página y página.

Orígenes del HTML

- 1986. Publicación de la ISO 8879 que presenta el Standard General Markup Language; origen del HTML.

- 1989. Tim Berners-Lee en el Centro Europeo de Investigaciones Nucleares presenta su artículo Information Management: A Proposal dedicándose de lleno al desarrollo de un sistema que permitiera el acceso en línea de manera

uniforme a la información disponible en varios recursos distintos y que pudiese funcionar en máquinas que conectadas por redes basadas en TCP/IP.

- 1990-1991. Tim Berners-Lee define el HTML como un subconjunto de SGML, que más tarde se llamará nivel 0; soporta encabezados, listas y anclas. Se crea el nombre World Wide Web.

- 1991. Tim Berners-Lee introduce el primer visor de HTML: LineMode, que trabaja en modo texto y sólo en plataformas UNIX. El Centro Europeo de Investigaciones Nucleares realiza la apertura del primer sitio con acceso público de World Wide Web (http://info.cern.ch).

- 1992. Dan Connelly produce la primera Definición de Tipo de Documento (DTD) para el lenguaje llamado HTML 1.0, agregando a la definición original atributos para modificar el estilo físico del texto. Se distribuye Viola, primer visor gráfico de Web y disponible sólo para X.11.

- 1993. Un nuevo visor que soporta un mayor nivel: Lynx, es producido por la Universidad de Kansas, si bien lee sólo texto. Aparece Mosaic, desarrollado por el Centro Nacional para Aplicaciones de Supercomputadoras; es el primer navegador Web en entorno gráfico que se hace disponible para computadoras personales lo que lo hace inmediatamente popular. A finales del año, comienzan a aparecer los primeros artículos sobre WWW en diarios y revistas de circulación masiva. Tim Berners-Lee utiliza el trabajo del año anterior de Connelly para presentar el borrador de la primera norma (RFC - Recommendation for Comments) de HTML para Internet.

- 1994. La Universidad Técnica de Graz desarrolla un servidor y clientes con mayores prestaciones para HTML: Hyper-G, que no tiene gran éxito. Cello, es el primer navegador de HTML que no requiere TCP/IP presentado por la Escuela de Leyes de la Universidad de Cornell. Dan Connolly y Karen Olson Muldrow redefinen el HTML para el nivel 2.0 que ahora soporta formularios. Un grupo de programadores que desarrollaran el Mosaic producen un nuevo navegador de World Wide Web: Netscape (también conocido como Mozilla), que tiene una amplia aceptación entre los usuarios pero soporta elementos de

programación que equivalen a una degeneración del HTML (tamaños de letra, fondos). Se define un equivalente para los modelos en tres dimensiones del HTML: el VRML (Virtual Reality Modeling Language), que permite moverse dentro de los ambientes definidos. En este mismo año se realizan la Primera y Segunda conferencias internacionales de WWW en Ginebra y Chicago, respectivamente. Se crea la W3 Organization.

- 1995. Dave S. Raggett (Hewlett-Packard, Inglaterra) comienza a compilar la normativa del nuevo nivel del lenguaje: el HTML 3.0, cuya principal novedad es el soporte de tablas. Microsoft produce su primer visor de Internet, el cual también utiliza elementos de HTML degenerados. Una nueva versión de Netscape, Navigator 2.0 agrega soporte de marcos. Sun Microsystems produce el primer navegador de World Wide Web con soporte de un lenguaje de programación: HotJava. Se celebran la Tercera y Cuarta conferencias internacionales de WWW en Boston y Darmstadt, respectivamente; y la conferencia de WWW para Asia y el Pacífico en Wagga-Wagga.

- 1996. Netscape Communications y Microsoft presentan las nuevas versiones de sus navegadores que soportan gran parte del nivel de HTML 3.0. Aparecen navegadores no comerciales que implementan la norma completa de HTML 3.0. Se formaliza un nuevo nivel para la modelación en tres dimensiones: VRML 3.0, que permite interactuar con los objetos definidos. Se celebra la Quinta conferencia internacional de WWW en Rocquencourt.

- 1997. D. Raggett presenta en enero la versión normalizada del 3.2. En julio, aparece la versión 4.0, experimental.

- 1998. Aparece la versión HTML 4.0.

1.6.2 Estructura de los documentos de HTML

Existen algunos programas especializados para crear documentos HTML como por ejemplo: Microsoft Front Page o Macromedia Dreamweaver. Otra forma de diseñar un archivo .html es trabajar con un editor de texto común como ya se había mencionado

anteriormente; ya que este sencillo programa cumple con un requisito mínimo que es la posibilidad de trabajar con las etiquetas de este lenguaje.

La unidad fundamental en el lenguaje HTML es el "*tag*" o la "*etiqueta*". Cada estructura de texto se encerrará entre una marca de inicio y otra de fin.

En el HTML las marcas vienen delimitadas con los signos menor que (<) y mayor que (>). De este modo el navegador sabe que debe interpretar el código comprendido entre estos símbolos.

En general, las etiquetas en HTML parecen por pares: una de comienzo y otra de terminación, en la que ésta última comienza con un signo " / " o slash:

<HTML> Mi página Web </HTML>

El olvido de este slash puede producir efectos imprevisibles al no cancelar el efecto de la etiqueta de apertura. Sin embargo, existen algunos que no llevan etiqueta de terminación.

*<HR>
 <P>*

Numerosos tags también llevan parámetros asociados:

<BODY bgcolor="#FFFFFF"> </BODY>

Todos los documentos HTML bien escritos comparten una estructura en común. Un documento HTML empieza con la etiqueta <HTML>, que es la que encerrará el documento actual. Contiene dos secciones primordiales: La cabecera y el cuerpo encerradas respectivamente por los elementos <HEAD> cabecera y <BODY> cuerpo.

La cabecera puede contener información y siempre contiene el título del documento encerrado por el elemento <TITLE>.

En el cuerpo se encuentra todo el contenido del documento, ya sea: texto, imágenes, sonidos, hipervínculos, etc.

Ejemplo:

```
<HTML>
      <HEAD>
     <TITLE> Título de mi página de Internet </TITLE>
      </HEAD>
      <BODY>
      <P>Aquí se introduce el contenido del documento</P>
      </BODY>
</HTML>
```

También se puede colocar código: java, javascript, visualbasic script, PHP, etc., dentro del documento HTML; para una mayor referencia consultese información acerca de estos lenguajes scripts y su relación con el HTML.

Como no es el objetivo de esta tesina el dar un curso de HTML, sólo se ocupará el tema de formularios; estos tienen una mayor relación con los CGI's. Para mas información acerca del lenguaje HTML, se puede consultar el anexo A de esta tesina y si se desea una mayor referencia existen libros dedicados a este lenguaje, como se amplían en las referencias bibliográficas.

1.6.3 Formularios HTML

Los formularios HTML al igual que sus homólogos en papel, se utilizan principalmente para reunir información del usuario. Los formularios ofrecen a los visitantes de un sitio Web una forma simple de proporcionar información útil, permitiendo intercambiar información y opiniones acerca del sitio. Pero también ofrecen infinitas posibilidades como una consulta a una base de datos, un envió de correo electrónico, etc.

El HTML utilizado para construir páginas de formularios es el mismo que se utiliza para construir cualquier otra página. En dichas páginas se añaden tags o etiquetas específicas para formularios de la misma forma en que se utilizan determinados tags para tablas o marcos (frames).

Estos tags específicos para formularios permiten añadir todo tipo de características a una página con uno o varios formularios con botones, casillas de texto, etc.

El formulario <FORM>

Para construir un formulario se utiliza el tag <FORM>. Y entre los tags <FORM> de apertura y cierre están los elementos del formulario junto con las tags de la estructura que constituyen la composición del formulario. Se pueden colocar en una página tantos formularios como necesite, sin embargo, no se pueden anidar, es decir, no puede situar uno dentro de otro.

El tag de apertura del elemento del formulario, por lo general, incluye dos atributos: METHOD y ACTION. El atributo METHOD puede ser GET o POST lo que determina cómo se envían los datos del formulario al script para procesarlos.

Sintaxis del tag

<FORM ACTION="action" METHOD="method" ENCTYPE="enctype" ACCEPT-CHARSET="charset">

Donde ACTION especifica un agente de proceso de formularios; METHOD el método http que se utilizará para presentar el conjunto de los datos del formulario (Tabla 1); ENCTYPE el tipo de contenido utilizado para presentar el formulario al servidor o en cualquier otro lugar; y CHARSET la lista de codificaciones de caracteres de los datos de entrada.

El atributo ACTION es un puntero al script que procesa el formulario en el lado del servidor. La acción se puede indicar por medio de una ruta relativa o completa hacia el servidor. Por ejemplo:

```
<HTML>
<HEAD>
<TITLE>Formulario de retroalimentación</TITLE>
</HEAD>
<BODY>
<H1>Formulario de retroalimentación mi empresa<H1>
<HR>
<FORM METHOD="post" ACTION="/cgi-bin/feedback.cgi">
...
</FORM>
```

Campos de texto

El tag base para definir cada elemento en un formulario de línea es <INPUT>, que se utiliza para añadir botones, imágenes, casillas de verificación, botones de radio, contraseñas y campos de texto.

Sintaxis del tag

<INPUT TYPE="type" NAME="name" VALUE="value" CHECKED SIZE="size" MAXLENGTH="maxlength">

Donde TYPE especifica text, password, hidden, checkbox, radio, submit, reset, button, file o image; NAME la etiqueta del nombre; VALUE el valor inicial; SIZE proporciona al agente del usuario la anchura inicial del control, y MAXLENGTH el número máximo de caracteres que puede introducir el usuario.

El campo password se diseñó especialmente para introducir contraseñas. Estos campos de texto son muy similares a los campos de texto normales excepto que los caracteres introducidos aparecen como asteriscos para enmascarar la entrada. Cualquier cosa que se teclee en un campo password no será visible y se puede utilizar para proteger contraseñas, números de cuenta ó cualquier otra información sensible. Aunque la pantalla enmascara los caracteres, el servidor recibe el texto original.

<INPUT TYPE="password" NAME="pswd">

Otro tipo de campo de texto es con la opción hidden. Al utilizar este campo no parece nada en la pantalla, sin embargo proporciona una forma de enviar información adicional al script CGI que el usuario no puede modificar. Por ejemplo:

<INPUT TYPE="hidden" NAME="tipo_empleado" VALUE="De planta">

Botones de radio

Los botones de radio indican una lista de temas de los que sólo se puede elegir uno, lo mismo que una pregunta que tiene una única respuesta. Si se elige un botón de radio de la lista, dejan de elegirse el resto de los botones de radio de dicha lista. Sólo puede hacer una selección cuando utiliza botones de radio.

Al utilizar el atributo TYPE=radio del tag <INPUT> se crean botones de radio. Se pueden crear grupos de ellos utilizando el mismo nombre para cada botón del grupo. Sin embargo, cada uno de ellos dentro del grupo debe tener un atributo VALUE único.

Sintaxis del tag

<INPUT TYPE="radio" NAME="name" VALUE="value" CHECKED>

Donde NAME es el nombre de grupo de este botón; VALUE es el valor del botón que se pasa al script CGI, y CHECKED se presenta para los botones preseleccionados, es decir, para especificar un valor por defecto en un grupo de botones.

Cuando finalmente se ha enviado el formulario al servidor, el navegador pasa un par nombre-valor para el grupo de botones completo al script CGI. Dicho par incluye el atributo NAME para cada grupo de botones de radio y el atributo VALUE del botón que se ha elegido.

Casillas de verificación

Cuando sea necesario selecciones de múltiples temas, se necesita utilizar casillas de verificación. Hay algunas preguntas que no tienen una sola respuesta, y ahí es donde entran en juego las casillas de verificación. Estas, como los botones de radio, se colocan en grupos y cada una puede verificarse o no. Por defecto, no se verifican.

Sintaxis del tag

<INPUT TYPE="checkbox" NAME="name" VALUE="value" CHECKED>

Donde NAME es el nombre del grupo de esta casilla; VALUE es el valor de la casilla que se pasa al script CGI, y CHECKED se elige para preseleccionar.

Las casillas de verificación también se pueden preseleccionar utilizando el atributo CHECKED en el tag <INPUT> igual que los botones de radio.

Listas despegables

Las selecciones permiten elegir uno o más temas de un menú o de una lista por la que se puede desplazar. Funcionalmente son similares a los botones de radio o las casillas de verificación pero aparecen de forma diferente en la pantalla.

Las selecciones se crean utilizando el tag <SELECT>, y las opciones individuales en la selección se indican con el tag <OPTION>. El tag <SELECT> también contiene un atributo NAME para mantener el valor cuando se envía el formulario.

Sintaxis del tag

<SELECT NAME="name" SIZE="size" MULTIPLE SELECTED>

Donde NAME es el nombre del elemento; SIZE especifica el número de filas en la lista que deberían ser visibles a la vez; MULTIPLE es un atributo booleano que permite selecciones múltiples; y SELECTED está presente como valor inicial. Y al igual que los botones y las casillas de verificación se puede fijar un valor inicial utilizando el atributo SELECTED.

Por defecto las selecciones funcionan como botones de radio, es decir, sólo se puede elegir un tema a la vez. Se puede modificar el comportamiento de las selecciones para permitir que se elijan opciones múltiples utilizando el atributo MULTIPLE que forma parte del tag *<SELECT>*.

Introducción de texto en un formulario

Cuando una simple línea en el campo de texto (<INPUT TYPE="text">) no sea espacio suficiente en el formulario para teclear la información se puede utilizar un campo de entrada de área de texto.

Al igual que en otros tags, <TEXTAREA> debe utilizarse con el tag de cierre </TEXTAREA>. El texto que se coloque entre los tags de comienzo y cierre será el texto que por defecto se situará en el campo de texto.

Sintaxis de tag

<TEXTAREA NAME="name" ROWS="rows" COLS="cols" WRAP="wrap">

Donde NAME es el nombre del elemento; ROWS especifica el número de líneas de texto visibles; COLS la anchura visible en las anchuras medias de caracteres; y WRAP el formato de la pantalla. La Tabla 5 muestra las opciones del atributo WRAP.

TABLA 5. OPCIONES DEL ATRIBUTO WRAP

Atributo	Descripción
WRAP=off	El texto por defecto estará en una línea desplazándose a la derecha hasta que se pulse Intro para comenzar una nueva línea.
WRAP=soft	Origina que el texto se envuelva automáticamente en la ventana del navegador, pero se envía al servidor como si sólo se tratase de una línea.
WRAP=hard	Origina que el texto se envuelva automáticamente en la ventana del navegador. También se envía al servidor con caracteres de nueva línea en todos los puntos dónde se envolvió el texto.

Botones de envío y reposición

Lo ultimo que hay que hacer para utilizar esta información válida, es pasar los datos introducidos en los formularios al script CGI. Para hacerlo se necesita utilizar un botón de envío. Aunque está incluido con los tipos de entrada, el tipo SUBMIT da como resultado un botón que cuando se pulsa envía el contenido del formulario al script CGI especificado en el atributo *ACTION* del tag <FORM>.

Por defecto, la etiqueta en un botón de envío es simplemente "Submit". Puede personalizarla utilizando el atributo VALUE para el botón de envío.

<INPUT TYPE="submit" VALUE="Enviar Formulario">

En segundo término, todo formulario debería tener una forma de borrar todas las selecciones y dejarlo en su estado original. Esto se hace utilizando el atributo Type=RESET del tag <INPUT>.

Al igual que con el atributo SUBMIT, RESET también creará un botón en el formulario. Si no se ha definido ningún valor por defecto, se borra cualquier contenido del elemento. Como en el botón de envío, también puede cambiar el nombre del botón de reposición utilizando el atributo VALUE de ese tag.

La primera versión de PERL que llegó a ser suficientemente conocida fue la versión 4, dada a conocer al mundo por el libro del camello. Esta versión se desarrolló entre 1991 y 1993, y coincidió con la popularidad del PERL como lenguaje para programación de servidores de Internet; aunque originalmente se había diseñado como lenguaje para administración de sistemas.

La versión 5 apareció hasta octubre de 1994, y ha sido tan popular que todavía se usa. Introdujo características que hacen a PERL tan fácil de programar, incluyendo los módulos, las facilidades para programación dirigida a objetos, referencias y mucha mejor documentación.

A partir de la versión 5.6, Perl sufrió una nueva transformación. Se incluye soporte pleno de caracteres internacionales, hebras y un mejor intérprete. Se institucionaliza un sistema de patch pumpkin, o encargado de cada nueva versión, él decide qué va a entrar de nuevo y qué no, sustituyendo a Larry Wall. La empresa comercial ActiveState (que ya participaba activamente en su desarrollo) comienza a controlar más de cerca a PERL, y a la vez, a crear herramientas más potentes (y comerciales) para el desarrollo.

A partir del año 2000, se empieza a discutir sobre la nueva versión, la 6, que será un gran salto sobre la versión anterior, pero todavía no está muy claro qué es lo que va a ser. Aparte de más rápida y más flexible, todavía no se ha comenzado su desarrollo.

El CGI es un método para la transmisión de información hacia un interprete instalado en el servidor (en este caso el interprete es Perl). Su función principal es la de añadir una mayor interacción a los documentos que por medio del HTML se presentan de forma estática.

Aunque es posible escribir un programa CGI en cualquier lenguaje, Perl se ha convertido en el sinónimo de la programación CGI. Existen diversas razones, aunque las más importantes son:

- Manejo de Sockets. Crea programas que establecen una interfaz fluida con los protocolos de internet.

- Comparación de Patrones. Ideal para el manejo de datos y textos de búsqueda.

- Manejo flexible de textos. La forma que en Perl maneja las cadenas, en términos de asignación y liberación de memoria es muy sencilla.

La ventaja de un lenguaje interpretado en aplicaciones CGI es la simplicidad en su desarrollo, depuración y revisión.

2.1 LENGUAJE DE PROGRAMACIÓN PERL

"Perl se creo para resolver una necesidad, no para coincidir con los ideales de la ciencia en cómputo. Evolucionó de ser un simple arreglo inteligente de los recursos del sistema para convertirse en un completo y moderno lenguaje de programación". [MEDINETS, 1997, PÁG. 15]

2.1.1 Variables

Perl, al igual que otros lenguajes de programación utiliza variables para llevar un registro del uso de la memoria de la computadora. Cada vez que se desee almacenar un nuevo dato de información se deberá asignarlo a una variable. La Tabla 6 muestra los tres tipos de variables:

TABLA 6. TIPOS DE VARIABLES

Tipo de variable	Descripción
Escalares	Contienen el valor de un número o de una cadena a la vez. Los nombres de variables escalares comienzan siempre con " $ ".
Arreglos	Contienen una lista de valores. Los valores pueden ser números, letras, o incluso otro arreglo. Los nombres de las variables de arreglo comienzan siempre con " @ ".
Arreglos asociativos	Usan cualquier valor como un índice dentro de un arreglo. Los nombres de las variables de arreglo asociativo comienzan con " % ".

Los caracteres de inicio de las variables, ayudan a entender y comprender como se están utilizando, si son escalares ($), arreglos (@) ó arreglos asociativos (%).

Cabe mencionar que las variables en Perl son sensibles al uso de mayúsculas y minúsculas. Además la longitud del nombre de variables no tiene límite, así que se pueden utilizar variables con nombres largos.

Variables escalares

"Una variable escalar puede almacenar un único valor ó una cadena de texto". [BARKAKATI, 1998, PÁG 63]. Por ejemplo:

```
$numero_de_paginas = 5;
$nombre_del_empleado = "Juan Carlos";
```

Se puede almacenar cualquier valor en una variable escalar. Los valores más comunes son las cadenas de texto, los números enteros y los reales, etc. Por ejemplo:

```
$paginas=300;
$error_maximo=1.57e-6;  #1.57 x 10^-6
$titulo_del_libro="El quijote de la mancha";
```

El símbolo # sirve para escribir comentarios dentro de los programas en Perl.

El uso de las comillas dobles (" ... ") en las cadenas de texto tiene el siguiente fin:

- Las variables escalares y de arreglos encerradas entre comillas dobles se sustituyen por sus valores.

```
$precio=300;
print "El precio del libro del Quijote es $precio";
```

El precio del libro del Quijote es 300

- En cadenas encerradas entre comillas dobles, los caracteres que tengan como prefijo una barra invertida se sustituyen por un carácter individual especial.

```
$precio=300;
print "El precio del libro del Quijote es \$$precio\n";
```

El precio del libro del Quijote es $300

Como se observa en los ejemplos anteriores, el uso de la barra invertida seguido de uno o más caracteres se conocen el Perl como "*Secuencias de Escape*", y se usan dentro de las cadenas de comillas doble. A continuación (Tabla 7) se enumeran algunas secuencias de escape.

TABLA 7. SECUENCIAS DE ESCAPE

Secuencia de escape	Interpretación
\a	Provoca un bip por el altavoz de la computadora.
\b	Se desplaza hacia atrás en un espacio.
\cn	Inserta un símbolo Ctrl.+N (donde N es cualquier carácter).
\e	Visualiza un carácter de escape.
\f	Desplaza hasta el comienzo de la siguiente página.
\l	Convierte la siguiente letra en minúscula.
\n	Desplaza hasta una nueva línea (línea nueva).
\r	Desplaza al comienzo de la línea actual (retorno de carro).
\t	Desplaza hasta la siguiente posición de tabulado (tabulador).
\u	Convierte la siguiente letra en mayúscula.
\L	Convierte todos los caracteres posteriores (Hasta el próximo \E) en minúsculas).
\U	Convierte todos los caracteres posteriores (Hasta el próximo \E) en mayúsculas).
\E	Fin de la secuencia de escape \L y \U
\"	Visualiza una comilla.
\$	Visualiza un signo de dólar.
\\	Visualiza una barra invertida

El uso de comillas simples muestra en pantalla la cadena de texto tal y como es, sin ninguna atención especial a las secuencias de escape. Para almacenar y visualizar estas cadenas, debe encerrarse la cadena entre comillas simples (' ... ').

```
$precio=300;
print 'El precio del libro del Quijote es \$$precio\n';
```

```
El precio del libro del Quijote es \$$precio\n
```

Perl, entonces visualiza la cadena de comillas simple al pie de la letra sin reemplazar los nombres de las variables por sus correspondientes valores.

Variables de Arreglo

Cuando se necesita almacenar varios valores en un programa en Perl, se utilizan estas variables. Estas variables trabajan como un vector. Visto de otra manera una variable de arreglo es una colección de variables escalares.

```
@alumnos = ("Juan Carlos", "Marisol", "Ruth", "Luis", "Silvestre");
```

Perl es muy accesible, por eso no todos los valores de las variables de arreglo deben de ser del mismo tipo, se puede almacenar cadenas o números. Sin embargo, normalmente se utilizan para almacenar un mismo tipo.

```
@valores = ("Juan",5,100,"Carlos");
```

Cada elemento en una variable de arreglo es una variable escalar, y se puede acceder a estos conociendo su posición. La posición del elemento se conoce como el índice: el primer elemento está en el índice 0, el segundo en el 1, y así sucesivamente.

```
@alumnos = ("Juan Carlos", "Marisol", "Ruth", "Luis", "Silvestre");
print "El primer alumno es $alumnos[0]\n";
```

```
El primer alumno es Juan Carlos
```

Cuando se necesita acceder a uno de los valores de las variables de arreglo, se tiene que escribir la variable como si fuera una variable escalar, ya que cada elemento en una variable de arreglo es una variable escalar y las variables escalares comienzan con el signo "$".

Perl también puede manejar los índices negativos accesando a los elementos en orden inverso.

```
@alumnos = ("Juan Carlos", "Marisol", "Ruth", "Luis", "Silvestre");
print "El último alumno es $alumnos[-1]\n";
```

El último alumno es Silvestre

Las variables de arreglo pueden también almacenar otras variables de arreglo, por ejemplo:

```
@alumnos = ("Juan Carlos", "Marisol", "Ruth", "Luis", "Silvestre");
@calificaciones=(9,10,9,8,9);
@grupo=(@alumnos,@calificaciones);
```

Variables de arreglo asociativo

"En los arreglos asociativos se puede emplear cualquier tipo de dato escalar como índice. Los nombres de los arreglos asociativos comienzan con el carácter "%" ". [MEDINETS, 1997, PÁG 39].

Los arreglos asociativos son llamados también hashes por la forma en cómo se almacenan en la memoria los elementos de un arreglo.

Las variables de arreglo asociativo están formadas por pares de clave-valor y para acceder a un elemento de este, se necesita la clave, por lo que Perl tiene una función llamada keys que toma el nombre del arreglo asociativo como argumento y devuelve un arreglo con todas las claves. Por ejemplo:

```
%alumnos = ("Juan Carlos"    => 9,
            "Marisol"        => 10,
            "Ruth"           => 9,
            "Luis"           => 8,
            "Silvestre"      => 9);
@grupo=keys(%alumnos);
print "@grupo\n";
```

Juan Carlos Marisol Ruth Luis Silvestre

Existe también una función análoga a la función keys llamada values, que devuelve un arreglo con todos valores de los elementos del hash.

```
%alumnos = ("Juan Carlos"     => 9,
            "Marisol"         => 10,
            "Ruth"            => 9,
            "Luis"            => 8,
            "Silvestre"       => 9);
@grupo=values(%alumnos);
print "@grupo\n";
```

```
9 10 9 8 9
```

Se pueden asignar valores a elementos individuales de un hash mediante el uso de las llaves ({}) alrededor de la llave índice.

```
%alumnos = ("Juan Carlos"     => 9,
            "Marisol"         => 10,
            "Ruth"            => 9,
            "Luis"            => 8,
            "Silvestre"       => 9);
$alumnos{"Fernando"} = 7;

print "La calificación más alta fue: " . $alumnos{"Marisol"} . "\n";
print "La calificación más baja fue: " . $alumnos{"Fernando"} . "\n";
```

```
La calificación más alta fue: 10
La calificación más baja fue: 7
```

Se utilizó en el ejemplo anterior el uso del punto en . *$alumnos{"Marisol"}* . ; Este es un operador de concatenación y funciona para unir dos cadenas.

Variables especiales

Perl posee una gran cantidad de variables especiales que contienen información útil que se podría necesitar en un programa. En la siguiente tabla (Tabla 8) se muestra algunas de estas variables.

TABLA 8. VARIABLES ESPECIALES

Nombre de la variable	Descripción
@ARGV	Arreglo de cadenas que contienen los argumentos de las líneas de comando utilizados para iniciar el programa en Perl
$ENV	Nombre de la variable de arreglo asociativo que almacena variables de ambiente. Se puede acceder a este arreglo usando el nombre de la variable de ambiente como clave.
$_	Argumento por defecto para varias funciones. Si existe una función Perl que se utiliza sin ningún argumento, la función obtiene su argumento por medio de la variable $_.
@_	Lista de argumentos que se transmite a una subrutina.

Para ver una lista completa de variables especiales existe un capítulo dedicado a este tema en el libro Perl 5 a través de ejemplos (consultese referencias bibliográficas).

2.1.2 Operadores, condicionales, bucles y subrutinas

Operadores

En un lenguaje de programación, los operadores indican a la computadora que acciones realizar. Perl tiene más operadores que la mayoría de los lenguajes.

"Los operadores son instrucciones que se dan a la computadora de modo que pueda efectuar alguna tarea u operación." [MEDINETS, 1997, PÁG 45]

- *Operadores aritméticos*. Perl maneja los siguientes

TABLA 9. OPERADORES ARITMÉTICOS

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
**	Exponenciación
/	División
%	Resto (Modulo)
.	Concatenación de dos cadenas
x	Repetición de caracteres

Entre los operadores aritméticos también existe un operador de *autoincremento (++)* y *autodecremento (--)*. Estos operadores son unarios y se realiza el incremento o decremento de la variable que se le aplica. Además de la acción de modificar la variable devuelven el valor de la variable.

El operador de incremento o decremento puede ir delante o detrás de la variable, teniendo diferente significado. Si el operador ++ se sitúa después de la variable se denomina *postincremento*, haciendo que primero se tome el valor y después se incremente la variable. Ejemplo:

```
$n = $k++;      # el valor de k se asigna a n y después se incrementa k
```

Y si el operador ++ se sitúa después de la variable se denomina *preincremento* y hace que primero se incremente la variable y después se tome el valor. Ejemplo:

```
$n = ++$k;      # primero se incrementa k y luego se asigna a n
```

Finalmente, con el operador de decremento se actúa de forma análoga obteniendo así el *predecremento* y el *postdecremento*. Los operadores se muestran en la tabla 10.

TABLA 10. OPERADORES DE INCREMENTO Y DECREMENTO

Operador	Descripción
+op	Incrementa la variable op
-op	Decrementa la variable op
op++	Postincremento de la variable op
++op	Preincremento de la variable op
op--	Postincremento de la variable op
--op	Predecremento de la variable op

o *Operadores relacionales.* Perl distingue dos tipos de operadores relacionales: los operadores específicos a valores numéricos y los propios de las cadenas de caracteres.

TABLA 11. OPERADORES RELACIONALES

Numérico	De cadena	Operador Relacional
==	eq	Igualdad
!=	ne	Diferencia
<	lt	Menor que
>	gt	Mayor que
<=	le	Menor o igual que
=>	ge	Mayor o igual que

Operador cmp

Este operador es utilizado para comparar caracteres, de manera que, retorna 0 si los caracteres comparados son iguales, 1 si la cadena de la derecha se encuentra al comienzo de la de la izquierda, y -1 en el caso contrario. Por ejemplo.

```
'one' cmp 'one'        # devuelve 0
'one dog' cmp 'one'    # devuelve 1
'dog one' cmp 'one'    # devuelve -1
'two' cmp 'one'        # devuelve -1
```

Operador <=>

Este operador se utiliza para comparar valores numéricos, retornando 0 cuando son iguales, 1 cuando el término de la derecha es menor que el de la izquierda y -1 en el caso contrario.

Operador =~

Este operador es usado en las expresiones regulares para indicar la presencia de un patrón de comparación dentro de una variable que contiene una cadena de caracteres.

- *Operadores lógicos.* Estos operadores son el resultado de las expresiones de los operadores relacionales.

TABLA 12. OPERADORES LÓGICOS

Operador	Descripción
op_1 && op_2	AND lógico de los dos operadores
op_1 || op_2	OR lógico de los dos operadores
!op_1	NOT lógico del operando

- *Operador de selección.* Es un operador triario que requiere una condición y dos expresiones.

Se utiliza para ejecutar una expresión u otra dependiendo de la condición. Su formato es el siguiente:

Condición ? Exp1 : Exp2

Si se cumple la condición se evalúa y devuelve la expresión Exp1 si no la Exp2. Por ejemplo:

```
n = (x<y ? 40:z+1);      # si x<y entonces n=40, si no n=z+1
```

- *Operadores a nivel bit.* Los operadores a nivel de bit operan independientemente sobre cada uno de los bits de un valor.

TABLA 13. OPERADORES A NIVEL BIT

Operador	Descripción
op1 & op2	Operador AND
op1 | op2	Operador OR
op1 ^ op2	Operador OR EXCLUSIVO
-op1	Operador COMPLEMENTO
op1 >> op2	Operador DESPLAZAMIENTO A LA DERECHA
op1 << op2	Operador DESPLAZAMIENTO A LA IZQUIERDA

- *Operadores de asignación.* Es un operador que devuelve la variable modificada.

TABLA 14. OPERADORES DE ASIGNACIÓN

Operador	Descripción
var = op1;	Asigna el valor de op1 a var
var += op1;	Asigna el valor de var + op1 a var
var -= op1;	Asigna el valor de var - op1 a var
var *= op1;	Asigna el valor de var * op1 a var
var /= op1;	Asigna el valor de var / op1 a var
var %= op1;	Asigna el valor de var % op1 a var
var .= op1;	Asigna el valor de var . op1 a var
var **= op1;	Asigna el valor de var ** op1 a var
var x= op1;	Asigna el valor de var x op1 a var
var <<= op1;	Asigna el valor de var << op1 a var
var >>= op1;	Asigna el valor de var >> op1 a var
var &= op1;	Asigna el valor de var & op1 a var
var |= op1;	Asigna el valor de var | op1 a var
var ^= op1;	Asigna el valor de var ^ op1 a var

Condicionales

Enunciado if

"Los enunciados de decisión usan la palabra clave if para ejecutar un bloque de enunciados con base en la evaluación de una expresión o para elegir entre ejecutar uno de dos bloques de enunciados con base en dicha evaluación." [MEDINETS, 1997, PÁG 119]

Sintaxis del enunciado if

```
if (CONDICION){
        #Bloque de código a ejecutar si la condición es cierta.
} else {
        #Bloque de código a ejecutar si la condición es falsa.
}
```

En algunas ocasiones esto no es suficiente para comprobar más de dos alternativas, por lo que se utiliza el enunciado if ... elsif.

Sintaxis del enunciado if ... elsif

```
if (CONDICION_UNO){
        #Bloque de código a ejecutar si la condición uno es cierta.
} elsif (CONDICION_DOS){
        #Bloque de código a ejecutar si la condición dos es cierta.
} else {
        #Bloque de código a ejecutar si las demás condiciones son falsas.
}
```

Enunciado unless

En algunas ocasiones se requiere ejecutar un bloque de sentencias si una condición es falsa. *"El enunciado unless tiene la misma estructura que if, hasta el uso de las cláusulas elsif y else. La diferencia estriba en que unless ejecuta su bloque de sentencias solamente si la condición es falsa." [BARKAKATI, 1998, PÁG 127]*

Sintaxis del enunciado unless

```
unless (CONDICION){
        #Bloque de código a ejecutar si la condición es falsa.
}
```

Bucles

Para ejecutar un bloque de sentencias repetidamente hasta que alguna condición llegue a ser falsa, esta operación se llama bucle o ciclo.

Enunciado while

El enunciado while, evalúa repetidamente la expresión mientras que la condición sea verdadera y cuando la condición se hace falsa, el enunciado termina.

Sintaxis del enunciado while

```
while (CONDICION){
        #Bloque de código a ejecutar mientras la condición sea verdadera.
}
```

Enunciado until

El enunciado until es igual que while, pero repite un bloque de sentencias hasta que una determinada condición se haga verdadera, en otras palabras, la sentencia until ejecuta el bloque mientras la condición es falsa.

Sintaxis del enunciado until

```
until (CONDICION){
        #Bloque de código a ejecutar mientras la condición sea falsa.
}
```

Enunciado for

El enunciado for es otra de las maneras en que Perl puede ejecutar un bloque de sentencias el número de veces que se especifique, basada en el valor de una expresión.

Sintaxis del enunciado for

```
for (expr_1; expr_2; expr_3)
{
        #Bloque de código a ejecutar
}
```

En donde expr_1 se evalúa una vez al inicio del bucle, y el bloque de código se ejecuta mientras expr_2 sea verdadera. La tercera expresión, expr_3 se evalúa después de cada ejecución del bloque de código. Se puede omitir alguna de estas expresiones, pero se deben incluir los puntos y comas.

Enunciado foreach

Para el manejo de arreglos, Perl proporciona el enunciado foreach, es decir, que ejecuta un bloque de código con cada elemento del arreglo.

Sintaxis del enunciado for

```
foreach $variable (@arreglo)
{
        #Bloque de código a ejecutar
}
```

Después de conocer los enunciados condicionales y ciclos se construirá un programa sencillo, el juego de serpientes y escaleras:

```
print "\n\nESTE PROGRAMA SIMULA:\n\n";
print "****** EL JUEGO DE SERPIENTES Y ESCALERAS *****\n\n";
print "Escribe el numero de jugadores ";
$personas=<STDIN>;
@jugador = ();
%escaleras=(11=>39, 17=>67, 19=>45, 21=>56, 26=>50, 43=>84, 52=>76, 70=>92, 74=>100);
%serpientes=(18=>6, 22=>2, 36=>20, 75=>30, 62=>14, 78=>49, 83=>18, 93=>40, 96=>69);
@Escalera=keys(%escaleras);
@Serpiente=keys(%serpientes);
srand(time());
```

```
do
{
 for (1..$personas)
  {
        print "\n \nEl jugador $_ le toca tirar \n";
        $dado=int(rand(5))+1;
        print "El dado cayo en $dado\n";
        $jugador[$_]=$dado+$jugador[$_];
        print "El jugador $_ esta en la casilla $jugador[$_]\n";
        foreach  $casilla(@Escalera){
          if ($jugador[$_]==$casilla)
                {
                 $jugador[$_]=$escaleras{$casilla};
                       print "El jugador $_ cayo en la escalera\n";
                       print "El jugador $_ sube a la casilla $escaleras{$casilla}\n";
                }
        }
        foreach  $casilla(@Serpiente){
        if ($jugador[$_]==$casilla)
           {
                $jugador[$_]=$serpientes{$casilla};
                print "El jugador $_ cayo en la serpiente\n";
                print "El jugador $_ baja a la casilla $serpientes{$casilla}\n";
               }
        }
        if ($jugador[$_]>100)
           {
                $m=($jugador[$_])-100;
                $jugador[$_]=(100-$m);
                print "El jugador $_ cayo mas de 100\n";
                print "Retrocede a la posicion $jugador[$_]\n";
               }
   if ($jugador[$_]==100)
               {
                print "!! FELICIDADES JUGADOR $_ caiste en la casilla 100 !! GANASTE \n";
                exit;
               }
        print "\nPresiona ENTER para el turno del siguiente jugador... \n";
        $t=<>;
       }
       print "\nPresiona ENTER para la siguiente ronda... \n";
}while(<>)
```

A veces es necesario interrumpir un ciclo o una condición antes de que haya realizado todas las iteraciones o validaciones. A continuación se muestran algunas sentencias de interrupción de condicionales y ciclos.

Enunciado goto.

La sentencia *goto label* permite cambiar el recorrido directo de las líneas de código prosiguiendo la ejecución del programa en la línea de etiqueta *label*. La etiqueta, se define colocando al final del identificador dos puntos *(:)*. Por ejemplo:

```
if ($expr ne $expr_correcta) {
goto error; #error es la etiqueta
}
...
error: print "expresión incorrecta";
```

La utilización del *goto* en Perl no es recomendable. Por que le quita al código legibilidad y aumenta la posibilidad de errores.

Enunciado last.

La sentencia *last* interrumpe la ejecución del ciclo actual y se ejecuta la instrucción que sigue al bloque. El ejemplo siguiente permite interrumpir el ciclo *while* cuando la variable *i* toma el valor 100.

```
$i = 0;
while($i < 200) {
        if($i == 100) {
            last;
        }
        $i++;
}
print "el valor de \$i es $i";
```

Cuando la sentencia tiene como argumento una etiqueta, la ejecución prosigue en la línea indicada por la etiqueta.

Enunciado next.

Interrumpe la ejecución del bloque de instrucción actual y prosigue la ejecución en la iteración siguiente. Esta instrucción no interrumpe completamente la ejecución del ciclo; la expresión que controla el ciclo se evalúa. Si el resultado de la expresión es válido, el ciclo se ejecuta de nuevo.

Cuando una sentencia tiene como argumento una etiqueta, la instrucción prosigue en la línea identificada por la etiqueta y no al principio del bloque. Por ejemplo:

```
print "Teclea \"x\" para salir;\n";
print "Si se pulsa la tecla \"s\" no se imprime.\n";
$ristra = "";
while ($ristra ne "x") {
        $ristra = ; chop($ristra);
        if ($ristra eq "s") {
        next;
        }
        print "Has escrito $ristra\n";
}
print "Salida.\n"
```

Subrutinas

Para evitar la peligrosa práctica de copiar y pegar el código a lo largo de un programa, haciendo que estos sean largos y complejos, se utilizan las subrutinas o funciones, Perl no distingue entre las funciones o subrutinas como lo hacen otros lenguajes como "C".

Perl interpreta su programa antes de ejecutarlo y no le importa donde se declaren las subrutinas, así que se pueden declarar al inicio, intermedio o final del programa.

La sintaxis de una función

```
sub nombre_de_funcion{


}
```

La llamada a la función se realiza de la siguiente manera:

```
& nombre_de_funcion;
```

La parte difícil viene cuando intervienen los parámetros. Los parámetros son valores que se pasan a la función. Estos se especifican dentro de los paréntesis que siguen inmediatamente al nombre de la función.

```
nombre_de_funcion (@variable, valor );
```

Uso del arreglo de parámetros @_

"Todos los parámetros para una función se almacenan en un arreglo denominado @_. Un efecto de ello es que se puede saber cuántos parámetros se transfirieron evaluando @_ en un contexto escalar." [MEDINETS, 1997, PÁG 83]

Ejemplo:

```
Grupo(1,2,3,4,5,6,7,8);
Grupo("A".."Z");

sub Grupo{
        $num_parametros = @_ ;
        print ("El numero de parametros es $num_parametros\n");
}
El numero de parametros es 8
El numero de parametros es 26
```

Con la variable @_ también se puede conocer cada uno de los parámetros de la función, por ejemplo $_[0] es el primer parámetro, $_[1] es el segundo parámetro y así sucesivamente.

Ejemplo

```
Grupo("A".."Z");

sub Grupo{
        $num_parametros = @_ ;
        print ("El primer parametro es $_[0]\n");
        print ("El primer parametro es $_[1]\n");
        print ("El primer parametro es $_[2]\n");

}
        El primer parametro es A
        El primer parametro es B
        El primer parametro es C
```

Todas las variables que se declaran son globales por lo que en las subrutinas se manejan variables a nivel local, es decir, las variables sólo existirán en la subrutina y tomarán un valor indefinido dentro de esta, Perl al final de la subrutina descarta la variable local y restaura el valor previo (variable global). A continuación se muestran dos ejemplos, el primero sólo toma la variable a nivel local y se declara como my variable, el segundo toma la variable a nivel global. Ejemplos:

```
Grupo("A".."Z");

sub Grupo{
 my $num_parametros;
 $num_parametros=@_;
 print ("El numero de parametros
 es $num_parametros\n");
 }

print ("El numero de parametros es
$num_parametros\n");


El numero de parametros es 26
El numero de parametros es
```

```
Grupo("A".."Z");

sub Grupo{
 $num_parametros=@_;
 print ("El numero de parametros
 es $num_parametros\n");
 }

print ("El numero de parametros es
$num_parametros\n");


El numero de parametros es 26
El numero de parametros es 26
```

Existe un segundo método para declarar variables locales a nivel subrutina, y se declara como local variable; se muestra en el siguiente ejemplo:

```
$value = "original";

tellme();
spoof();
tellme();

 sub spoof {
   local ($value) = "temporal";
   tellme();
 }

 sub tellme {
 print "El valor actual value\n";
 }

El valor actual original
El valor actual temporal
El valor actual original
```

```
$value = "original";

tellme();
spoof();
tellme();

 sub spoof {
        my ($value) = "temporal";
    tellme();
 }

 sub tellme {
 print "El valor actual value\n";
 }

El valor actual original
El valor actual original
El valor actual original
```

Al analizar estos ejemplos se concluye que cuando se usa la variable my, solo existe en la subrutina spoff y pierde su valor cuando se llama a la subrutina tellme, sin

embargo cuando se utiliza local, esta va más allá, las variables local son visibles en las llamadas a las subrutinas del bloque en donde estás variables fueron declaradas.

En la siguientes dos tablas (Tabla 15 y 16) se muestran algunas funciones (o subrutinas) integradas de Perl.

TABLA 15. FUNCIONES DE CADENA EN PERL

Funciones de cadena	Descripción
chop (CADENA) chop (ARREGLO)	Elimina el último carácter de una CADENA o de cada elemento de un ARREGLO. Retorna el último carácter eliminado
chr (NÚMERO)	Retorna el carácter representado por NÚMERO en la tabla de caracteres ASCII
crypt (CADENA1, CADENA2)	Encripta la CADENA1.
lc (CADENA)	Retorna una cadena con todas las letras de la CADENA en minúsculas.
lenght (CADENA)	Retorna la longitud de CADENA
split (PATRÓN, CADENA, LÍMITE)	Divide una cadena con base en cierto delimitador. En un contexto de arreglo, retorna una lista de las cosas que encontró. En un contexto escalar, retorna el número de cosas encontradas.
uc (CADENA)	Retorna una cadena con todas las letras de CADENA en mayúsculas.

TABLA 16. FUNCIONES DE ARREGLOS EN PERL

Funciones de arreglos	Descripción
delete (LLAVE)	Elimina la pareja clave-valor del arreglo asociativo dado.
each (ARREGLO_ASOC)	Retorna una lista de dos elementos que contiene una pareja de clave-valor del arreglo asociativo dado.
keys (ARREGLO_ASOC)	Retorna una lista que contiene todas las claves en un arreglo asociativo dado. La lista no tiene un orden en particular.
pack (CADENA, ARREGLO)	Crea una estructura binaria de los elementos del ARREGLO, usando la CADENA como guía.
pop (ARREGLO)	Retorna el último valor de un ARREGLO
push (ARREGLO1, ARREGLO2)	Agrega el contenido de ARREGLO2 a ARREGLO1.
reverse (ARRREGLO)	Cuando se usa en un contexto de arreglo, invierte los elementos de un ARREGLO dado. Cuando se usa en un contexto escalar, el arreglo es convertido a una cadena, y ésta se invierte.
sort (ARREGLO)	Retorna una lista que contiene los elementos del ARREGLO clasificados.

2.1.3 Entrada y Salida

Los programas de Perl se diseñan generalmente para leer información de la entrada estándar y enviarla a la salida estándar. Por default, la entrada estándar es el teclado y la salida estándar es la pantalla. Sin embargo los sistemas operativos como UNIX o Windows permiten redireccionar la entrada y la salida de tal manera que se pueden recibir o enviar datos desde o hacia un archivo. La entrada y salida estándar se utilizan en los scripts CGI.

STDIN (Entrada)

En los scripts Perl, para referenciar un archivo se utiliza el término de identificador de archivo. Para obtener un nombre válido de identificador de archivo es necesario usar la función OPEN y un nombre de archivo. Aunque por defecto, Perl define y abre automáticamente STDIN y STDOUT como identificador de archivo para la entrada y salida estándar. Así que STDIN toma por default los valores que se introduzcan por el teclado y STDOUT los muestra en pantalla con la ayuda de la función print.

"Existen dos formas de redireccionar STDIN y STDOUT:

1. *Usando operadores de redireccionamiento en la línea de comandos: desde la línea de comandos, puede enlazar STDIN y STDOUT a un archivo.*

2. *Usando la salida de otro programa como STDIN: desde la línea de comandos del sistema operativo, se puede redireccionar la salida de un programa a la entrada estándar de un programa Perl." [BARKAKATI, 1998, PÁG. 145]*

Ejemplo: Si se ejecuta un programa Perl de tal forma que obtenga su entrada a partir de un archivo y que envié su salida a otro archivo, se debe ejecutar el programa de la siguiente manera:

perl traductor.pl < ing.txt > esp.txt

El signo menor que es un operador de redireccionamiento de entrada y el signo mayor que es el de salida, es decir, el sistema operativo conecta los archivos especificados a STDIN y STDOUT.

Para leer una línea del identificador de archivo STDIN, es necesario escribir <STDIN>. Para almacenar la línea de texto en una variable, todo lo que se necesita es asignar el valor a una variable escalar. Perl toma toda la línea incluyendo el carácter nueva línea como su último carácter.

Existe una manera especial de almacenar la línea y es utilizando la variable especial $_, ejemplo:

```
while (<STDIN>){
print $_;
}
```

En este ejemplo devolverá cualquier carácter introducido hasta encontrar el carácter de fin de archivo. En Unix se utiliza Ctrl+D y en la ventana de MSDOS en Windows Ctrl+Z. Cabe destacar que la asignación automática del texto a la variable especial $_ se produce únicamente en la sentencia while.

Pero, si se desea almacenar múltiples líneas se puede asignar STDIN a una variable de arreglo y en este caso Perl, evalúa <STDIN> en un contexto de línea, ya que se necesita inicializar el arreglo para una lista de valores escalares.

Otra forma de leer la entrada estándar es obviando el identificador de archivo y usar la expresión del operador diamante <>. Por ejemplo: la línea contenida en el programa print.pl mostrará cada una de las líneas introducidas, hasta que se le indique el final del archivo.

```
print while (<>);
```

Si se introducen varios nombres de archivos en la línea de comandos, el operador diamante (<>) leerá todas las líneas de todos los ficheros especificados, es decir los concatenará en un único archivo.

```
perl print.pl archivo1 archivo2 archivo3
```

STDOUT (Salida)

En Perl, generalmente se utiliza la función print para enviar datos a la salida estándar. Sin embargo para enviar la salida a un archivo que debe estar abierto (reconocido a través de su identificador de archivo) lo que se utiliza es la función print con el nombre del identificador de archivo como argumento.

```
print IDENTIFICADOR_DE_ARCHIVO $cadena1, $cadena2;
```

En caso de que se llegará a omitir el nombre del identificador de archivo, la salida se reflejará en la salida estándar.

Existe una forma especial de utilizar la función print y es útil para mostrar un conjunto de líneas de texto y no utilizar la función print repetidamente. Aquí se pueden incluir nombres de variables dentro del texto y Perl las substituirá.

En el ejemplo siguiente se muestran un conjunto de líneas de texto sin utilizar la funciòn print repetidamente:

```
$cadena1="Perl";
print <<Final_del_texto;
Esta es una prueba de escritura
que se puede hacer utilizando
la funcion print en un
conjunto de lineas en $cadena1
Final_del_texto
```

Apertura de Archivos. Uso de la Función Open

"La función open ( ) se usa para abrir un archivo y crear una conexión con él denominada identificador de archivo." [MEDINETS, 1997, PÁG 173]

La sintaxis es la siguiente:

```
open (IDENTIF_ARCH);
```

En donde IDENTIF_ARCH es el nombre del identificador. También es el nombre de la variable escalar que contiene el nombre del archivo que se desea abrir. Ejemplo:

```
$ARCHIVO_ENTRADA = "bitacora.log";
open ($ARCHIVO_ENTRADA);
@bitacora = <$ARCHIVO_ENTRADA>;
close ($ARCHIVO_ENTRADA);
foreach (@bitacora) {
        print ();
}
```

En este caso, se asigna el nombre de archivo "bitacora.log" a la variable $ARCHIVO_ENTRADA, se lee el archivo y almacena en @bitacora convirtiéndolo en un solo elemento del arreglo, ahora se debe cerrar el archivo con la función close. Y con el enunciado foreach se mostrarán cada una de las líneas almacenadas en la variable @bitacora.

La función open ( ) tiene variantes para acceder a los archivos y la tabla 17 muestra los distintos métodos.

TABLA 17. MÉTODOS PARA ABRIR UN ARCHIVO

Método	Descripción
open(IDENTIF_ARCH);	El archivo se abrirá sólo para entrada.
open (IDENTIF_ARCH, NOMARCH.EXT);	Abre el archivo denominado NOMARCH.EXT como de entrada, utilizando IDENTIF_ARCH como el identificador de archivo.
open (IDENTIF_ARCH, <NOMARCH.EXT);	Abre el archivo NOMARCH.EXT como de entrada, utilizando IDENTIF_ARCH como identificador de archivo
open (IDENTIF_ARCH, >NOMARCH.EXT);	Abre el archivo NOMARCH.EXT como de salida, utilizando IDENTIF_ARCH como el identificador de archivo.
open (IDENTIF_ARCH, -);	Abre la entrada estándar.
open (IDENTIF_ARCH, >-);	Abre la salida estándar.
open (IDENTIF_ARCH, >>NOMARCH.EXT);	Abre el archivo NOMARCH.EXT para añadir información.
open (IDENTIF_ARCH, +<NOMARCH.EXT);	Abre el archivo NOMARCH.EXT tanto de entrada como para salida, utilizando IDENTIF_ARCH como el identificador de archivo. Si el archivo no existe dará un error.
open (IDENTIF_ARCH, +>NOMARCH.EXT);	Abre el archivo NOMARCH.EXT tanto de entrada como para salida, utilizando IDENTIF_ARCH como el identificador de archivo. Si el archivo no existe se creerá.
open (IDENTIF_ARCH, | PROGRAMA);	Envía la información mandada a la salida estándar IDENTIF_ARCH a otro programa.
open (IDENTIF_ARCH, PROGRAMA |);	Lee la salida de otro programa utilizando IDENTIF_ARCH

2.1.4 Visión general de las expresiones regulares

Una expresión regular sirve para encontrar patrones en cadenas, por ejemplo, un nombre específico en un directorio telefónico. La comparación de patrones es una de las características más poderosas de Perl y una de las más complicadas.

"En Perl, existen tres principales usos de las expresiones regulares: comparación, sustitución y traducción. La operación de comparación utiliza el operador m / /, el cual se evalúa como verdadero o falso. La operación de sustitución reemplaza una expresión por otra; usa el operador s / /. La operación de traducción, traduce un conjunto de caracteres a otro y emplea el operador tr / /." [MEDINETS, 1997, PÁG 193]

Los tres operadores de expresión regular trabajan con la variable $_ como la cadena a inspeccionar y si se desea utilizar una cadena distinta a $_, se usarán los operadores de vínculo. Las expresiones regulares se encierran entre un par de barras (/.../) para emparejar la cadena en una línea de texto.

El operador de comparación (m / /)

El operador de comparación se usa para localizar patrones dentro de cadenas. El uso más común es para buscar una cadena específica dentro de un archivo de datos. Este operador es tan frecuente que Perl permite omitir la m de operador, siempre y cuando utilice las diagonales como delimitadores. Ejemplo:

```
$cadena = "Juan";
open (INPUT, "<directorio.dat");
while (<INPUT>){
        if (/$cadena/) {
                print "Cadena $cadena encontrada en la linea $.";
        }
}
close (INPUT);
```

Este ejemplo lee todas las líneas de entrada del archivo directorio.dat y cuando encuentra la cadena "Juan" se ejecuta el enunciado print. Este imprimirá la cadena encontrada y el número de línea donde se encontró la cadena. El operador de comparación tiene diversas opciones y estas se especifican después del último delimitador de patrón. La Tabla 18 muestra las opciones del operador.

TABLA 18. OPCIONES DEL OPERADOR DE COMPARACIÓN

Opción	Descripción
g	Localiza todas las ocurrencias del patrón en la cadena.
i	Ignora el uso de mayúsculas o minúsculas en la cadena
m	Trata a la cadena como de varias líneas.
o	Compila el patrón sólo una vez.
s	Trata a la cadena como a una sola línea
x	Permite ampliar las expresiones regulares.

El operador de sustitución (s / /)

Este operador se utiliza para modificar cadenas. Requiere de dos operandos, patrón y reemplazo (s/PATRON/REEMPLAZO/). Es decir, este enunciado cambia el primer PATRON en $_ por REEMPLAZO. Ejemplo:

```
print "Escribe la oracion a validar\n";
$cadena1=<STDIN>;
$cadena1=lc($cadena1);
$cadena1=~s/\s+//g;
$cadena2=reverse($cadena1);
print "\nLa oracion es un palindromo\n" if ($cadena2 eq $cadena1);
print "\nLa oracion no es un palindromo\n" if ($cadena2 ne $cadena1);
```

Este ejemplo verifica si la cadena introducida es un palíndromo, almacena el enunciado en cadena1, lo convierte a minúsculas con la función lc; en la expresión regular se sustituye los espacios en blanco eliminándolos de la cadena1, posteriormente se invierte la cadena almacenándola en cadena2 y se hace una comparación entre cadena1 y cadena2 si el enunciado es o no un palíndromo el programa lo imprimirá en la pantalla.

En este ejemplo aparece una tilde (~) que tiene un significado especial en Perl ya que por omisión, las operaciones de búsqueda, modificación y traducción trabajan sobre la variable $_, así que si se desea vincular los operadores de expresión regular a una variable distinta a $_ se debe utilizar alguno de los dos operadores de vínculo: el normal (=~) y su complemento (!~).

El operador de sustitución así como el operador de comparación tienen varias opciones.

TABLA 19. OPCIONES DEL OPERADOR DE SUSTITUCIÓN

Opción	Descripción
e	Obliga a evaluar el patrón de reemplazo como una expresión.
g	Reemplaza en la cadena todas las ocurrencias del patrón.
I	Ignora el uso de mayúsculas o minúsculas en la cadena.
m	Trata a la cadena como de varias líneas
o	Compila el patrón una sola vez.
s	Trata a la cadena como a una sola línea.
x	Permite ampliar las expresiones regulares.

El operador de traducción (tr / /)

Se utiliza para cambiar caracteres individuales en la variable $_ (tr/CARACTERES/REEMPLAZO/), es decir se sustituye los caracteres especificados en CARACTERES con los caracteres indicados en REEMPLAZO. Si la lista de REEMPLAZO de caracteres es más corta que la lista de CARACTERES, el último carácter de la lista de REEMPLAZO se repite con la misma frecuencia que sea necesaria. Pero si se da más de un carácter de REEMPLAZO para un carácter comparado, sólo se usa el primero.

El operador de traducción no hace interpolaciones de variables, a diferencia de los operadores de comparación y sustitución. Ejemplo:

```
open (INPUT, "<directorio.txt");
while (<INPUT>){
        $_= tr/a/ /; print $_;
}
close (INPUT);
```

El ejemplo anterior revisará en el archivo directorio.txt el número de letras "a" que contiene el texto y mostrará el total de letras.

El operador de traducción tiene algunas opciones que se explican en la tabla 20.

TABLA 20. OPCIONES DEL OPERADOR DE TRADUCCIÓN

Opción	Descripción
c	Complementa la lista de caracteres de cotejo. La traducción se realiza para cada carácter que no coincida con la lista de comparación.
d	Elimina cualquier carácter en la lista de comparación que no tenga un carácter correspondiente en la lista de reemplazo.
s	Reduce a un solo caso los casos repetidos de caracteres coincidentes.

Los cuantificadores permiten especificar cuántas veces tiene que estar presente un determinado componente para que la comparación sea verdadera. Estos, se utilizan cuando no se sabe cuantos caracteres se tienen que comparar. A continuación se mencionan los seis tipos (Tabla 21):

TABLA 21. CUANTIFICADORES

Cuantificador	Descripción
*	El componente debe estar presente cero o más veces.
+	Una o más veces.
?	Cero o una vez
{n}	"n" veces.
{n,}	Debe estar presente por lo menos "n" veces.
{n,m}	Debe estar presente por lo menos "n" veces y no más de "m" veces.

Para crear patrones de comparación se utilizan los meta-caracteres, que son caracteres que tienen un significado adicional más allá de su significado literal. Algunos ejemplos de ellos se mencionan a continuación (Tabla 22):

TABLA 22. META-CARACTERES, META-PARÉNTESIS Y META-SECUENCIAS DE LAS EXPRESIONES REGULARES

Meta-caracteres	Descripción
^	Se utiliza para comparar al principio de la cadena.
.	Compara cualquier carácter, excepto el de línea nueva.
$	Compara al final de una cadena.
|	Permite especificar dos valores que pueden hacer que la comparación tenga éxito.
*	Indica que "aquello" que sigue inmediatamente a la izquierda debe compararse 1 o más veces a fin de que se evalúe como verdadero.
?	Indica que "aquello" que sigue inmediatamente a la izquierda debe compararse 0 o 1 vez a fin de que se evalué como verdadero.
Meta-paréntesis	**Descripción**
()	Los paréntesis permiten afectar el orden de evaluación del patrón.
{n,m}	Las llaves especifican cuántas veces debe compararse "aquello" que está inmediato a la izquierda. Más adelante se desglosan este tipo de cuantificadores.
[]	Permiten crear una clase de caracteres.
Meta-secuencias	**Descripción**
\b	Busca el límite de una palabra.
\B	Busca no delimitadores de palabra.
\d	Busca un carácter que represente un dígito.
\D	Busca un carácter que no represente un dígito.
\e	Carácter de escape.
\E	Termina la secuencia \L o \U.
\f	Avance de página.
\G	Busca sólo donde los previos m//g no existan.
\l	Cambia el siguiente carácter a minúsculas.
\L	Cambia los siguientes caracteres a minúsculas hasta \E.
\n	Salto de línea.
\Q	Manejo de meta-caracteres.
\r	Retorno de carro.
\s	Busca un carácter en blanco.
\S	Busca no caracteres en blanco.
\t	Tabulador.
\u	Cambia el siguiente carácter a mayúscula.
\U	Cambia los siguientes caracteres a mayúsculas hasta \E.
\w	Busca una palabra incluyendo el carácter underscore (_).
\W	Compara un solo carácter que no sea palabra.
\Z	Busca sólo al final de la cadena o antes de una nueva línea.
\$, \@, \& \%	Sigo de pesos, arroba, ampersand, signo de porcentaje.

Función Split

La función split es muy útil para separar cadenas de caracteres. Ejemplo:

```
$cadena="empresa=Donuts&producto=Donettes&precio=100";
@tabla=split (/&/,$cadena);
print $tabla[0],"\n";
print $tabla[1],"\n";
print $tabla[2],"\n";
```

En el ejemplo anterior, la función split divide a la cadena usando el carácter &, imprime las parejas nombre-valor (empresa=Donuts; producto=Donettes; precio=100).

Para construir expresiones regulares complejas, consultar la tabla 23, en donde se muestran algunos ejemplos de estas:

TABLA 23. EJEMPLOS DE EXPRESIONES REGULARES

Expresión	Patrón de la expresión regular
d.l	una "d" seguida de un carácter cualquiera y una "l" (del, dal, dzl)
^hol	"hola" al principio de la cadena (hola, holita)
e$	una "e" al final de la cadena (este, ese)
ind*	"in" seguido de cero o más caracteres "d" (in, ind, indd)
.*	cualquier cadena, sin retorno de carro
^$	una cadena vacía
[qjk]	una "q", o una "j" o una "k"
[^qjk]	no sea "q", o una "j" o una "k"
[a-z]	cualquier letra entre la "a" y la "z"
[^a-z]	no sean letras minúsculas
[a-zA-Z]	una letra minúscula o mayúscula
[a-z]+	una secuencia no vacía de letras minúsculas
f.*ca	coincide con p.e. "fca", "foca", "flaca", "flor vaca"
^[\t]*$	una línea en blanco, o combinaciones de espacios y tabuladores
[-+]?\d*\.?\d*	lo mismo que [-+]?[0-9]*\.?[0-9]* (números decimales)
pepe|juan	ó "pepe" ó "juan"
(pe|hue)cos	ó "pecos" ó "huecos"
(da)+	ó da ó dada ó dadada
[01]	un "0" ó un "1"
fia|fea|fua	coincida con "fia", "fea" ó "fua"
f(i|e|u)a	coincida con "fia", "fea" ó "fua"

Perl maneja otros conceptos tales como referencias, módulos, reportes, archivos, entre otros, no se abarcará sobre estos temas porque no es el objetivo de esta tesina, si se desea consultar sobre estos temas ver la referencias bibliográficas.

2.2 CGI

Como ya se había hablado anteriormente, el CGI es un estándar para comunicar aplicaciones externas con los servidores Web. Las capacidades de Perl en procesamiento de texto como ya se ha visto, hacen que sea una herramienta excelente en la escritura de programas CGI.

2.2.1 CGI en Acción

Cuando el usuario recibe un documento dinámico HTML a través de CGI, la secuencia básica que se sigue es la siguiente:

- El usuario selecciona un enlace que provoca que el navegador Web solicite un documento HTML que contiene un formulario.

- El servidor Web envía el formulario HTML, que se muestra en el navegador del usuario.

- El usuario llena los campos del formulario y pulsa el botón de Envió. A su vez, el navegador envía los datos del formulario usando el método GET o POST (como se especifica el atributo method en la etiqueta <FORM> del formulario HTML). En cada método, el navegador envía el URL especificado en el atributo action de la etiqueta <form>.

- A partir del URL, el servidor Web determina la activación del script CGI definido en el URL y envía la información a ese script.

- El programa CGI procesa la información y devuelve el texto HTML al servidor Web (que lee la salida del programa CGI). Este programa puede realizar consultas o actualizaciones en bases de datos, lectura o escritura en disco, etc. El servidor, a su vez, anexa una cabecera MIME y devuelve el texto HTML al navegador del usuario.

- El navegador Web muestra el documento recibido del servidor Web. Dicho documento contiene la información que depende de lo que el usuario haya introducido en el formulario HTML.

En general, este proceso es el más común. Otro elemento importante dentro de este proceso es la forma de obtener información ya sea de un usuario o del servidor Web, y ahí es donde intervienen las variables de ambiente.

2.2.2 Variables de ambiente

Los sistemas operativos usan las variables de ambiente para almacenar bits de información que son necesarios para el funcionamiento de la computadora. Así almacenan la ruta del directorio temporal, ubicación de archivos ejecutables, etc. *"Una variable de ambiente no es nada más que un nombre asociado a una cadena". [BARKAKATI, 1998, PÁG 369]*

Para obtener las variables de ambiente en un script Perl, es a través del hash %ENV. Las variables de ambiente proporcionan un mecanismo apropiado para transferir información de un programa a otro. A continuación se muestran algunas variables de ambiente (Tabla 24).

TABLA 24. VARIABLES DE AMBIENTE CGI

Nombre de la variable	Descripción
AUTH_TYPE	El método de autentificación usado para validar a un usuario. Este está en blanco si la respuesta no requiere autentificación.
CONTENT_LENGTH	Proporciona, en forma opcional, la longitud en bytes, del contenido proporcionado por el script a través del identificador de archivos STDIN.
CONTENT_TYPE	Proporciona en forma opcional el tipo de contenido disponible del identificador de archivo STDIN.
DOCUMENT_ROOT	El directorio donde están los documentos estáticos.
GETAWAY_INTERFACE	Proporciona la versión de CGI manejada por el servidor Web local.
HTTP_ACCEPT	Proporciona una lista separada por comas de los tipos MIME que aceptará el software del navegador.
HTTP_FORM	Proporciona la dirección e-mail del usuario.
HTTP_USER_AGENT	Proporciona el tipo y versión del navegador Web del usuario.
PATH_INFO	Contiene de manera opcional cualquier información adicional de trayectoria de la solicitud HTTP que invocó el script.
PATH_TRANSLATED	Mapea la trayectoria virtual del script a la trayectoria física empleada para invocar el script.
QUERY_STRING	Contiene de manera opcional información del formulario cuando se emplea el método GET del procesamiento de formularios. Esta información se agrega al URL
REMOTE_ADDR	Contiene la dirección IP del usuario que envió la petición.
REMOTE_HOST	Proporciona, en forma opcional el nombre de dominio del sitio desde el que se ha conectado el usuario.
REMOTE_IDENT	Proporciona, en forma opcional, una identificación del cliente cuando su servidor local ha contactado un servidor IDENTD en una máquina cliente.
REMOTE_USER	Es el login del usuario, autentificado por el servidor Web.
REQUEST_METHOD	El método (GET o POST) mediante el cual se pondrá a disponibilidad del script información y formularios.
SCRIPT_NAME	Contiene la trayectoria virtual a un script.
SERVER_NAME	Contiene el nombre de host o su dirección IP.
SERVER_PORT	Contiene el número de puerto sobre el que el servidor Web local está escuchando. El puerto estándar es el 80.
SERVER_PROTOCOL	Contiene la versión del protocolo Web que usa el servidor.
SERVER_SOFTWARE	Contiene el nombre y versión del software de servidor Web.

Para conocer las variables de ambiente del sistema, se puede escribir y ejecutar el siguiente script de Perl, escribiendo en el navegador: http:\\localhost\cgi-bin\variables.cgi donde localhost es el nombre del host local y variables.cgi es el nombre del script que se muestra a continuación:

```
#!c:\perl\bin\perl.exe

print "Content-type: text/html\n\n";

my $var_name;
foreach $var_name (sort keys %ENV) {
        print "<P><B>$var_name</B></BR>";
        print $ENV{$var_name};
}
```

Este script es ejecutado en un sistema operativo Windows, por lo que la ruta especificada en la línea 1 (#!c:\perl\bin\perl.exe) del script variables.cgi corresponde a la ruta donde se encuentra instalado el interprete de Perl. Cabe destacar también que se debe tener instalado un servidor Web y colocar el script en el directorio cgi del mismo.

Si se ejecuta correctamente el script anterior, en el navegador aparecerán las variables de ambiente encontradas en el sistema, lógicamente cambiarán algunas variables de acuerdo al sistema operativo y software que se utilice, además de algunas otras como por ejemplo: El nombre del host, la dirección IP, el software del servidor Web, etc.

COMSPEC
C:\WINDOWS\COMMAND.COM

DOCUMENT_ROOT
c:/archivos de programa/apache group/apache/htdocs

GATEWAY_INTERFACE
CGI/1.1

HTTP_ACCEPT
*image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */**

HTTP_ACCEPT_ENCODING
gzip, deflate

HTTP_ACCEPT_LANGUAGE
es-mx

HTTP_CONNECTION
Keep-Alive

HTTP_HOST
prometeo.universo

HTTP_USER_AGENT
Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)

PATH
C:\Archivos de programa\Apache Group\Apache;C:\PERL\BIN\;C:\WINDOWS;C:\WINDOWS\COMMAND

REMOTE_ADDR
143.168.1.2

REMOTE_PORT
1029

REQUEST_METHOD
GET

REQUEST_URI
/cgi-bin/variables.cgi

SCRIPT_FILENAME
c:/archivos de programa/apache group/apache/cgi-bin/variables.cgi

SCRIPT_NAME
/cgi-bin/variables.cgi

SERVER_ADDR
143.168.1.2

SERVER_ADMIN
you@your.address

SERVER_NAME
prometeo.universo

SERVER_PORT
80

SERVER_PROTOCOL
HTTP/1.1

SERVER_SIGNATURE

Apache/1.3.12 Server at prometeo.universo Port 80

SERVER_SOFTWARE
Apache/1.3.12 (Win32)

WINDIR
C:\WINDOWS

NOTA: Para ejecutar este script, hay que comprobar si en el servidor Web es una versión http de NCSA, por lo que existirá un directorio cgi-bin en donde se almacenará este script. Este es un directorio especial, donde todos los scripts CGI residen. El servidor conoce este directorio, y sabe que existen scripts que deberán ser ejecutados y su salida deberá ser enviada al navegador del usuario. Algunos servidores Web, están configurados de tal manera que los archivos con una determinada extensión (generalmente ".cgi") son reconocidos como scripts y serán ejecutados como si estuvieran en un directorio cgi-bin.

2.2.3 Escritura de scripts CGI con Perl

Para entender la construcción de scripts se desarrolla el siguiente ejemplo:

1. Construir una página HTML, en donde los empleados de una compañía reportarán las fallas que tengan con su equipo de cómputo al departamento de soporte técnico a través de un formulario. Está página se llamara indicencias.html y se colocará en el directorio htdocs del Servidor Web (en este caso se utilizará Apache Web Server). En el anexo C se encuentra la instalación del servidor.

```
<HTML>
<HEAD>
<TITLE>Incidencias de los usuarios de PCs.</TITLE>
<BODY>
<H1>Incidencias de los usuarios de PCs.</H1>
<H3>Si tiene problemas o necesita de los servicios del departamento</H3><P>
<H3>de Soporte Técnico rellene y envié este formulario.</H3>
<FORM ACTION='/cgi-bin/avisos.cgi' METHOD="POST">

<PRE>
Nombre: <INPUT TYPE="TEXT" NAME="Nombre" SIZE="40" MAXLEGHT="40"><P>
Departamento: <SELECT NAME="Centro" SIZE="0">
<OPTION VALUE="Direccion General"> Dirección General
<OPTION VALUE="Mercadotecnia"> Mercadotecnia
<OPTION VALUE="Ventas"> Ventas
<OPTION VALUE="Compras"> Compras
<OPTION VALUE="Almacen"> Almacén
<OPTION VALUE="Recursos Humanos"> Recursos Humanos
<OPTION VALUE="Contabilidad"> Contabilidad
</SELECT><P>
Extensión: <INPUT TYPE="TEXT" NAME="Telefono" SIZE="5" MAXLENGTH="5"><P>
Problema:
<INPUT TYPE="RADIO" NAME="Tipo" VALUE="Averia" CHECKED> Averia
<INPUT TYPE="RADIO" NAME="Tipo" VALUE="Programa"> Instalación de programas
<INPUT TYPE="RADIO" NAME="Tipo" VALUE="Configuracion"> Configuración de equipo
<INPUT TYPE="RADIO" NAME="Tipo" VALUE="Informacion" > Información
<INPUT TYPE="RADIO" NAME="Tipo" VALUE="Otro" > Otros<P>
N. Inventario del equipo: <INPUT TYPE="TEXT" NAME="Inventario" SIZE="15"
MAXLENGTH="15"><P>
Descripción de la incidencia:
<TEXTAREA NAME="Descripcion" ROWS="2" COLS="50"></TEXTAREA><P>
</PRE>
<INPUT TYPE="SUBMIT" VALUE="Enviar">
<INPUT TYPE="RESET" VALUE="Borrar">
</FORM>
</BODY></HTML>
```

A continuación se muestra la pantalla (Figura 2.1) que será el resultado del código anterior. Escribiendo en el navegador la dirección donde se encuentra la página de incidencias.html. Por ejemplo *http://localhost/incidencias.html*

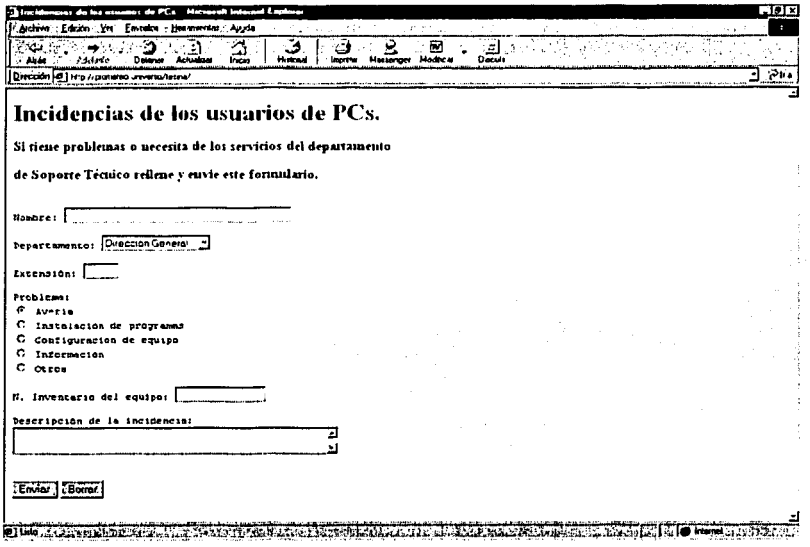


Fig 2.1 Página de incidencias

2. El siguiente paso, es procesar la información que se enviara, almacenándola en un archivo de datos llamado reportes y mostrará al usuario un mensaje de datos recibidos. Este script se llamará avisos.cgi, recordando que se hace referencia a este archivo en el método ACTION de la etiqueta FORM del formulario anterior. Este script se coloca en el directorio cgi-bin del Servidor Web.

```
#!C:\Perl\bin\Perl.exe

&Cabecera_HTML;

print "<HTML><HEAD>\n";
print "<TITLE>Datos de la incidencia</TITLE>\n";
print "</HEAD><BODY>\n";
print "<H3>Datos de la incidencia recibida</H3>\n";
print "<HR>\n";
print "<PRE>\n";
%hash = Parametros(&Entrada);
print "</PRE>\n";
print "Sus datos han sido recibidos correctamente. Gracias por su colaboracion.<BR>\n";
print "<P><HR>\n";
print "</BODY></HTML>\n";
```

```
sub Entrada{
        local ($entrada);
        if ($ENV{'REQUEST_METHOD'} eq "POST"){
                read(STDIN, $entrada, $ENV{'CONTENT_LENGTH'});
        }elsif ($ENV{'REQUEST_METHOD'} eq "GET"){
                $entrada = $ENV{'QUERY_STRING'};
        }else{
                $entrada = $ARGV[0];
        }
        return $entrada;
}

sub Parametros{
        local ($linea) = @_;
        local ($clave, $valor);
        local %hash;

        open(REPORTES, ">>c:\\reportes.dat");
        $linea =~ s/%(..)/pack("C",hex($1))/ge;
        $linea =~ s/\+/ /g;
        foreach (split(/\&/, $linea)){
                ($clave, $valor) = split (/=/, $_);
                eval "\$$clave \= q($valor)";
                $hash{"$clave"} = $valor;
                print "$clave = $valor\n";
                print REPORTES "$clave = $valor\n";
        }
        close(REPORTES);
        return %hash;
}

sub Cabecera_HTML{
        print "Content-type: text/html\n\n";
}
```

Cuando el usuario envía un formulario, el script CGI recibe los datos como pares de nombre-valor. Los nombres son lo que se definió en las etiquetas INPUT, y los valores aquello que el usuario haya escrito o seleccionado.

Estos pares nombre-valor llegan como una larga cadena que necesitamos formatear, por ejemplo:

```
"http://prometeo.universo/cgi-
bin/avisos.cgi?Nombre=Juan+Carlos&Centro=Recursos+Humanos&Telefono=53298&Tipo=Programa&
Inventario=2R64736280&Descripcion=Instalaci%F3n+de+programa+de+Dise%F1o+Gr%E1fico"
```

Así que sólo se tiene que dividir la cadena donde están los signos '&' y '=', y luego hacer lo siguiente a cada nombre-valor.

1. Convertir todos los signos '+' a espacios.

2. Convertir todas las secuencias '%xx' al valor del carácter cuyo valor ASCII sea 'xx' en hexadecimal.

Esto se hace necesario porque la larga cadena original esta codificada según el URL, para permitir los signos '&' , '=' y todo lo que el usuario introduzca.

Para los envíos con GET, será la variable de ambiente QUERY_STRING. Y para los envíos con POST, habrá que leer del STDIN. El número exacto de bytes a leer estará en la variable de ambiente CONTENT_LENGTH. En las subrutinas parámetros y entrada es donde se reconoce el método del formato de envió y posteriormente la división de la cadena enviada a través de las variables de ambiente de acuerdo al método utilizado.

En la subrutina parámetros, se abre el archivo reportes.dat para añadir la información que envió el usuario, utilizando REPORTES como el identificador de archivo.

Para devolver la respuesta al usuario, se escribe la siguiente línea al principio del script: Content-Type: text/html, que especifica el tipo de documento que se enviará al navegador del usuario.

Después, se escribe la página de respuesta en HTML al STDOUT, y será enviada al usuario cuando el script esté ejecutado. La Figura 2.2 muestra el contenido de la página html una vez ejecutado el script.

El código del script Perl es sencillo y muy robusto ya que se utiliza expresiones regulares, subrutinas, archivos, enunciados de control, ciclos, etc, además del código HTML. Pero esta programación se puede hacer aun más fácil utilizando algunos agregados del lenguaje de programación Perl, como los modulos.

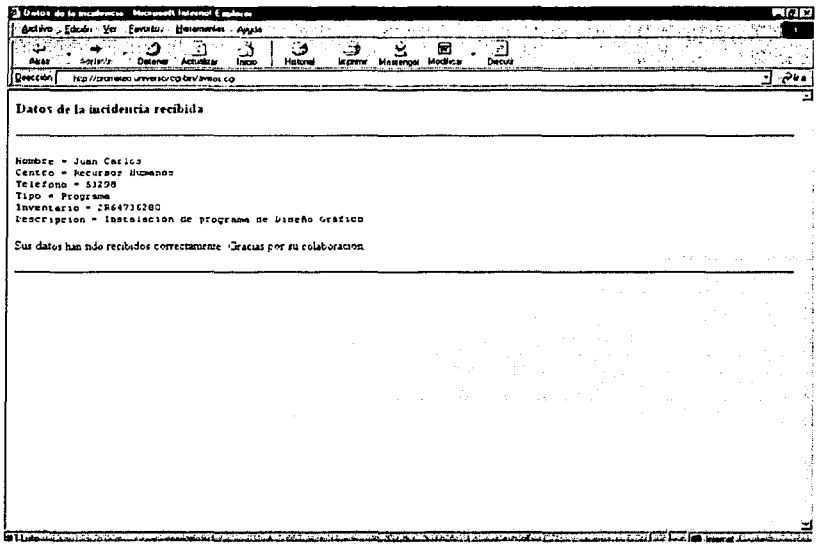


Fig 2.2 Datos recibidos de la incidencia

2.2.4 Uso del modulo CGI.pm

El modulo CGI.pm fue desarrollado por Lincoln D. Stein utiliza el estilo de programación orientada a objetos. Este módulo debe copiarse en el directorio de librerías Perl. Para una mayor referencia acerca de la instalación, funcionalidad, errores, etc., consultese la siguiente dirección electrónica

http://stein.cshl.org/WWW/software/CGI

CGI.pm permite aceptar una consulta y extraer fácilmente los parámetros enviados por el usuario. Para utilizar el módulo CGI se tiene que introducir la siguiente línea en el script Perl:

```
use CGI;
```

A continuación se debe crear un objeto CGI como el siguiente:

```
$query = new CGI;
```

La creación del objeto CGI desencadena el procesamiento de la consulta y extrae todas las variables de ambiente importantes para la programación CGI. El valor devuelto $query es una referencia al objeto CGI. A partir de esta referencia, es posible acceder a los métodos del objeto.

El procesamiento de la consulta es sólo una parte del modulo CGI. Adicionalmente, el objeto tiene otras capacidades como las siguientes:

- Métodos para crear las cabeceras HTTP que se deben incluir en los documentos HTML que se envían de vuelta al usuario a partir del script.
- Métodos para crear formularios HTML
- Métodos de gestión de capacidades avanzadas.

Para este caso no se presentará toda la funcionalidad del módulo, se utilizara lo que resulte útil a la hora de escribir scripts.

El módulo CGI.pm puede ser utilizado de dos formas diferentes:

Procedural. Adecuada para scripts pequeños

```
use CGI qw/:standard/;
print  header(),
start_html(-title=>'Saludos'),
      h1('Saludos'),
      'Hola, mundo !',
      end_html();
```

Orientada a objetos. Más adecuada para scripts grandes; además permite disponer de varios objetos CGI dentro del mismo programa, con estados diferentes.

```
use CGI;
$q = new CGI;
print  $q->header(),
$q->start_html(-title=>'Saludos'),
            $q->h1('Saludos'),
            'Hola, mundo !',
$q->end_html();
```

Cabecera http. El método header imprime la cabecera (por defecto text/html).

```
print $q->header();
```

Comienzo del documento html. Genera la cabecera HTML colocando un título a la página y abriendo el tag BODY:

```
print $q->start_html(-title=>'Prueba Perl', -BGCOLOR=>'white');
```

Final del documento html. Escribe el cierre del tag BODY y del tag HTML.

```
print $q->end_html();
```

Algunos tags de formato

```
print $q->hr;              # imprime <hr>
print $q->i("cursiva");    # imprime <i>cursiva</i>
print $q->b("negrita");    # imprime <b>negrita</b>
print $q->h1("Encabezado"); # imprime <h1>encabezado</h1>
```

Lectura de parámetros. El uso más frecuente del módulo CGI es la lectura de los parámetros que recibe de un formulario, independientemente de si se han enviado a través de GET o de POST.

```
$name = $q->param('nombre');
$age  = $q->param('edad');
```

Tags con atributos. Para añadir atributos a un tag, se puede pasar una referencia a un arreglo asociativo como primer argumento; las claves y valores del arreglo se convierten en los nombres y valores de los atributos. Por ejemplo:

```
print $q->a({-href=>"enlace.html"}, "Pulsa para ir al enlace");
# imprime: <a href="enlace.html">Pulsa para ir al enlace</a>
```

Tags para tablas.

```
start_table()   # imprime <TABLE>
end_table()     # imprime </TABLE>
start_Tr()      # imprime <TR>
end_Tr()        # imprime </TR>
start_th()      # imprime <TH>
end_th()        # imprime </TH>
start_td()      # imprime <TD>
end_td()        # imprime </TD>
```

Tags para formulario. Para abrir el tag del formulario <FORM>:

```
print $q->startform($method, $action);
```

Para insertar una etiqueta de texto:

```
print $q->textfield( -name=>'NombreDelCampo',
                     -default=>'valor por defecto',
                     -size=>20,
                     -maxlength=>40 );
```

Para insertar un botón de submit:

```
print $q->submit( -name=>'button_name',
                  -value=>'caption');
```

Para cerrar el tag del formulario </FORM>

```
print $q->endform();
```

Recordando el script avisos.cgi, ahora se programará utilizando el módulo CGI.pm facilitándose aun más la escritura del script. Reconstruyendo el script avisos.cgi con el módulo quedará de la siguiente manera:

```
#!c:\Perl\bin\Perl.exe

use CGI;

$query = new CGI;

$nombre=$query->param('Nombre');
$departamento=$query->param('Centro');
$ext=$query->param('Telefono');
$problema=$query->param('Tipo');
$inventario=$query->param('Inventario');
$descripcion=$query->param('Descripcion');

print $query->header;
print $query->start_html("Datos de la incidencia");

print <<END;
        <H3>Datos de la incidencia recibida</H3><HR><PRE>
        Nombre = $nombre
        Centro = $departamento
        Telefono = $ext
        Tipo = $problema
        Inventario = $inventario
        Descripcion = $descripcion
        </PRE>
        Sus datos han sido recibidos correctamente. Gracias por su
```

```
        colaboracion.<BR>
        <P><HR>
END

print $query->end_html;

open(REPORTES, ">>c:\\reportes.dat"); #Lo siguiente es una sola linea
        print REPORTES "Nombre = $nombre\nCentro = $departamento\nTelefono =
        $ext\nTipo = $problema\nInventario = $inventario\nDescripcion =
        $descripcion\n";
close(REPORTES);
```

Como se puede observar la programación es orientada a objetos y en este caso al objeto CGI. Es muy fácil construir este tipo de scripts, a continuación algunos ejemplos.

2.2.5 Ejemplos

Se necesita validar a usuarios para que entren al sistema y en caso de que el usuario no este registrado preguntar sus datos para darlo de alta. Este ejemplo se desarrollará con tres scripts CGI y un formulario HTML.

1. Generar un script dinámico que contenga un formulario en donde se introducirá el usuario y password. Su nombre será validar.cgi.

2. Un script que valide al usuario y su password. Se llamará entrar.cgi.

3. Mostrar un formulario que solicite la información del usuario en caso de no estar registrado y la página html se llamará forma.html.

4. Y por último un script que almacene la información del usuario llamado nuevo.cgi.

Los scripts cgi serán almacenados en el directorio cgi-bin del directorio del servidor Web, y el formulario dentro del directorio htdocs del servidor Web.

A continuación se desarrolla el primer script: validar.cgi

```
#!c:\Perl\bin\Perl.exe

use CGI;

$query = new CGI;

print   $query->header,
        $query->start_html("Acceso a los usuarios"),
        $query->h1("Bienvenido"),
        $query->hr,
        $query->p("Por favor escriba su login y su password"),
        $query->startform("POST","/cgi-bin/entrar.cgi'),
        $query->start_table ( {-border=>1} ),
        $query->start_Tr(),
                $query->start_td(),
                        "Usuario",
                $query->start_td(),
                $query->textfield( -name=>'Usuario',
                                   -size=>20,
                                   -override => 1,
                                   -maxlength=>20 ),
        $query->end_Tr(),
        $query->start_Tr(),
                $query->start_td(),
                "Password",
                $query->start_td(),
                        $query->password_field( -name=>'Password',
                                                -override => 1,
                                                -size=>8,
                                                -maxlength=>8 ),
    $query->end_Tr(),
    $query->end_table(),
    $query->br,
    $query->submit( -name=>'Enviar',
            -value=>'Enviar'),
    "   ",
    $query->reset( -name=>'Borrar',
            -value=>'Borrar'),
    $query->endform(),
    $query->hr,
    $query->p("En caso de no tener una cuenta presione ",
    $query->a({-href=>'./forma.html'}, "aqui")),
    $query->end_html;
```

Escribiendo en el navegador la siguiente dirección *http://localhost/cgi-bin/validar.cgi* .La salida se observa en la Figura 2.3:

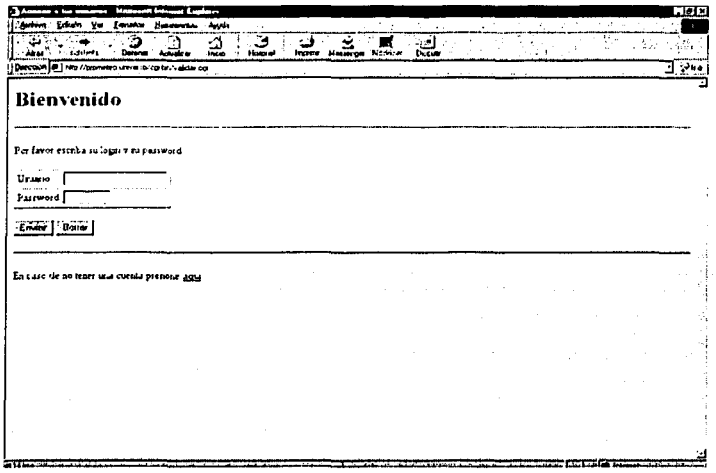


Fig 2.3 Formulario de entrada al sistema

Ahora el script que valide a los usuarios del sistema: *entrar.cgi*

```
#!c:\Perl\bin\Perl.exe

use CGI;

$queryv = new CGI;

print $queryv->header;
        $Usuario=$queryv->param("Usuario");
        $Password=$queryv->param("Password");

if ($Usuario eq "" || $Password eq "" ){
                &Error
}else{
        $patron=$Usuario."=".$Password;
        open(CONSULTA,"<c:\\usuarios.dat");
        while (<CONSULTA>) {
                if (/$patron/) {
                                print $queryv->start_html("Bienvenido al sistema"),
                                $queryv->h1("Bienvenido $Usuario"),
                                $queryv->hr,
                                $queryv->end_html;
                }
                else {
                        &Error;
                    }
        }close(CONSULTA);
}
```

```
sub Error{
        print $queryv->start_html("No tiene acceso al sistema"),
        $queryv->h1(("Usuario o Contraseña no validas, por favor vuelva a intentarlo"),$queryv->a({ -
href=>'/cgi-bin/validar.cgi'}, "aqui")),
        $queryv->hr,
        $queryv->end_html;
}
```

Existe en el script una subrutina que envía un error, este error aparecerá cuando el usuario no este registrado o haya olvidado incluir algún dato (usuario ó password) y regresará a la página inicio.

Los passwords de los usuarios están en el archivo usuarios.dat y por lo que la estructura de este archivo debe ser la siguiente:

```
jclp77=1a2b3c4d,mont9139=2761,jc_2000=2000ac,
```

El usuario, signo de igual, password y por último una coma.

El tercer paso es un formulario en caso de que el usuario no este registrado:

forma.html

```
<HTML><HEAD>
<TITLE>Registro de Usuarios</TITLE>
</HEAD>
<H1>Por favor llene los campos del formulario</H1>
<HR>
<FORM NAME=forma_01 METHOD=post ACTION='/cgi-bin/nuevo.cgi'>
 <TABLE>
  <TR>
   <TH>
    <div align="left"><FONT FACE=ARIAL SIZE= 3> Nombre: </FONT> </div>
   </TH>
   <TD><INPUT TYPE=text NAME=nombre VALUE='' MAXLENGTH=20 SIZE=20></TD>
  <TR>
   <TH>
    <div align="left"><FONT FACE=ARIAL SIZE= 3> Apellido Paterno: </FONT>
    </div>
   </TH>
   <TD><INPUT TYPE=text NAME=appat VALUE='' MAXLENGTH=20 SIZE=20></TD>
  <TR>
   <TH>
    <div align="left"><FONT FACE=ARIAL SIZE= 3> Usuario: </FONT> </div>
   </TH>
   <TD><INPUT TYPE=text NAME=usuario VALUE='' MAXLENGTH=20 SIZE=20></TD>
  <TR>
   <TH>
    <div align="left"><FONT FACE=ARIAL SIZE= 3> Password: </FONT> </div>
   </TH>
   <TD><INPUT TYPE=password NAME=password VALUE='' MAXLENGTH=8 SIZE=8></TD>
```

```
  <TR>
   <TH>
    <div align="left"><FONT FACE=ARIAL SIZE= 3> Confirmar Password: </FONT>
    </div>
   </TH>
   <TD><INPUT TYPE=password NAME=confirmar_password VALUE=" MAXLENGTH=8></TD>
  <TR>
   <TH>
    <div align="left"><FONT FACE=ARIAL SIZE= 3> e-mail: </FONT> </div>
   </TH>
   <TD><INPUT TYPE=text NAME=mail VALUE=" MAXLENGTH=30 SIZE=30></TD>
 </TABLE>
<BR><BR>
<INPUT TYPE=submit VALUE='Enviar' NAME=enviar>
<INPUT TYPE=reset VALUE='Borrar' NAME=borrar>
</FORM><HR>
</BODY></HTML>
```

Y su pantalla se muestra en la figura 2.4:

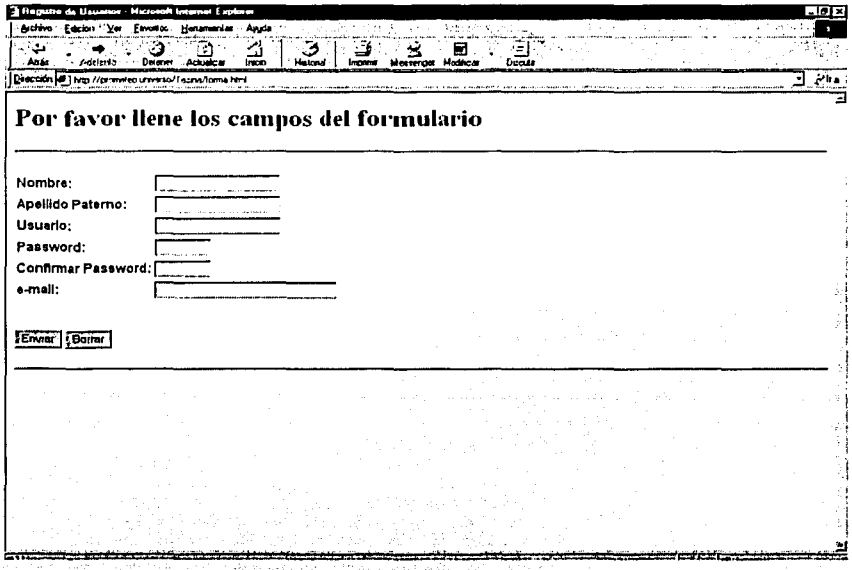


Fig 2.4 Datos para inscribirse al sistema

Por último el script que almacena la información del usuario: *nuevo.cgi*

```
#!c:\Perl\bin\Perl.exe

use CGI;

$queryn= new CGI;
```

```
$nombre=$queryn->param('nombre');
$apellido_paterno=$queryn->param('appat');
$usuario=$queryn->param('usuario');
$password=$queryn->param('password');
$confirmar_password=$queryn->param('confirmar_password');
$email=$queryn->param('mail');

if ($password eq $confirmar_password) {
        open(REGISTRO,'>>c:\\usuarios.dat');
        print REGISTRO "$usuario=$password,";
        close REGISTRO;
        open (DATOS,'>>c:\\datosgen.dat');
        print DATOS "Nombre=$nombre\nApellido
Paterno=$apellido_paterno\nUsuario=$usuario\nPassword=$password\nEmail=$email\n";
        close DATOS;
        print   $queryn->header,
                        $queryn->start_html("Registro"),
                        $queryn->h1("Gracias por registrase"),
                        $queryn->hr,
                        $queryn->p("Para entrar al sistema presione",
                        $queryn->a({-href=>'/cgi-bin/validar.cgi'}, "aqui")),
                        $queryn->br,
                        $queryn->hr,
                        $queryn->end_html;
}else{
        print   $queryn->header,
                        $queryn->start_html("Registro"),
                        $queryn->h1("Confirme su password"),
                        $queryn->hr,
                        $queryn->p("Para regresar al registro presione",
                        $queryn->a({-href=>'../Tesina/forma.html'}, "aqui")),
                        $queryn->br,
                        $queryn->hr,
                        $queryn->end_html;
}
```

El proceso de este script es el siguiente: primero almacena la información del usuario en variables y para evitar posibles errores de password se coloco un campo de confirmación en el formulario, así sólo cuando el password del usuario es confirmado este se almacena en el archivo usuarios.dat y la información restante en otro archivo de datos. En caso de que el usuario se haya equivocado en la confirmación de su password, será avisado enviándole una página de error.

Estos ejemplos, se pueden complementar con algunas rutinas javascript, para controlar algunos datos o a través de scripts CGI más complejos y robustos.

Las bases de datos almacenan datos. Estos son representaciones de sucesos y objetos a diferente nivel existentes en el mundo real, y en su conjunto representan algún tipo de entidad existente. En el mundo real se tiene percepción sobre las entidades u objetos y sobre los atributos de esos objetos; en el mundo de los datos hay registros de eventos y datos de eventos. Además, en ambos escenarios se puede incluso distinguir una tercera faceta: aquella que comprende las definiciones de las entidades externas, o bien las definiciones de los registros y de los datos.

3.1 MODELO RELACIONAL

Edgar F. Codd (1970), propone un modelo de datos basado en la Teoría de las Relaciones, donde los datos se estructuran lógicamente en forma de relaciones (tablas), siendo un objetivo fundamental mantener la independencia de la estructura lógica respecto al modelo de almacenamiento y a otras características del tipo físico.

3.1.1 Antecedentes

Modelo de datos Jerárquico

Una base de datos jerárquica consiste en una colección de registros que se conectan entre sí por medio de enlaces o ligas llamadas "punteros"; es decir, direcciones físicas dentro de la misma. Cada registro es una colección de campos que contiene un solo valor en cada uno de ellos.

En este tipo de modelos la organización se establece en forma de árbol, donde la raíz es un nodo ficticio. Así, una base de datos jerárquica es una colección de árboles de este tipo (Figura 3.1).

Fig 3.1 Modelo de datos Jerárquico

El contenido de un registro específico puede repetirse en varios sitios (en el mismo árbol o en varios árboles). Esta repetición de los registros tiene dos desventajas principales:

- Puede producirse una inconsistencia de datos.
- El espacio de almacenamiento es desperdiciado.

Un diagrama de estructura de árbol es la representación de un esquema de la base de datos jerárquica, de ahí el nombre, ya que un árbol está desarrollado en orden descendente formando una estructura jerárquica.

Este tipo de diagramas está formado básicamente por dos componentes:

- Rectángulos: que representan a los registros.
- Líneas: que representan a los enlaces o ligas entre los registros.

Para buscar información de un nivel intermedio en el árbol, se tiene que recorrer la estructura empezando por el nodo padre y en forma descendente hasta que se encuentre la información deseada. Es un procesamiento TOP-DOWN.

Algunos manejadores comerciales de este modelo son: IMS de IBM, Mark IV de Informatics, TOTAL de Circom, SYSTEM 2000 de Intel Corporation, etc.

Modelo de datos de Red

El modelo de red intenta superar las deficiencias del enfoque jerárquico, permitiendo el tipo de relaciones de muchos a muchos. Una base de datos de red, esta formada por una colección de registros, los cuales están conectados entre sí por medio de enlaces.

Una estructura de datos de red llamada algunas veces estructura plex, abarca más que la estructura de árbol porque un nodo hijo en la estructura de red puede tener más de un padre. En otras palabras, la restricción de que en un árbol jerárquico cada hijo puede tener un solo padre, se hace menos severa (Figura 3.2)

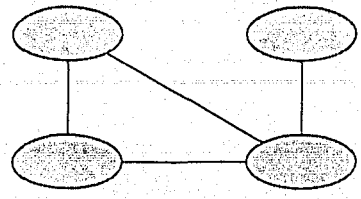


Fig 3.2 Modelo de datos de Red

Existen relaciones en las que participan sólo dos entidades (binarias) y relaciones en las que participan más de dos entidades (generales) ya sea con o sin atributo descriptivo en la relación.

La forma del diagramado consta de dos componentes básicos:

- Celdas: representan a los campos del registro.
- Líneas: representan a los enlaces entre los registros.

Es muy similar a una estructura jerárquica, de hecho no es más que un superconjunto de ésta. Algunas de las ventajas de este modelo consisten en que cualquier tipo de relación de registros puede ser modelada, además, se puede recorrer la estructura no necesariamente desde un punto inicial.

Los manejadores comerciales del modelo de red son: ADABAS, IDMS de Cullinet, DMS 1100, IMF, etc.

Modelo de datos Entidad-Relación

El modelo Entidad/Relación es un modelo de datos semántico cuyo objetivo inicial era vencer algunas de las dificultades mostradas por el modelo relacional, al que pretendía sustituir. Concretamente, pretendía dotar de "significado" a las estructuras de datos carentes del mismo modelo relacional.

En la práctica, este modelo de datos no ha llegado a implementarse en ningún DBMS (Sistema Manejador de Bases de Datos) comercial, pero ha tenido una enorme repercusión como herramienta de modelado de bases de datos (paradójicamente de bases de datos relacionales), existiendo hoy en día herramientas de diseño conceptual que

incorporan la totalidad de sus conceptos e incluso productos que transforman diagramas conceptuales E/R en bases de datos reales en diversos formatos.

El modelo E/R se basa en una percepción del mundo real, la cual esta formada por objetos básicos llamados entidades y las relaciones entre estos objetos así como las características de estos objetos llamados atributos.

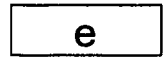
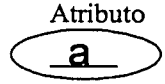
Una entidad es un objeto que existe y se distingue de otros de acuerdo a sus características llamadas atributos, en ocasiones llamadas propiedades. Los atributos de una entidad pueden tomar un conjunto de valores permitidos al que se le conoce como dominio del atributo. Así cada entidad se describe por medio de un conjunto de parejas formadas por el atributo y el valor de dato.

Una relación es la asociación que existe entre dos o más entidades. Un conjunto de relaciones es un grupo de relaciones del mismo tipo.

La cantidad de entidades en una relación determina su grado, por ejemplo la relación ALUMNO-ASIGNATURA es de grado dos, ya que intervienen la entidad ALUMNO y la entidad ASIGNATURA; la relación familia puede ser de grado tres, ya que involucra a las entidades PADRE, MADRE e HIJO.

La simbología utilizada para construir diagramas entidad-relación se integra con los siguientes componentes (Tabla 25):

TABLA 25. SIMBOLOGÍA DEL MODELO ENTIDAD-RELACIÓN

Descripción	Símbolo
Rectángulos: representan conjuntos de entidades	Entidad 
Elipses: representan atributos	Atributo 
Rombos: representan conjuntos de relaciones	Relación 
Líneas: conectan los atributos a los conjuntos de entidades, y los conjuntos de relaciones	Conexión 

Una llave primaria es identificada gráficamente en el modelo E-R con una línea debajo del nombre del atributo. Finalmente, un rectángulo doble significa que esa entidad es dependiente o débil, es decir, su existencia depende de la existencia de otra entidad. En algunos diagramas E/R el rombo que indica la relación entre una entidad independiente y otra dependiente también aparece con líneas dobles.

Existen cuatro tipos de relaciones que pueden establecerse entre entidades, las cuales establecen con cuantas entidades de tipo B se pueden relacionar una entidad de tipo A.

- Relación uno a uno. Se presenta cuando existe una relación uno a uno. Una entidad del tipo A sólo se puede relacionar con una entidad del tipo B, y viceversa.

A: Representa a una entidad de cualquier tipo diferente a una entidad B

R: Es la relación que existe entre las entidades

El extremo de la flecha que se encuentra punteada indica el uno de la relación. Es representado gráficamente (Figura 3.3) de la siguiente manera:

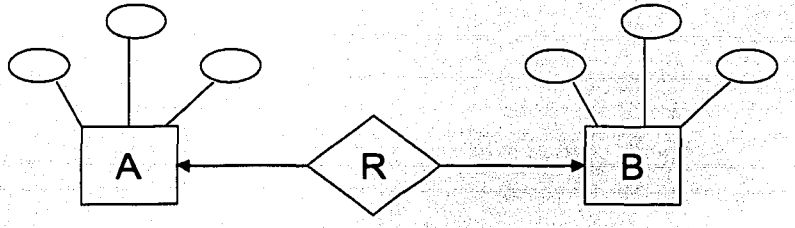


Fig 3.3 Relación uno a uno

- Relación uno a muchos. Significa que una entidad del tipo A puede relacionarse con cualquier cantidad de entidades del tipo B, y una entidad del tipo B solo puede estar relacionada con una entidad del tipo A.

Su representación gráfica (Figura 3.4) es la siguiente:

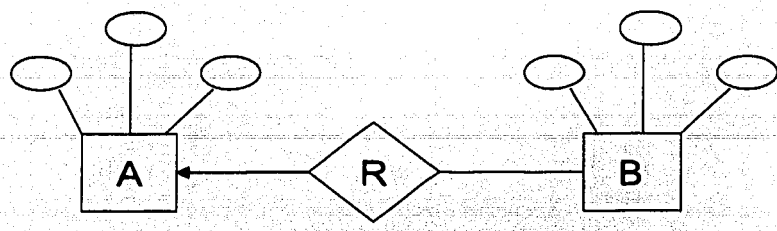


Fig 3.4 Relación uno a muchos

En este caso el extremo con punta de flecha de la relación de A y B, indica una entidad A conectada a muchas entidades B

- Relación muchos a uno. Una entidad del tipo B puede relacionarse con cualquier cantidad de entidades del tipo A, mientras que cada entidad del tipo A sólo puede relacionarse con una entidad del tipo B (Figura 3.5).

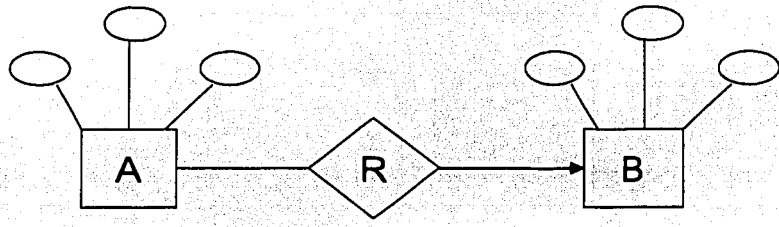


Fig 3.5 Relación muchos a uno

- Relación muchos a muchos. Establece que cualquier cantidad de entidades del tipo A pueden estar relacionadas con cualquier cantidad de entidades del tipo B. El gráfico que representa esta relación es la figura 3.6:

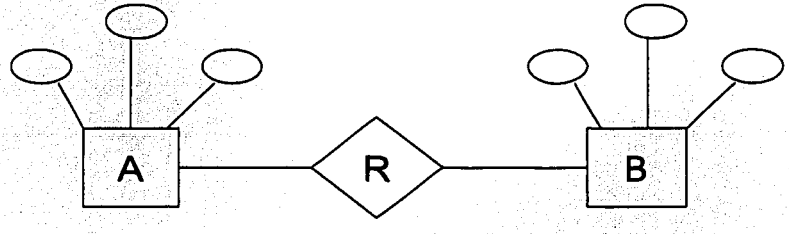


Fig 3.6 Relación muchos a muchos

La cardinalidad especifica los tipos de relaciones que existen entre las entidades en el modelo E-R y establecer con esto, las validaciones necesarias para conseguir que los datos de la instancia correspondan con la realidad.

Algunos ejemplos de cardinalidades: El RFC de cada persona (uno a uno), Autobús y Pasajeros (uno a muchos), fiesta y personas (muchos a muchos).

La distinción de una entidad de otra se debe a sus atributos, lo cual lo hacen único. Una llave primaria es aquel atributo el cual consideremos clave para la identificación de los demás atributos que describen a la entidad. Por ejemplo, la entidad ALUMNO podría tener los atributos de Nombre, Apellido Paterno, Apellido Materno, Dirección, Teléfono, Número de Cuenta, Semestre, y de todos estos atributos el que puede ser una llave primaria es NUMERO DE CUENTA, ya que es diferente para cada alumno.

Puede existir más de un atributo que puede identificarse como llave primaria, en este caso, se selecciona la que se considere más importante, y los demás atributos son denominados llaves secundarias.

3.1.2 Modelo del Dr. Edgar F. Codd

El modelo relacional de datos supuso un gran avance con respecto a los modelos anteriores (jerárquico y de red). Este modelo está basado en el concepto de *relación*. Una relación es un conjunto de *n*-tuplas. Una tupla puede representar tanto entidades como interrelaciones.

Las estructuras de datos que se manejan en el modelo relacional corresponden a los conceptos de relación, entidad, atributo y dominio, los cuales se explican a continuación:

- Relación. Por una relación se entiende una colección o grupo de objetos que tienen en común un conjunto de características o atributos.

- Entidad. Es una unidad de datos en una relación con un conjunto finito de atributos. Es también conocido como n-ada, a raíz de que consiste de n-valores, uno por cada atributo.

- Atributo. También llamado característica, cada atributo de una relación tiene asociado un dominio en el cual toma sus valores.

- Dominio. Es un conjunto de valores que puede tomar un atributo en una relación.

La ventaja del modelo relacional consiste en que los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios lo entienden con mayor facilidad.

Algunas propiedades de las bases de datos relacionales:

- No redundancia: Cada evento sólo debe almacenarse una vez, en las bases de datos.

- Consistencia: La base de datos debe ser consistente bajo las operaciones permitidas de manipulación de los datos.

- Flexibilidad: La estructura de la base de datos debe ser fácil de modificar.

- Integridad: La base de datos debe contener toda la información relacionada con un evento.

- Interrelación: Los distintos datos de la base de datos pueden interrelacionarse.

- Escalabilidad: Las tablas pueden crecerse ilimitadamente y pueden definirse nuevas tablas para garantizar el desarrollo futuro del sistema.

A continuación la definición matemática del modelo relacional según Edgar F. Codd:

Dada una serie de conjuntos S_1, S_2,..., S_n (no necesariamente distintos), se dice que R es una relación sobre estos n conjuntos si es un conjunto de n tuplas cada una de las cuales tiene su primer elemento para S_1, su segundo elemento para S_2, y así sucesivamente. Nos referiremos a S_j como el jth dominio de R. Como se definió, R es un conjunto que tiene grado n. Las relaciones de grado 1 son llamadas como unarias, de grado 2 binarias, de grado 3 ternarias, y de grado n n-arias.

Es conveniente representar una relación en forma de tabla (para un mayor entendimiento) y además tiene las siguientes características:

- Cada renglón de la tabla representa una n tupla de R.
- El orden de los renglones es irrelevante, porque una relación es un conjunto y los conjuntos no son ordenados.
- Todos los renglones son distintos.
- El orden de las columnas significa el orden de S1, S2,..., Sn de los dominios que están definidos en R.
- El significado de cada columna se cambia parcialmente por el nombre de su dominio correspondiente.
- El número de tuplas de una relación se llama cardinalidad de la relación.

En una base de datos relacional se requiere que todas las relaciones cumplan la condición siguiente: *Que todo valor de la relación sea atómico*. En otras palabras, en cada intersección de un renglón y una columna de la tabla siempre hay exactamente un valor, nunca un conjunto de valores. A continuación se muestra la relación Alumno (Figura 3.7).

Atributo 1	**Atributo 2**	**Atributo 3**	**Atributo n**	
Numero de cuenta	Nombre	Apellido Paterno	...	Tupla 1
X	X	X	...	Tupla 2
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
X	X	X	...	Tupla n

Fig 3.7 Relación Alumno.

Normalmente, un dominio (o una combinación de dominios) de una relación dada tiene valores únicos que identifican a cada elemento (n-tupla) de esa relación. Tal que un dominio (o una combinación) es llamada llave primaria. En la Figura 3.8 la llave primaria

de la relación es el número de cuenta, ya que es el valor único que identifica a cada alumno.

(Llave primaria)	**Atributo 2**	**Atributo 3**	**Atributo n**	
Numero de cuenta	Nombre	Apellido Paterno	...	Tupla 1
X	X	X	...	Tupla 2
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
X	X	X	...	Tupla n

Fig 3.8 Llave primaria en una relación

Cuando una relación tiene dos o más atributos que pueden ser identificados como llaves primarias, se selecciona una arbitrariamente y esa será llamada llave primaria de la relación.

Es un requerimiento común de los elementos de una relación hacer referencia a otros elementos de la misma relación o a una relación diferente. Las llaves proveen un medio de orientación para el usuario de expresar tal referencia. Un domino (o una combinación de dominios) de una relación R se llamará llave secundaria (foreign key) si no es la llave primaria de R pero sus elementos son valores de la llave primaria de alguna relación S.

Un *valor nulo* es un valor que está fuera de la definición de cualquier dominio el cual permite dejar el valor del atributo "*latente*", su uso es frecuente en las siguientes situaciones:

i) Cuando se crea una *n-ada* y no se conocen todos los valores de cada uno de los atributos.

ii) Cuando se agrega un atributo a una relación ya existente.

iii) Para no tomarse en cuenta al hacer cálculos numéricos.

Las dos *reglas de integridad* tienen que ver precisamente con los conceptos antes mencionados y son las siguientes:

— Regla de integridad 1: Integridad de Relaciones. Ningún atributo que forme parte de una llave primaria puede aceptar valores nulos.

Es decir, todas las entidades deben ser distinguibles por definición. Las llaves primarias realizan la función de identificación única en la base de datos relacional. Un identificador (valor de la llave primaria) que fuera totalmente nulo sería una contradicción de términos; entonces equivaldría a decir que hubo alguna entidad que no tuvo ninguna identificación única. Es común que una relación incluya referencias a otras.

Un dominio específico puede designarse como primario si y sólo si existe alguna llave primaria de un solo atributo definida sobre ese dominio. Entonces, cualquier relación que incluya un atributo que se defina sobre un dominio primario debe obedecer a la siguiente regla:

— Regla de integridad 2: Integridad Referencial. Sea D un dominio primario, y sea R_1 una relación con un atributo A que se define sobre D. Entonces, en cualquier instante dado, cada valor de A en R_1 debe ser:

1. Nulo o
2. Igual a V; donde V es el valor de la llave primaria de alguna tupla de alguna relación en R_2 (R_1 y R_2 no son por fuerza distintas) con llave primaria definida sobre D.

Además de las restricciones impuestas por las reglas generales del modelo relacional, y de las reglas específicas impuestas por el DBA (Administrador de la Base de Datos) para una base de datos, es conveniente la observación de otras "reglas" que reforzaran el modelo además ayudaran a mantener la integridad de los datos y a evitar la redundancia. Esto es lo que se conoce como normalización.

Existen tres formas normales básicas, expuestas por Codd en la primera versión del modelo, conocidas como 1NF, 2NF y 3NF, respectivamente, más otras tres que fueron añadidas con posterioridad (BCNF, 4NF y PJ/NF ó 5NF). En realidad, BCNF (la forma normal de Boyce/Codd) no es más que un intento de tapar los huecos de 3NF, y durante un tiempo fue llamada simplemente 3NF. Las dos restantes, 4NF y PJ/NF, no fueron definidas por Codd, sino por R. Fagin. A continuación se muestra las tres formas normales básicas (las expuestas por Codd), pues la exposición de las tres posteriores

implica un estudio previo necesario para comprenderlas, y se halla fuera de los intereses de este trabajo.

1. 1NF : Una relación R está en primera forma normal (1NF) si y sólo si todos los dominios simples subyacentes contienen únicamente valores atómicos.

2. 2NF : Una relación R está en segunda forma normal (2NF) si y sólo si R está en 1NF y además todos los atributos no clave (es decir, los que no forman parte de la clave primaria) dependen por completo de la clave primaria.

3. 3NF : Una relación R está en tercera forma normal (3NF) si los atributos no clave (si los hay) son:

a. mutuamente independientes, y

b. dependientes por completo de la clave primaria

3.1.3. Álgebra Relacional

La parte dinámica del modelo, propone un conjunto de operadores que se aplican a las relaciones. Algunos de estos operadores son clásicos en la teoría de conjuntos (una relación es un conjunto) mientras que otros fueron introducidos específicamente para el modelo relacional.

Existen dos tipos básicos de formalismos para expresar las consultas sobre las relaciones de una base de datos relacional: el álgebra relacional y el cálculo relacional. El cálculo relacional es complejo y se necesitan más conocimientos como el cálculo de predicados y la lógica proposicional.

- Unión

El operador de unión es el operador generalmente utilizado para unir dos relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$ en donde la unión sea compatible. Dos relaciones son llamadas uniones compatibles si las dos relaciones son del mismo grado *m*, y tienen el mismo encabezado, entonces $\mathbf{R_1 \cup R_2}$ ($\mathbf{R_1}$ UNION $\mathbf{R_2}$) consiste en todas las tuplas que están tanto en $\mathbf{R_1}$ ó en $\mathbf{R_2}$; o en ambos.

- Intersección

El operador de intersección es el tradicional conjunto de intersecciones de dos relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$ compatibles, $\mathbf{R_1}$ INTERSECCION $\mathbf{R_2}$ ($\mathbf{R_1 \cap R_2}$), la intersección provee una relación consistente en todas las tuplas que tienen en común ambas relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$. El grado de $\mathbf{R_1 \cap R_2}$, es el mismo de $\mathbf{R_1}$ y $\mathbf{R_2}$ y la cardinalidad es la misma al número de tuplas en común de $\mathbf{R_1}$ y $\mathbf{R_2}$.

- Diferencia

El operador de diferencia cuando es aplicado a dos relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$ (escrito como $\mathbf{R_1\text{-}R_2}$) da como resultado una relación que consiste en todas las tuplas en la primera relación que no se encuentran en la segunda relación. Si $\mathbf{R_1 \cap R_2}$ es nulo entonces $\mathbf{R_1\text{-}R_2=R_1}$ de otra manera, $\mathbf{R_1\text{-}R_2 = R_1 - R_1 \cap R_2}$. El grado de $\mathbf{R_1\text{-}R_2}$ es el mismo que $\mathbf{R_1}$ y $\mathbf{R_2}$, y su cardinalidad es igual a la cardinalidad de $\mathbf{R_1}$ menos la cardinalidad de $\mathbf{R_1 \cap R_2}$.

- División

El concepto de división está relacionado con el producto cartesiano en el que $\mathbf{R_1}$ **x** $\mathbf{R_2}$ dividido por $\mathbf{R_2}$ da como resultado $\mathbf{R_1}$. Dividir sin embargo, es esencialmente el inverso del producto cartesiano.

La división de dos relaciones es otra relación cuya extensión estará constituida por las tuplas que al complementarse con las tuplas de la segunda relación permiten obtener la primera.

- Producto cartesiano

El producto cartesiano de dos relaciones es el conjunto de pares ordenados de tuplas. Sin embargo, para propósitos del modelo relacional, se necesitan simplemente las tuplas, y no un par ordenado de ellas; para ello la versión del producto cartesiano en el álgebra relacional es una extensión de la operación en donde cada par ordenado de tuplas es reemplazada por una sola tupla a esto se le llama *coalición* de dos tuplas en cuestión.

Formalmente podemos definir al producto cartesiano de dos relaciones $\mathbf{R_1}$ (de grado m) y $\mathbf{R_2}$ (de grado n), da una relación $\mathbf{R_1}$ x $\mathbf{R_2}$ de grado $m + n$. Esta relación de producto tiene todas los atributos que están presentes en la relación $\mathbf{R_1}$ y $\mathbf{R_2}$ y las tuplas

en $\mathbf{R_1}$ x $\mathbf{R_2}$ son todas las posibles combinaciones de tuplas de $\mathbf{R_1}$ y $\mathbf{R_2}$. La cardinalidad de $\mathbf{R_1}$ x $\mathbf{R_2}$ entonces es *ab* si *a* es la cardinalidad de $\mathbf{R_2}$ y b de $\mathbf{R_1}$. El grado de $\mathbf{R_1}$ x $\mathbf{R_2}$ es $x + y$ si x y y son grados de $\mathbf{R_1}$ y $\mathbf{R_2}$, respectivamente.

- Proyección(π)

La operación de Proyección es la selección de ciertos atributos de una relación $\mathbf{R_1}$ para formar una nueva relación $\mathbf{R_2}$, eliminando aquellas tuplas que se encuentren duplicadas. La proyección de R_1 en los atributos *a*, *b*, *c*, está denotada por **R [a, b, c]**, algunos autores utilizan **π** a, b, c [R] para denotar la proyección, para generalizar esta notación tenemos **π s(R)** donde R es la relación y *s* la lista de atributos a proyectar.

- Selección(σ)

La operación de seleccionar ciertas tuplas de una relación es llamada selección. Usualmente se seleccionan tuplas que satisfagan cierta condición. La selección es una operación unitaria ya que opera sobre una sola relación.

Formalmente, la selección de una relación $\mathbf{R_1}$ es un subconjunto de la relación $\mathbf{R_2}$ de las tuplas en $\mathbf{R_1}$, donde esas tuplas seleccionadas satisfacen cierta condición. El grado de $\mathbf{R_2}$ es el mismo que el de $\mathbf{R_1}$. Y la cardinalidad de $\mathbf{R_2}$ es igual al número de tuplas en $\mathbf{R_1}$ que satisfagan una condición específica.

La restricción de una relación $\mathbf{R_1}$ cuyo atributo $\boldsymbol{A}$ tenga como valor $\boldsymbol{a}$ está denotada por la siguiente expresión **σ A =a [$\mathbf{R_1}$]**. Para generalizar esta expresión se utilizará **σ p [R]** donde **R** es la relación con un predicado $\boldsymbol{p}$.

- Join(⊖)

El "join" de dos relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$ es una restricción de sus productos cartesianos $\mathbf{R_1}$ x $\mathbf{R_2}$ tal que una condición específica sea satisfecha. El join se define normalmente sobre un atributo *a* de $\mathbf{R_1}$ y un atributo *b* de $\mathbf{R_2}$ tal que los atributos sean del mismo dominio y se encuentren relacionados.

El join que se utiliza comúnmente es aquel en el cual la condición especificada es equivalente a los dos atributos (uno de cada relación) $\mathbf{R_1}$ x $\mathbf{R_2}$. Este join es llamado:

Equi-join. El *equi-join* por definición tiene dos atributos (columnas) una de esas pueden ser removida utilizando el operador de proyección. El resultado de esta operación es llamado *join natural.*

- Join Natural

El join natural de dos relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$ es obtenido aplicando una selección y proyección al producto cartesiano de $\mathbf{R_1}$ x $\mathbf{R_2}$ de la siguiente manera:

1. Por cada atributo ***a*** que tengan en común ambas relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$, seleccionaremos las tuplas que satisfagan la condición $\mathbf{R_1.a = S_1.a}$.

2. Por cada atributo ***a*** que tengan en común ambas relaciones $\mathbf{R_1}$ y $\mathbf{R_2}$, proyectaremos la columna $\mathbf{R_2.a}$

El grado de un equi- join es $x + y$ si el grado de $\mathbf{R_1}$ y $\mathbf{R_2}$ son x y y respectivamente, para que las operaciones de *equi — join* y *join natural* de dos relaciones puedan efectuarse es necesario que ambas relaciones tengan un atributo que sea del mismo dominio.

Para crear las relaciones, modificarlas, eliminarlas, recuperar los datos almacenados en ellas, y para manipularlas en general, se necesita un lenguaje formal que facilite el acceso, de lo contrario se debería trabajar a bajo nivel, o nivel de máquina. Este lenguaje debe ser lo suficientemente expresivo para permitir llevar a cabo todas estas operaciones, y debe estar basado en formalismos que cumplan con todas las premisas expuestas anteriormente respecto a reglas de integridad, formas normales, etc.

3.2 SQL

"El SQL surge originalmente con el nombre de SEQUEL (Structured English QUEry Language) instrumentado en un prototipo de IBM, el SEQUEL-XRM, durante los años 1974-1975. Posteriormente se introduce en el SISTEMA R, y tras una serie de modificaciones pasa a denominarse SEQUEL II, para convertirse más tarde, por cuestiones legales, en el SQL." [CASTAÑO Y PIATTINI, 1993, PÁG 491]

El lenguaje de consulta de bases de datos relacionales es llamado SQL (*Structured Query Language*). Este lenguaje esta basado en el álgebra relacional y el cálculo relacional anteriormente descritos, actúa de interfaz entre el usuario y la base de datos y

facilita realizar todas las operaciones permitidas. El lenguaje fue diseñado para que, mediante un número muy reducido de comandos y una sintaxis simple, fuese capaz de realizar un gran número de operaciones.

La curva de aprendizaje de SQL es realmente rápida. Además, SQL es bastante flexible; en el sentido de que las sentencias SQL pueden ser anidadas indefinidamente dentro de otras sentencias SQL, facilitando así las consultas que utilizan varias relaciones, vistas u otras consultas.

El lenguaje SQL es un estándar, pero cada manejador de base de datos tiene sus propios agregados. Para el caso de la aplicación de este trabajo (Parte IV), se utilizará el manejador de Base de Datos Relacional: MySQL (ver Anexo C) este manejador es un software de código abierto, esto quiere decir que es accesible para cualquiera, para usarlo o modificarlo. Por lo que es conveniente introducir la estructura de las sentencias SQL que utiliza este manejador.

3.2.1 Sentencias de Definición

El lenguaje de definición de datos (DDL) es la parte del SQL que más varía de un sistema a otro ya que tiene que ver con la organización interna de los datos, cada sistema lo hace de una manera u otra.

— **Sentencia CREATE DATABASE**

En MySQL, la sentencia CREATE DATABASE esencialmente crea un nuevo directorio que almacenará las objetos de la base de datos, esta será creada como un directorio sobre el directorio del RDBMS (Sistema Manejador de Bases de Datos Relacionales) y cualquiera de los nuevos objetos que sean creados son puestos en ese fólder

Sintaxis MySQL

CREATE DATABASE database_name

— **Sentencia CREATE TABLE**

Este comando define el nombre de una tabla, las columnas de elementos, y cualquier propiedad de las columnas y/o tabla.

En general, el nombre de la tabla siempre empieza con un carácter alfabético. La longitud permitida del nombre depende del RDBMS. Los números pueden ser utilizados en el nombre de la tabla, pero no se puede utilizar cualquier otro símbolo a menos que sea el carácter underscore (_).

Algunos RDBMS soportan la declaración de llave primaria (PRIMARY KEY). Una llave primaria es una designación especial que describe a cada renglón de una tabla como es identificado únicamente. La llave primaria esta compuesta de una o más columnas en la tabla que proporciona a cada renglón un identificador único.

Las llaves secundarias pueden ser declaradas en una tabla que establece una relación directa con una llave primaria en otra tabla. En esta forma, las relaciones padres/hijos ó maestro/detalle pueden crearse.

Varios RDBMS soportan el valor DEFAULT para una columna dada. En el momento en que un registro se inserta en la tabla y no tiene valor en la columna, el valor DEFAULT se inserta.

Para los datatype ó tipos de datos, MySQL maneja los tipos numericos, tipo cadenas, los tipo fecha y hora, etc. Para una mayor referencia acerca de los tipos de datos consultese la siguiente dirección electrónica:

http://www.mysql.com/documentation/mysql/bychapter/manual_Reference.html#Column_types

Sintaxis MySQL

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] table_name
(column_name datatype [NULL|NOT NULL] [DEFAULT default value]
    [AUTO_INCREMENT]
    [PRIMARY KEY] [reference_definition] |
    [CHECK (expression) |
    [INDEX [index_name] index_col_name1[(length)],...n)] |
    [UNIQUE [INDEX] [index_name] (index_col_name1,...n)] |
    [CONSTRAINT symbol] FOREIGN KEY index_name (index_col_name),...n)
        [REFERENCES table_name [(index_col_name,...)]
        [MATCH FULL |MATCH PARTIAL]
        [ON DELETE {RESTRICT|CASCADE|SET NULL|NO ACTION|SET DEFAULT}]
```

```
    [ON UPDATE {RESTRICT|CASCADE|SET NULL|NO ACTION|SET DEFAULT}]
{[TYPE={ISAM|MYISAM|HEAP} |
    AUTO_INCREMENT=int |
    AVG_ROW_LENGTH=int |
    CHECSUM={0|1} |
    COMMENT="string" |
    DELAY_KEY_WRITE={0|1} |
    MAX_ROWS=int |
    MIN_ROWS=int |
    PACK_KEYS={0|1} |
    PASSWORD="string"|
    ROW_FORMAT={default|dynamic|static|compressed}]}
[IGNORE |REPLACE]
SELECT_statement
```

MySQL tiene opciones sobresalientes para crear tablas. La opción TEMPORARY crea una tabla que persiste durante la conexión bajo la cual fue creada. Cuando la conexión es cerrada, la tabla temporal es eliminada automáticamente. La opción IF NOT EXIST prevé un error en caso de que la tabla ya exista.

Cuando se crea una tabla en MySQL, también se crean tres archivos de sistema: el archivo de la definición de la tabla con la extensión .frm, un archivo de datos con la extensión .myd y finalmente un archivo de índices con la extensión .myi.

La cláusula AUTO_INCREMENT prepara una columna de enteros e incrementa su valor en 1 (empezando con el valor de 1) MySQL sólo permite una columna de AUTO_INCREMENT por tabla. Cuando el valor máximo es eliminado, el valor es reutilizado. Cuando todos los registros son eliminados, el valor vuelve a empezar de nuevo.

Una columna o columnas con una llave primaria, puede definirse con la condición de que esos valores sean definidos como NOT NULL. Cuando la característica de índice se asigna a una columna, el nombre para el índice debe ser incluido. Si el nombre no se asigna, MySQL asigna un nombre de index_column_name con un sufijo numérico (_2,_3,...) para hacerlo único. Solamente el tipo de tabla MyISAM soporta los índices sobre columnas NULL ó sobre BLOB ó columnas con datos de tipo TEXT.

Las cláusulas FOREIGN KEY, CHECK y REFERENCES, no agregan funcionalidad a la tabla y sólo son soportados únicamente para la compatibilidad con otras bases de datos.

Las opciones de la tabla TYPE describen cómo los datos deben guardarse físicamente, ISAM es la tabla original por definición. MyISAM es relativamente nuevo, es una estructura de almacenamiento binaria más portable. El tipo HEAP almacena la tabla en memoria.

AUTO_INCREMENT define un valor de incremento para la tabla (solamente MyISAM).

AVG_ROW_LENGTH define un promedio de renglones aproximado estableciendo un promedio de longitud de las filas para la tabla con los registros de tamaño variable.

CHECKSUM cuando se establece para 1, mantiene una checksum para todos los renglones en la tabla (solamente MyISAM). Este hace el procesamiento lento, pero menos propenso a la corrupción de datos.

COMMENT permite un comentario de hasta 60 caracteres.

MAX_ROWS Establece un número máximo de renglones para el almacenamiento en la tabla. El valor máximo por default es 4 GB de espacio en disco.

MIN_ROWS establece un número mínimo de renglones para el almacenamiento en la tabla.

PACK_KEYS cuando se establece a 1, compacta los índices de la tabla haciendo que se puedan leer más rápido pero las actualizaciones son más lentas (solamente MyISAM e ISAM).

PASSWORD encripta el archivo .frm con un password, pero no encripta la tabla.

ROW_FORMAT determina cómo deben guardarse los futuros renglones en la tabla.

La cláusula SELECT_statement crea una tabla cuyos campos son basados en los elementos en la sentencia SELECT.

Por ejemplo:

```
CREATE TABLE test_example
 (colum_a INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY (column_a),
 INDEX (column_b))
 TYPE=HEAP
 SELECT column_b, column_c FROM samples;
```

Esto crea una tabla en memoria con tres columnas: column_a, column_b y column_c.

— **Sentencia CREATE INDEX**

Los índices son objetos especiales construidos sobre las tablas y agilizan las operaciones de manipulación, como las sentencias SELECT, UPDATE y DELETE. Cuando un índice se crea, la localización y la extensión de valores son construidos para la columna que será indexada.

Sintaxis MySQL

```
CREATE [UNIQUE] INDEX index_name ON table_name
      (column_name (length)[,...n])
```

MySQL soporta el estándar básico de ANSI para la sentencia CREATE INDEX, incluyendo la habilidad para construir un índice en múltiples columnas. Un índice puede ser definido como UNIQUE, forzando al índice a aceptar solamente valores únicos. Cualquier inserción de un valor no único en la tabla con un índice UNIQUE se rechaza.

Todos los objetos de la base de datos creados con el comando CREATE pueden ser eliminados usando las sentencias complementarias DROP. El comando DROP es irreversible y permanente, así que se debe utilizar con mucho cuidado.

— **Sentencia DROP DATABASE**

Este comando elimina la base de datos. Algunos RDBMS sólo permiten al administrador realizar esta tarea.

Sintaxis MySQL

```
DROP DATABASE database_name
```

— **Sentencia DROP TABLE**

Este comando elimina una tabla y todos sus datos, indices, triggers, constraints y permisos especificados para la tabla. Cualquier vista o procedimiento relacionado a la tabla eliminada tendrá problemas, a menos que sean alterados o eliminados.

Sintaxis MySQL

DROP TABLE [IF EXISTS] table_name

MySQL elimina la tabla completamente y permanentemente además de todos los archivos asociados. La sintaxis IF EXIST puede agregarse para evitar un error cuando se quiera eliminar la tabla y esta ya no exista.

— **Sentencia DROP INDEX**

El comando DROP INDEX elimina uno o más índices de la base de datos actual. Cuando un índice es eliminado todo el espacio previamente consumido por este, se recupera inmediatamente. DROP INDEX no realiza la destrucción de la llave primaria o constraints, esto debe hacerse con la sentencia ALTER TABLE... en la cláusula DROP.

Sintaxis MySQL

DROP INDEX table_name index_name [,...n]

MySQL permite eliminar múltiples índices separando cada tabla y nombre del índice con una coma.

— **Sentencia ALTER TABLE**

La sentencia ALTER TABLE permite a una tabla existente ser modificada eliminando la tabla o alterando permisos existentes.

Sintaxis MySQL

```
ALTER [IGNORE] TABLE table_name
[ADD [COLUMN] column_name datatype attributes]
[FIRST|AFTER column_name]][,...n]
| [ADD INDEX [index_name](index_col_name,...)][,...n]
| [ADD PRIMARY KEY (index_col_name,...)][,...n]
| [ADD UNIQUE [index_name] (index_col_name,...)][,...n]
| [ALTER [COLUMN] column_name {SET DEFAULT literal|DROP DEFAULT}][,...n]
| [CHANGE [COLUMN]old_col_name create_definition][,...n]
| [MODIFY COLUMN] column_name datatype attributes][,...n]
| [DROP [COLUMN] column_name][,...n]
| [DROP PRIMARY KEY][,...n]
| [DROP INDEX index_name][,...n]
| [RENAME [AS] new_table_name][,...n]
| [table_options]
```

La opción IGNORE elimina renglones duplicados cuando se define una llave primaria. Si no se especifica IGNORE, la operación se interrumpe cuando existe un registro duplicado en la llave primaria.

La opción FIRST se usa cuando se agrega una nueva columna, siendo la primera columna de la tabla. Cuando se utiliza AFTER se agrega una nueva columna en la tabla después de la column_name especificada.

En suma, MySQL admite funciones adicionales en la sentencia ALTER TABLE permitiendo a los usuarios formular múltiples sentencias ADD, ALTER, DROP y CHANGE en una sola sentencia ALTER TABLE.

3.2.2 Sentencia de Manipulación

El DML (Data Manipulation Language), lenguaje de manipulación de datos, permite recuperar los datos almacenados en la base de datos.

— . Sentencia SELECT

Sintaxis MySQL

```
SELECT [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] [HIGH_PRIORITY]
    [DISTINCT | DISTINCTROW | ALL]
 select_expression,...
  [INTO {OUTFILE | DUMPFILE} 'file_name' export_options]
  [FROM table_references
   [WHERE where_definition]
   [GROUP BY {unsigned_integer | col_name | formula} [ASC | DESC], ...]
   [HAVING where_definition]
   [ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC] ,...]
   [LIMIT [offset,] rows]
   [PROCEDURE procedure_name]
   [FOR UPDATE | LOCK IN SHARE MODE]]
```

La sentencia SELECT se usa para recuperar renglones seleccionados de una o más tablas. Select_expression indica las columnas que se quiere recuperar.

Las palabras clave (FROM, WHERE, GROUP BY, etc.) deben usarse exactamente en el orden mostrado anteriormente. Por ejemplo una cláusula HAVING debe estar después de cualquier cláusula GROUP BY y antes de cualquier cláusula ORDER BY.

Una expresión SELECT puede utilizar un alias usando la palabra reservada AS. El alias es utilizado como un nombre de una columna en una expresión y puede usarse con las cláusulas ORDER BY ó HAVING. Por ejemplo:

```
SELECT CONCAT(last_name,', ',first_name) AS full_name
FROM mytable ORDER BY full_name;
```

No se permite usar un alias de una columna en una cláusula WHERE, porque el valor de la columna no puede determinarse cuando se ejecuta la sentencia, ya que esta no existe lógicamente.

La cláusula FROM table_references indica las tablas para la recuperación de registros. Si el nombre es más de una tabla, es preferible utilizar un join.

Las columnas seleccionadas para la salida pueden referirse en las cláusulas ORDER BY y GROUP BY usando los nombres de columna, alias de columna o posiciones de las columnas. Las posiciones de columna empiezan con 1,Por ejemplo:

```
SELECT college, region, seed
FROM tournament
ORDER BY region, seed;

SELECT college, region AS r, seed AS s
FROM tournament
ORDER BY r, s;

SELECT college, region, seed
FROM tournament
ORDER BY 2, 3;
```

Se puede utilizar en la cláusula WHERE cualquier función que soporte MySQL. Para más información acerca de las funciones de MySQL consulte la documentación que esta en línea en la página *http://www.mysql.org.*

La cláusula HAVING se refiere a cualquier columna o nombre de alias en la select_expression. No se debe utilizar HAVING para los elementos que deben ser de la cláusula WHERE. Por ejemplo:

```
SELECT col_name FROM tbl_name HAVING col_name > 0;
```

En cambio si se escribe:

```
SELECT col_name FROM tbl_name WHERE col_name > 0;
```

Las opciones DISTINCT, DISTINCTROW y ALL especifican si los renglones duplicados son regresados. El valor por default es ALL, por lo que todos los registros son regresados.

DISTINCT y DISTINCTROW son sinónimos y especifican que los registros duplicados en el resultado de la consulta deben ser removidos de esta.

Todas las opciones que empiezan con SQL_, STRAIGHT_JOIN y HIGH_PRIORITY son extensiones de MySQL para el ANSI SQL. Para una mayor referencia consultar la documentación del manejador MySQL en el sitio antes mencionado.

Si se utiliza GROUP BY, los registros mostrados serán ordenados en grupos.

STRAIGHT_JOIN obliga a optimizar el join de tablas en el orden en que fueron listadas en la cláusula WHERE.

La cláusula LIMIT puede ser utilizado como si fuera un constraint del número de registros devueltos por la sentencia SELECT. LIMIT puede tener uno ó más argumentos numéricos. El argumento debe ser un número entero. Si se utilizan dos argumentos, el primero especifica el desplazamiento del primer registro a regresar. El segundo especifica el número máximo de registros a regresar. El desplazamiento del registro inicial es 0 (no 1), en otras palabras, LIMIT n es equivalente a LIMIT 0,n. Por ejemplo:

*SELECT * FROM table LIMIT 5,10; # Regresa los registros 6-15*

*SELECT * FROM table LIMIT 5; # Regresa los primeros 5 registros*

3.2.3 Sentencias de Actualización

— Sentencia INSERT

La sentencia INSERT agrega columnas de datos a una tabla o vista. La sentencia INSERT permite introducir los registros en una tabla a través de varios métodos:

- El primer método es insertar registros usando los valores creados por default en las columnas dadas de la tabla por las sentencias CREATE TABLE ó ALTER TABLE.
- El segundo y el método más común es declarando los valores actuales para ser insertados en cada columna del registro.
- El tercer método (qué rápidamente llena una tabla con muchos registros), es insertar el resultado de una sentencia SELECT en la tabla.

Sintaxis MySQL

INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] [[database_name.]owner]{table_name | view_name} [(column_list)]
{VALUES (value[,...]) | SELECT_statement | SET column=value [,...n]}

La opción LOW_PRIORITY especifica a MySQL que retrase la ejecución de inserción de datos hasta que los usuarios terminen de leer la información de la tabla. La

opción DELAYED permite al usuario que continué, aun cuando la inserción de datos todavía no se haya completado.

La palabra IGNORE indica a MySQL no intentar insertar registros que duplicarían el valor de alguna llave primaria o una llave única; de lo contrario, sin esta cláusula, la inserción falla.

La sintaxis SET column=value permite a las columnas de una tabla ser declaradas y los valores que se insertaran en ellas.

— **Sentencia DELETE**

La sentencia DELETE elimina los registros de una tabla específica o de varias tablas. Es una operación registrada, significando que puede deshacerse con el comando ROLLBACK (sólo en algunos manejadores de bases de datos).

Sintaxis MySQL

DELETE [LOW_PRIORITY] FROM table_name [WHERE clause] [LIMIT rows]

La opción LOW_PRIORITY permite retrazar la ejecución del comando DELETE hasta que ninguno de los otros usuarios este leyendo la tabla.

— **Sentencia UPDATE**

El comando UPDATE actualiza los datos existentes en la tabla.

Sintaxis MySQL

UPDATE [LOW_PRIORITY] table_name
SET {column_name|variable_name}={DEFAULT|expression}
WHERE conditions
[LIMIT integer]

La cláusula LOW_PRIORITY como en las sentencias de INSERT y DELETE hace que la sentencia no se ejecute hasta que ningún usuario este consultando la tabla.

La cláusula LIMIT restringe la acción de actualizar a un número específico de renglones a través de la designación de un valor entero.

3.2.4 Sentencias de Control para Recuperación y Concurrencia

— Sentencias COMMIT y ROLLBACK

Cada vez que se realiza alguna operación en la base de datos se realiza no sobre la tabla en sí, sino sobre una copia local de la misma. Así, si se desea que los resultados de la modificación se trasladen a la base de datos y perduren en el tiempo hay que confirmar dicha operación con el comando COMMIT. También se puede impedir que los últimos cambios lleguen a efectuarse con el comando ROLLBACK, aunque existen algunas sentencias SQL que se "autoconfirman" y no se puede volver atrás.

La sentencia COMMIT explícitamente termina una transacción abierta y la sentencia ROLLBACK deshace la transacción desde el principio de esta ó a un punto previo.

Sintaxis MySQL

Por default, MySQL trabaja en modo autocommit. Esto significa que cuando se ejecuta una actualización, MySQL guardará la actualización en la base de datos.

Si se utilizan transacciones en tablas seguras como BDB, InnoDB se puede iniciar la opción no-autocommit con la siguiente orden:

SET AUTOCOMMIT=0

Después de especificar esta opción, se debe utilizar COMMIT para almacenar los cambios en las bases de datos y si se desea ignorar los cambios que se han hecho desde el principio de la transacción se debe utilizar ROLLBACK.

Si se desea cambiar el modo de AUTOCOMMIT para una serie de declaraciones, se puede usar el BEGIN o la sentencia BEGIN WORK como se muestra a continuación.

```
BEGIN;
SELECT @A:=SUM(salario) FROM table1 WHERE type=1;
UPDATE table2 SET summmary=@A WHERE type=1;
COMMIT;
```

Si se utilizan transacciones en tablas no seguras, los cambios se guardarán inmediatamente, independientemente del status del modo AUTOCOMMIT.

Sin embargo, si se utiliza el ROLLBACK cuando se han actualizado transacciones no seguras en las tablas se mostrará un mensaje de advertencia (ER_WARNING_NOT_COMPLETE_ROLLBACK). Todas las transacciones en tablas seguras se restaurarán pero aquellas transacciones en tablas no seguras no cambiarán.

Utilizando BEGIN o SET AUTOCOMMIT=0, MySQL utiliza un registro binario para los respaldos en lugar del registro más antiguo que antes se haya actualizado. Las transacciones son almacenadas en un pedazo de ese registro binario.

Los siguientes comandos terminan la transacción automáticamente:

ALTER TABLE BEGIN CREATE INDEX
DROP DATABASE DROP TABLE RENAME TABLE
TRUNCATE

3.2.5 Sentencias de Seguridad

El DCL (Lenguaje de Control de Datos), contiene elementos útiles para trabajar en un entorno multiusuario en el que es importante la protección de los datos, la seguridad de las tablas y el establecimiento de restricciones en el acceso así como elementos para coordinar los datos compartidos por parte de usuarios concurrentes, asegurando que no interfieren unos con otros.

— **Sentencia GRANT**

En SQL99, la sentencia GRANT autoriza a usuarios y roles para acceder y usar los objetos de la base de datos.

Sintaxis MySQL

```
GRANT {ALL PRIVILEGES
| SELECT
| INSERT [(column_name[,...n])]
| DELETE
| UPDATE [(column_name[,...n])]
| REFERENCES [(column_name[,...n])]
| USAGE
| ALTER
| CREATE
| DROP
| FILE
| INDEX
| PROCESS
```

| RELOAD
| SHUTDOWN} [,...n]
*ON {table_name | * | *.*| database_name.*}*
TO grantee_name [IDENTIFIED BY 'password'] [,...n]
[WITH GRANT OPTION]

MySQL provee privilegios de acceso adicionales, primeramente relacionado a la manipulación de objetos dentro de una base de datos. Y con los otros privilegios, concede cualquier privilegio de acceso (como ALTER, CREATE, INDEX o RELOAD) y de ejecución del comando al usuario. REFERENCES por el momento no tiene funcionalidad. El cláusula USAGE actualmente, deshabilita los privilegios del usuario de la sesión.

Los siguientes son privilegios de acceso que son usados conjuntamente con las tablas: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT, INDEX y los faltantes pueden ser aplicados a nivel columna: ALTER, INSERT, UPDATE y SELECT.

La implementación de la cláusula ON permite opciones interesantes. Pueden aplicarse privilegios globales a todas las bases de datos en el servidor con la opción ON *.*.

Los privilegios extensos pueden ser asignados en las bases de datos especificando ON database_name.* u ON * con la base de datos actual. El host, tabla, base de datos y el nombre de la columna deben ser de 60 o menos caracteres.

MYSQL tiene la posibilidad de otorgar permisos a un usuario específico de un host determinado si en la opción grantee_name se coloca el USER@HOST. Pueden ser agregados caracteres de comodín en la opción grantee_name para proporcionar los permisos de acceso a un extenso número de usuarios en una sola vez. La opción grantee_name debe ser de 16 o menos caracteres. Al especificar al usuario, se puede especificar la contraseña del usuario ejecutando la cláusula IDENTIFIED BY.

— **Sentencia REVOKE**

La sentencia REVOKE quita los permisos de un usuario, grupo o rol sobre un objeto específico de la base de datos o hacia un comando del sistema.

Sintaxis MySQL

ROVOKE {ALL PRIVILEGES
| SELECT
| INSERT [(column_name[,...n])]
| UPDATE [(column_name[,...n])]
| REFERENCES [(column_name[,...n])]
| DELETE
| USAGE
| ALTER
| CREATE
| DROP
| FILE
| INDEX
| PROCESS
| RELOAD
| SHUTDOWN}[,...n]
*ON {table_name | * | *.* | database_name.*}*
FROM user_name [,...n]

La sentencia REVOKE quita cualquier permiso previamente otorgado a uno o varios usuarios. Los permisos pueden ser revocados globalmente, como se describió en la sentencia GRANT. En general REVOKE es la contraparte del comando GRANT, ya que se pueden quitar permisos sobre ejecución de comandos o comandos a nivel tabla.

3.3 EJEMPLOS:

CREATE DATABASE Escuela;

DROP TABLE IF EXISTS Alumno;
CREATE TABLE Alumno (nombre VARCHAR(20) NOT NULL,sexo ENUM('F','M') NOT NULL, Alumno_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY);

INSERT INTO Alumnos VALUES ('Marisol','F',1);

GRANT ALL ON Alumnos. TO juan IDENTIFIED BY "jc2000";*

```
DROP TABLE IF EXISTS miembro;
CREATE TABLE miembro
(       Apellido_paterno VARCHAR(20) NOT NULL,
        Apellido_materno VARCHAR(20) NOT NULL,
        Nombre VARCHAR(20) NOT NULL,
        Email VARCHAR(100) NULL,
        Calle VARCHAR(50) NULL,
        Numero INT NULL,
        Colonia VARCHAR (50) NULL,
        Municipio VARCHAR(50) NULL,
        Estado VARCHAR(50) NULL,
        Pais VARCHAR(50) NULL,
        Codigo_postal VARCHAR(10) NULL,
        Telefono VARCHAR(20) NULL, intereses VARCHAR(255) NULL);


ALTER TABLE miembro
        ADD miembro_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY;


SELECT * FROM Alumno;


SELECT nombre FROM Alumno;


SELECT sexo, COUNT(*) FROM Alumno GROUP BY sexo;


DELETE FROM miembros WHERE Estado="Mexico";


UPDATE miembro
SET Email='jclp77@aol.com',Calle='Tierra Nueva 99',Colonia='Tierra Nueva',
Municipio='Azcapotzalco',Codigo_postal='02130'
WHERE Apellido_paterno='López' AND Apellido_materno='Peñafort' AND Nombre='Juan Carlos';
```

PARTE IV. ACCESO A UNA BASE DE DATOS RELACIONAL EN AMBIENTE WEB CON EL LENGUAJE DE PROGRAMACIÓN PERL

4.1. MODULO DBI

DBI (Interfaz de Base de Datos-*Data Base Interfaz*) es un módulo de Perl para acceso a bases de datos. DBI define un conjunto de funciones, variables y convenciones que ofrecen una interfaz de base de datos consistente e independiente de la base de datos que se este utilizando.

DBD (Manejador de Base de Datos-*Data Base Driver*) es el manejador de la base de datos. Existe un módulo DBD para cada tipo de base de datos, dicho módulo se encarga de pasar las peticiones que se realizan del DBI a peticiones a la base de datos sobre la que se esta ocupando. Para trabajar con una base de datos determinada sólo hace falta tener instalado el módulo DBD correspondiente, por ejemplo, para trabajar con la base de datos MySql se debe instalar el módulo DBD-MySql.

4.1.1. Arquitectura

La arquitectura DBI se divide en dos partes: El propio interfaz DBI y los drivers (manejadores). El DBI define la interfaz de programación real, dirige las llamadas de los métodos a los drivers apropiados y les proporciona varios servicios de ayuda. Los drivers específicos son implementados para cada tipo de base de datos y realizan operaciones dentro de esta.

Por lo tanto, si se diseña software usando la interfaz de programación del modulo DBI, el método que se utiliza se define dentro del propio modulo. De allí, el módulo de DBI decide qué driver debe manejar para la ejecución del método y pasar el método al driver apropiado para su ejecución.

Bajo esta arquitectura, es relativamente directo colocar un driver en ejecución para cualquier tipo de base de datos. Todo lo que se requiere es colocar los métodos de ejecución definidos en la especificación de DBI, de una manera que sea significativa para esa base de datos. Los datos devueltos por este módulo se pasan nuevamente dentro del módulo de DBI, y de él se vuelve al programa de Perl. Toda la información que pasa entre el modulo DBI y sus drivers son los tipos de datos estándares de Perl, de tal modo

que se preserva el aislamiento del módulo de DBI de cualquier conocimiento de la base de datos

La separación de los drivers de modulo DBI, permite una interfaz de programación de gran alcance y ayuda a que se pueda extender a casi cualquier base de datos disponible. Los drivers que existen actualmente para las bases de datos son: Oracle, Informix, mSQL, MySQL, Ingres, Sybase, DB2, Empress, SearchServer, y PostgreSQL; y hay drivers estándares para los archivos de XBase y de CSV.

Los manejadores también son llamados drivers de la base de datos, o DBDs, después del namespace en el cual se declaran. Por ejemplo, el Oracle utiliza DBD::Oracle, Informix utiliza DBD::Informix, etc. Un consejo útil para recordar en la arquitectura de DBI es que DBI puede soportar las bases de datos independientes y DBD soporta bases de datos dependientes. (Figura 4.1)

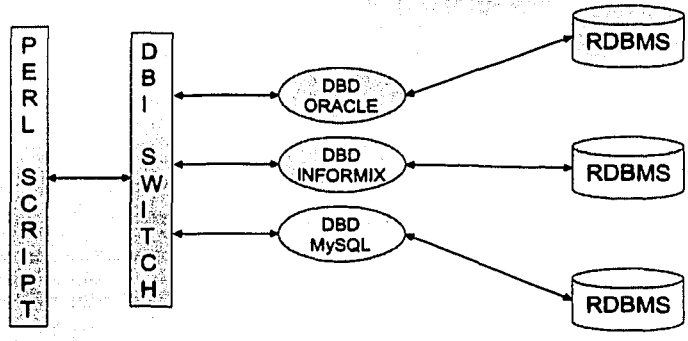


Fig 4.1 Drivers o DBD's

DBI utiliza las características de objetos orientados de Perl, por eso es extremadamente simple iniciar el uso del DBI agregando la siguiente línea dentro de los scripts:

use DBI;

al inicio de los programas. Esta línea sitúa y carga el módulo de la base DBI. Los módulos individuales del driver de la base de datos se cargan según se requiera ya que generalmente no son cargados explícitamente.

4.1.2. Manejadores (HANDLES)

El modulo DBI define tres tipos de objetos que usa para interactuar con las bases de datos. Esos objetos se conocen como handles (manejadores). Existen manejadores para drivers, manejadores para las conexiones a la base de datos y manejadores para los comandos individuales en la base de datos, conocidos como sentencias.

— *Manejadores de Drivers*

Estos representan los drivers que están cargados en el modulo DBI y se inicializan cuando el driver es cargado por el modulo DBI. Hay exactamente un manejador de driver por cada driver cargado. Inicialmente, el manejador de driver es el único contacto que el modulo DBI tiene con el driver, y en esta etapa, no se ha hecho ningún contacto con la base de datos a través del driver.

Los únicos dos métodos significativos disponibles a través del manejador de driver son data_sources (), que especifica con qué se puede conectar, y connect (), para crear una conexión. Estos métodos se utilizan comúnmente como los métodos de la clase de DBI.

Puesto que un manejador de driver encapsula totalmente al driver, pueden ser cargados múltiples drivers simultáneamente. Esto permite que el modulo DBI sea un interfaz de gran alcance.

Por ejemplo, si un programador trabaja con el intercambio de datos de una base de datos Oracle a una base de datos de Informix, es posible escribir un solo programa de DBI que se conecta simultáneamente con ambas bases de datos y pasar simplemente los datos de una a otra. En este caso, se crean dos manejadores de driver, uno para Oracle y otro para Informix. En esta situación no hay ningún problema, puesto que cada manejador de driver es un objeto totalmente separado de Perl.

Dentro de las especificaciones del modulo DBI, un manejador de driver se refiere generalmente como nombre de variable $drh.

— *Manejadores de Bases de Datos*

Los manejadores de la base de datos son el primer paso para trabajar con la base de datos, encapsulan una sola conexión a una base de datos en particular. Antes de

ejecutar sentencias SQL dentro de una base de datos, hay que conectarse con esta. Esta se alcanza generalmente utilizando el método connect () del modulo DBI:

```
$dbh = DBI->connect ( $data_source, ... );
```

La mayoría de bases de datos tiende hoy en día a funcionar en un modo multiusuario, permitiendo muchas conexiones simultáneas y los manejadores de la base de datos son diseñados por consiguiente.

Los manejadores de la base de datos son procesos hijos de su correspondiente manejador de driver, que apoya la noción que también poder hacer múltiples conexiones simultáneas a los múltiples tipos de la base de datos, así como múltiples conexiones simultáneas a las bases de datos del mismo tipo. Por ejemplo, un script mas complicado podría ser dos conexiones, una para Oracle y otra para Informix para monitorear un proceso de cambio en la estructura de los datos de diferentes usuarios.

Dentro de la especificación del DBI y del código de ejemplo, las manejadores de la base de datos se les suele denominar como nombre de variable $dbh.

— *Manejadores de Sentencias*

Los manejadores de sentencias son el tipo final de objeto que DBI define para la interacción y la manipulación de la base de datos. Estos manejadores encapsulan las sentencias SQL individualmente que serán ejecutadas dentro de la base de datos.

Los manejadores de sentencias son procesos hijos correspondientes al manejador de base de datos. Puesto que los manejadores de sentencias son objetos en su propia derecha, los datos dentro de una sentencia son protegidos de alteraciones o modificaciones por otras manejadores de sentencias.

Para una determinada base de datos, no hay límite práctico al número de manejadores de sentencias que pueden ser creadas y ejecutadas. Múltiples sentencias pueden ser creadas y ejecutadas en un solo script, y los datos se pueden ser procesados de acuerdo a como regresen. El número de manejadores de sentencias que se pueden ejecutar a la vez (concurrentemente) depende del driver de la base de datos (modulo DBD).

Dentro de las especificaciones del DBI y del código de ejemplo, los manejadores de sentencias se refieren generalmente como nombre de variable $sth.

— *Origen de Datos*

Al conectase a una base de datos por el modulo DBI, se debe especificar al modulo donde buscar la base de datos para conectarse. Por ejemplo, el driver de la base de datos puede necesitar un nombre de la base de datos o el nombre físico de la computadora donde reside la base de datos. Esta información se conoce como origen de datos o data source name; y de todos los aspectos de DBI, este es posiblemente la mayor dificultad para estandarizar la conexión debido al número y a la diversidad de sintaxis que se utilizan.

El modulo DBI requiere el origen de datos, empezando con los caracteres dbi: y el nombre del driver, seguido por dos puntos (por ejemplo dbi:MySql:). Cualquier texto que siga se pasa al metodo connect () del driver para su interpretación. La mayoría de los drivers tienen un nombre simple para la base de datos o más a menudo, un sistema de nombres de pares-valor, separados por puntos y comas

Por ejemplo, MySQL necesita un nombre de host, nombre de la base de datos, y potencialmente el número de puerto TCP/IP para conectarse al servidor de la base de datos. Sin embargo Oracle puede necesitar sólo una simple palabra que es un alias de un identificador más complicado de conexión que esta almacenado en varios archivos de configuración de Oracle.

Conectar y Desconectar con la Base de Datos

La actividad principal en la programación con las bases de datos generalmente implica la ejecución de sentencias SQL dentro de la base de datos. Sin embargo, para lograr esta tarea, primero se debe establecer una conexión a la base de datos. Además, después de que se finalizó el trabajo con la base de datos, es un buen consejo desconectarse de esta, para liberar los recursos locales del CPU y recursos de la base de datos.

Conexión

Un RDBMS relativamente sencillo es MySQL, porque tiene un simple método de conexión: to connect, el programa se conecta al puerto TCP/IP de la computadora donde se tiene la base de datos. Esto establece una conexión viva con la base de datos. Sin

embargo, sistemas más complejos como Oracle, tienen una seguridad interna, por lo que se necesita un nombre de usuario y una contraseña para conectarse con la base de datos.

Algunos de los parámetros más utilizados en la sintaxis de conexión a las diferentes bases de datos son los siguientes.

- El data source name, una cadena que contiene información específica del driver a utilizar, con que base de datos se va a conectar y posiblemente su ubicación.
- El nombre de usuario que se utiliza para conectarse con la base de datos.
- Algunos sistemas de bases de datos no manejan el concepto de usuario basado en la autenticación, pero aun así se debe especificar los argumentos de usuario y contraseña con cadenas vacías (' ').
- La contraseña asociada con el nombre de usuario.

La sintaxis para conectarse con la base de datos a través del modulo DBI usando el método: connect () se puede definir como:

```
$dbh = DBI->connect( $data_source, $username, $password, \%attr );
```

El argumento final \%attr, es opcional y puede ser omitido. \%attr esta referenciado al hash que contiene los atributos del manejador que serán aplicados en la conexión.

Este método cuando es invocado regresa un manejador de base de datos si la conexión fue exitosa con la base de datos y cuando falla regresa un valor undef.

A continuación se muestra un ejemplo para ilustrar el método de conexión: DBI→connect( )

```
#!c:\perl\bin\perl.exe  #Ruta donde esta instalado Perl

use DBI;        # Carga el modulo DBI

### Haciendo la conexión con el driver Oracle
my $dbh = DBI->connect( "dbi:Oracle:prometeo", "username", "password" )
   or die "No se puede conectar con la base de datos Oracle: $DBI::errstr\n";
exit;
```

En caso de existir un error en la conexión con la base de datos y asegurase de que muestre el tipo de error, se utiliza $DBI::errstr.

Desconexión

DBI provee un método para desconectar el manejador de base datos utilizado. Es una buena práctica, especialmente en los programas que se tienen múltiples conexiones o que se tendrán múltiples conexiones.

El método para llevar a cabo la desconexión del manejador de base de datos es el siguiente:

```
$rc = $dbh->disconnect();
```

El siguiente ejemplo muestra el método de desconexión:

```
#!c:\perl\bin\perl.exe
#
#                              Conexión a una base de datos Oracle
#                        con reporte de auto-error deshabilitado
#                        entonces realiza una desconexión explicita.

use DBI;        # Carga el modulo DBI

### Realiza la conexión a la base de datos utilizando el driver de Oracle.
my $dbh = DBI->connect( "dbi:Oracle:prometeo", "username", "password" , {
                        PrintError => 0 } )
  or die "No se puede conectar a la base de datos Oracle: $DBI::errstr\n";

### Ahora, la desconexión a la base de datos
$dbh->disconnect
  or warn "Error en la Desconexión: $DBI::errstr\n";

exit;
```

Es importante llevar a cabo la desconexión de la base de datos, ya que podrían surgir problemas con transacción rollback o commit, porque el manejador no terminará de realizar operaciones de inserción, actualización ó eliminación de datos que se hayan hecho durante el programa.

4.1.3. Ejecución de Sentencias SQL

Para manipular los datos de la Base de Datos desde el script Perl, se deben ejecutar sentencias SQL desde el mismo script. El proceso de obtener datos usando el modulo DBI se divide en cuatro etapas:

— **Preparación**: En esta etapa se analiza y valida la sentencia SQL y se devuelve un manejador de sentencia representando la sentencia SQL en la base de datos. Se utiliza el método prepare (con la consulta SQL como parámetro) del manejador de la base de datos, devolviendo el manejador de la sentencia.

— **Ejecución**: En esta etapa se ejecuta el manejador de sentencia obtenido en la etapa anterior. Se hace la consulta y se almacenan los datos en las estructuras de datos correspondientes de la Base de Datos, aunque en esta etapa, el script en Perl no puede acceder a los datos obtenidos. Se utiliza el método **execute** del manejador de sentencia.

— **Extracción**: En esta etapa se extraen los datos de la base de datos usando el manejador de sentencia. En dicha captación de datos se va almacenando los datos consultados, tupla a tupla en las estructuras de datos de Perl, para que sean posteriormente manipulados por el script.

— **Liberación**: Aquí se liberan los recursos ocupados por el manejador de sentencia y por la base de datos. Se utiliza el método **finish** del manejador de sentencia.

```
#!c:\perl\bin\perl.exe

use DBI;

#Conexión a la base de datos
my $dbh = DBI->connect('dbi:ODBC:eventos', '', '') or die "No se puede conectar con la base de datos:
$DBI::errstr\n";

#Etapa de preparación de la sentencia
my $sth = $dbh->prepare("SELECT titulo, descripcion, contacto, fecha FROM evento ORDER BY fecha
DESC");

#Etapa de ejecución de la sentencia
$sth->execute;

#Empieza la extracción de datos. Imprimiendo tupla por tupla
while (@row = $sth->fetchrow_array) {
        print "Nombre del evento: $row[0]\n";
        print "Descripcion del evento:$row[1]\n";
        print "Contacto: $row[2] \n";
        print "Fecha de publicacion:$row[3]\n";
}

#Ahora la liberación de recursos ocupados por la sentencia
 $sth->finish();
#Desconexión de la base de datos
$dbh->disconnect || warn "\nFallo al desconectar.\nError: $DBI::errstr\n";
```

Para la extracción de datos provenientes de la consulta, el modulo DBI proporciona una serie de métodos para el manejador de sentencia:

1. *fetchrow_array*: Regresa un arreglo con los atributos de la consulta de cada tupla. Cada vez que se solicite devolverá una tupla y avanzará el cursor a la siguiente tupla. Si devuelve un arreglo vacío, es que no quedan más tuplas.

2. *fetchrow_arrayref*: Es análogo al método anterior, pero devuelve una referencia a un arreglo en lugar del arreglo en si.

3. *fetchrow_hashref*: Este método devuelve una referencia a un arreglo asociativo (hash), donde las claves son los nombres de los campos consultados con la sentencia SELECT.

Las operaciones que se realizan en la base de datos que al no ser consultas (sentencias SELECT) no necesitan realizar las cuatro etapas de: Preparación, Ejecución, Extracción y Liberación. Porque no existe la etapa de extracción de datos.

DBI define el método do para el manejador de la base de datos. Este método devuelve false sólo si hay un error. Ejemplo:

```
#!c:\Perl\bin\Perl.exe

use DBI;

$Evento="I ENCUENTRO DE LICENCIADOS EN MATEMÁTICAS APLICADAS Y COMPUTACION";

$Descripcion="ES EL PRIMER ENCUENTRO DE LOS LICENCIADOS EN M.A.C. QUE SE LLEVARA A
CABO DURANTE LOS DIAS 14,15 Y 16 DE FEBRERO DEL 2003. EN ESTOS DIAS HABRA
CONFERENCIAS, MESAS DE TRABAJO, EXHIBICIÓN DE NUEVAS TECNOLOGÍAS, ETC.";

$Pagina="www.egresadosdemac.org";

$Contacto="jclp77@hotmail.com";

my $dbh = DBI->connect('dbi:ODBC:eventos', '', '') or die "$DBI::errstr\n";

($sec, $min, $hour, $mday, $mon, $year, $wday, $yday, $isdst) = gmtime;
$Fecha=($mday . "/" . ($mon+1) . "/" . ($year+1900));

#Insertar datos en la base de datos
$cursor = $dbh->do("INSERT INTO evento VALUES
('$Fecha','$Evento','$Descripcion','$Pagina','$Contacto')") || warn "Error al insertar el evento:
$DBI::errstr\n";
#Desconexión de la base de datos
$dbh->disconnect || warn "\nError al desconectar.\nError: $DBI::errstr\n";
```

El método do, se utiliza principalmente para la ejecución de sentencias de inserción de datos, actualización o eliminación de datos. También para crear algunos objetos dentro de la base de datos como las tablas.

Como se observa en el script Perl, la ejecución de sentencias SQL estándar no tienen mayor problema, sin embargo se necesita saber la sintaxis correcta. Es importante mencionar que cada RDBMS tiene su propio agregado del SQL, como por ejemplo para manejar valores nulos en Oracle se utiliza la función NVL, en Access IsNull, etc; por lo que es necesario conocer los agregados.

En MySQL existen diversas funciones y agregados que se pueden utilizar junto con las consultas, para mayor información consultar la documentación del manejador.

4.2. APLICACIÓN

4.2.1 Problemática

A partir de 1995, la fundación W.K. Kellogg (FWKK) implementó una Iniciativa de Nutrición Humana en América Latina y el Caribe (LAC) como un medio para contribuir a romper el ciclo de la pobreza en la región. Como parte de esta iniciativa, hasta la fecha, un total de 55 proyectos en 12 países del área están vigentes y en operación. Estas experiencias han llegado a construir un referente importante en el desarrollo rural en sus ámbitos de operación micro-regional en la región LAC.

En el período 2000-2002 el Área de Planeación y Evaluación de Recursos y Programas de Desarrollo Rural (APERPRODER) del Colegio de Postgraduados (CP) llevó a cabo la evaluación socioeconómica de 21 proyectos que conforman la Iniciativa de Nutrición Humana de la FWKK en 10 países de LAC. En este esfuerzo de investigación, se ha obtenido una gran cantidad de información de cada uno de estos proyectos, tales como: informes de evaluación de cada uno de los proyectos y a nivel global, anexos estadísticos de cuadros de resultado, archivos fotográficos y de video, así como bases de datos con información socioeconómica de las familias beneficiarias y de los cambios ocurridos con su participación en los proyectos.

De lo anterior, surge la necesidad de dar una mayor proyección e impacto a las experiencias micro-regionales de la Iniciativa de Nutrición Humana en diferentes ámbitos, así como dar un mayor aprovechamiento de la información disponible, la cual

constituye un elemento importante de referencia en la toma de decisiones para los diversos actores involucrados en el desarrollo rural regional y comunitario de LAC, sean las propias organizaciones que operan los proyectos u otras involucradas en este aspecto.

En Agosto del año 2002 durante el Tercer Encuentro de Proyectos de la Iniciativa de Nutrición Humana en Santo Domingo, República Dominicana hubo consenso de establecer la Red Latinoamericana y El Caribe sobre Nutrición Humana y Desarrollo Rural Sustentable (RED LAC) y se están estableciendo las bases para su operación. En ese momento, se establecieron los enlaces regionales para su operación (México, Centro América, El Caribe, Región Andina y América del Sur).

Esta RED, en su reunión en octubre de 2002, solicitó al APERPRODER del CP diseñar y establecer una página Web como una herramienta que dinamice la comunicación y consulta de información entre sus miembros.

El APERPRODER propone un proyecto que plantea crear, poner en operación, administrar y dar mantenimiento una página Web, en donde los integrantes de la Red Latinoamericana y el Caribe sobre Nutrición Humana y Desarrollo Rural Sustentable (RED LAC) tengan un espacio de consulta, acceso a información (bases de datos de la evaluación socioeconómica), análisis y retroalimentación de experiencias y lecciones aprendidas. Así también generar vínculos con otras organizaciones y redes a nivel nacional, regional y mundial. Este espacio tiene la virtud de supera barreras de distancia, tiempo y costos.

4.2.2. Creación de la Base de Datos

El grupo del APERPRODER que llevo a cabo la evaluación socioeconómica de los 21 proyectos de la Iniciativa de Nutrición Humana, recogió una gran cantidad de información que les sirvió para analizar las condiciones en las cuales se encontraban en ese momento las familias beneficiadas de los proyectos. Esta fue recopilada a través de cuestionarios de aproximadamente 1000 a 1500 variables y fue capturada en varias hojas de cálculo (Figura 4.2) que les permitió crear un sinnúmero de cuadros de resultados (Figura 4.3), utilizando funciones propias de la hoja de cálculo tales como formulas estadísticas y tablas dinámicas.

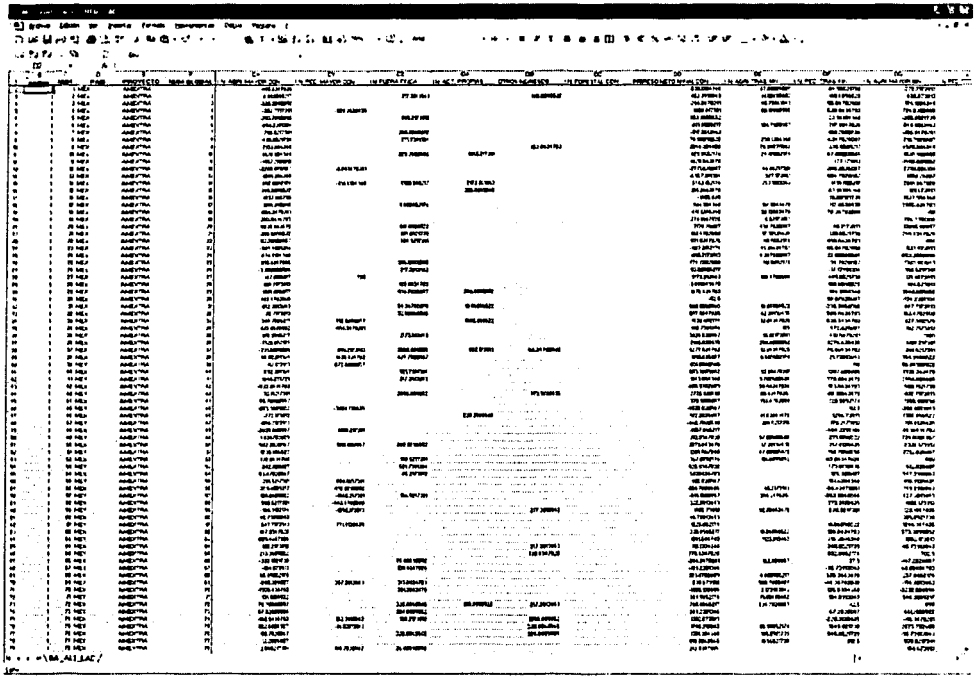


Fig 4.2 Hoja de cálculo con la información estadística

Cuadro 5. Ocupación principal de los miembros de la familia. Microrregión henequenera, Yuc. Año 2000.

Ocupación Principal	Jefes de familia		Cónyuges		Hijos		Otros familiares	
	n	%	n	%	n	%	n	%
Menores de edad o no trabajan	0	0.00	0	0.00	49	19.07	7	35.00
Agricultor	15	17.86	0	0.00	1	0.39	2	10.00
Ganadero	2	2.38	0	0.00	0	0.00	0	0.00
Comerciante	3	3.57	0	0.00	0	0.00	0	0.00
Albañil	14	16.67	0	0.00	8	3.11	0	0.00
Obrero industrial	17	20.24	0	0.00	27	10.51	3	15.00
Empleado publico o privado	7	8.33	0	0.00	13	5.05	1	5.00
Oficios (electricista, plomero, carpintero)	8	9.52	0	0.00	1	0.39	0	0.00
Estudiante	0	0.00	0	0.00	134	54.14	3	15.00
Ama de casa	4	4.76	76	100.00	19	7.39	4	20.00
Jornalero	11	13.10	0	0.00	1	0.39		
Otro *	3	3.57	0	0.00	4	1.56	0	0.00
Total	84	100.00	76	100.00	179	100.00	20	100.00

* Otro: jefes de familia (chofer y pescador). hijos (militar, pescador).

Fig 4.3 Cuadro de resultado generado con la hoja de calculo

Las hojas de cálculo están organizadas en las siguientes categorías:

- Características socio-demográficas de las familias
- Participación en las líneas de operación del proyecto
- Características de la producción de alimentos de origen vegetal y animal en la finca (traspatio y parcelas)
- Fondos comunitarios
- Características de la alimentación de las familias
- Microempresas familiares
- Vivienda y sus servicios
- Salud familiar
- Evaluación global del proyecto por las familias beneficiarias

El proyecto de la página Web de la RED LAC propuesto por el APERPRODER, consideró que las hojas de cálculo generadas con la información de las familias beneficiadas eran de gran utilidad para las instituciones involucradas y los tomadores de decisiones en esta Iniciativa porque pueden consultar la información relevante de su proyecto para implementar nuevas acciones y no hacer esfuerzos en vano. Con este fin se genero la propuesta de migrar estas hojas de cálculo para incluirlas en la página Web y mostrar con cuadros de resultado (Figura 4.3) los datos de la bases de datos.

Para llevar a cabo esta migración, se realizó como primer paso un análisis de esta información, el resultado de este análisis fue que las hojas de cálculo se podían estructurar en una o varias bases de datos relacionales, ya que los datos se estructuran lógicamente en forma de relaciones (tablas), siendo un objetivo fundamental mantener la independencia de la estructura lógica respecto al modelo de almacenamiento y a otras características del tipo físico.

Para este trabajo de tesina se ha considerado explicar solo la primera base de datos: **Características socio-demográficas de las familias**, para mostrar el panorama de complejidad que existe en el proceso de migración y además de construir un generador de

consultas en la propia página Web de la RED LAC, que permita como se mencionó anteriormente tomar decisiones a los actores involucrados en esta Iniciativa.

El análisis es la etapa más importante de todo el proceso, ya que aquí es donde se define la posible estructura que tendrá la base de datos. Para llevar a cabo esta estructura se deben homogenizar todas las variables que existen en cada una de las hojas de cálculo. Ya que cada proyecto tiene sus propias particularidades.

En esta etapa se revisan los tipos de datos que existen en la base de datos ya se numéricos o caracteres, y como se definirán en la nueva base de datos.

La estructura de la hoja de cálculo que contiene la información de características socio-demográficas de las familias (Tabla 26) es la siguiente:

TABLA 26. ESTRUCTURA DE LA HOJA DE CÁLCULO

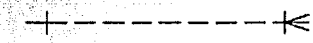
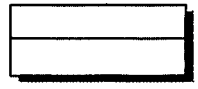
Nombre	Tipo	Descripción	Ejemplo
Sub-región	Texto	Región del continente	MEX, CEN, SUD
Código	Numérico	Código del proyecto	1..21
País	Texto	Países en la evaluación	México, Brasil, etc.
Proyecto	Texto	Abreviación del Proyecto	Amextra,Uruza
Parentesco	Numérico	Integrantes de la familia	1..8
Genero	Numérico	Genero del integrante de familia	1, 2
Edad	Numérico	Edad del integrante de la familia	0..100
Alfabetismo	Numérico	Alfabetización del integrante de familia	1,2
Escolaridad	Numérico	Grado de escolaridad de integrante de familia	0..21
Sigue estudiando	Numérico	Sigue estudiando el integrante de familia	1,2
Ocupación principal	Numérico	Ocupación principal del integrante de familia	1..12
Ocupación temporal	Numérico	Ocupación temporal del integrante de familia	1..12
Ocupación propia	Numérico	Ocupación propia de integrante de familia	1..12

Sin embargo se pretende que la estructura no se modifique demasiado ya que dificultaría la comprensión de los datos al grupo del APERPRODER que realizó la evaluación socioeconómica. Con esta premisa, las consultas que se generaran por medio de la página Web también deberán ser comprensibles y de fácil manejo para presentar cuadros de resultado como los de la Figura 4.3

Para llevar a cabo el modelado de la base de datos, se utilizó el software Allfusion™ ERWIN® Data Modeler, este software tiene la ventaja que el modelo Lógico-Físico se puede exportar directamente a la base de datos y crea automáticamente los objetos necesarios. Para una mayor referencia acerca del funcionamiento de este software consultar la siguiente dirección electrónica: *http://www.ca.com*

Para continuar con el proceso se construyó un primer modelo lógico de la base de datos familia, que se muestra en la siguiente figura (Figura 4.4), en este modelo se muestran las relaciones y las llaves primarias (Primary Keys) y secundarias (Foreign Key). En la Figura 4.5 se muestra el modelo físico que tiene la base de datos, este modelo despliega los tipos de datos que hay en la base de datos. La simbología utilizada es la siguiente (Tabla 27):

TABLA 27. SIMBOLOGÍA UTILIZADA EN LOS MODELOS DE BASES DE DATOS

Descripcion	Figura
Relación 1 a muchos	
Entidad	

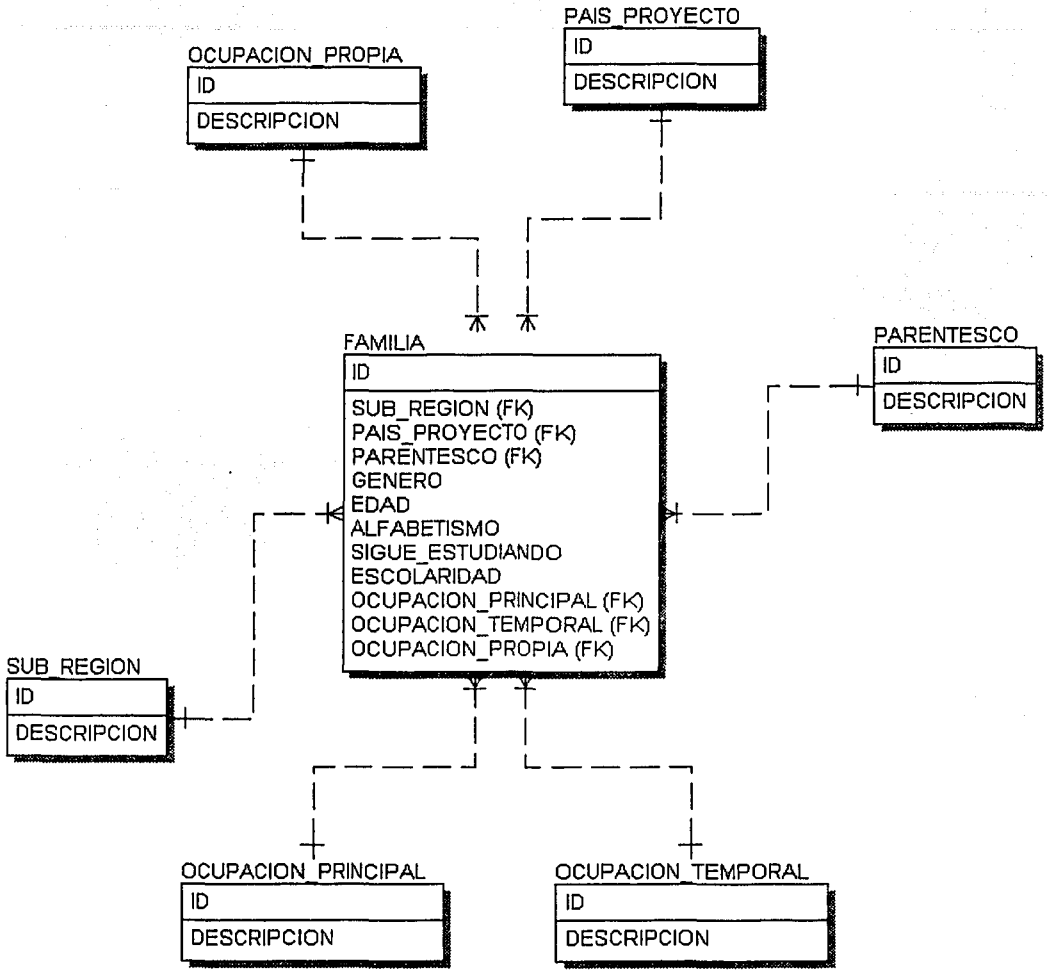


Fig 4.4 Modelo lógico 1 de la base de datos

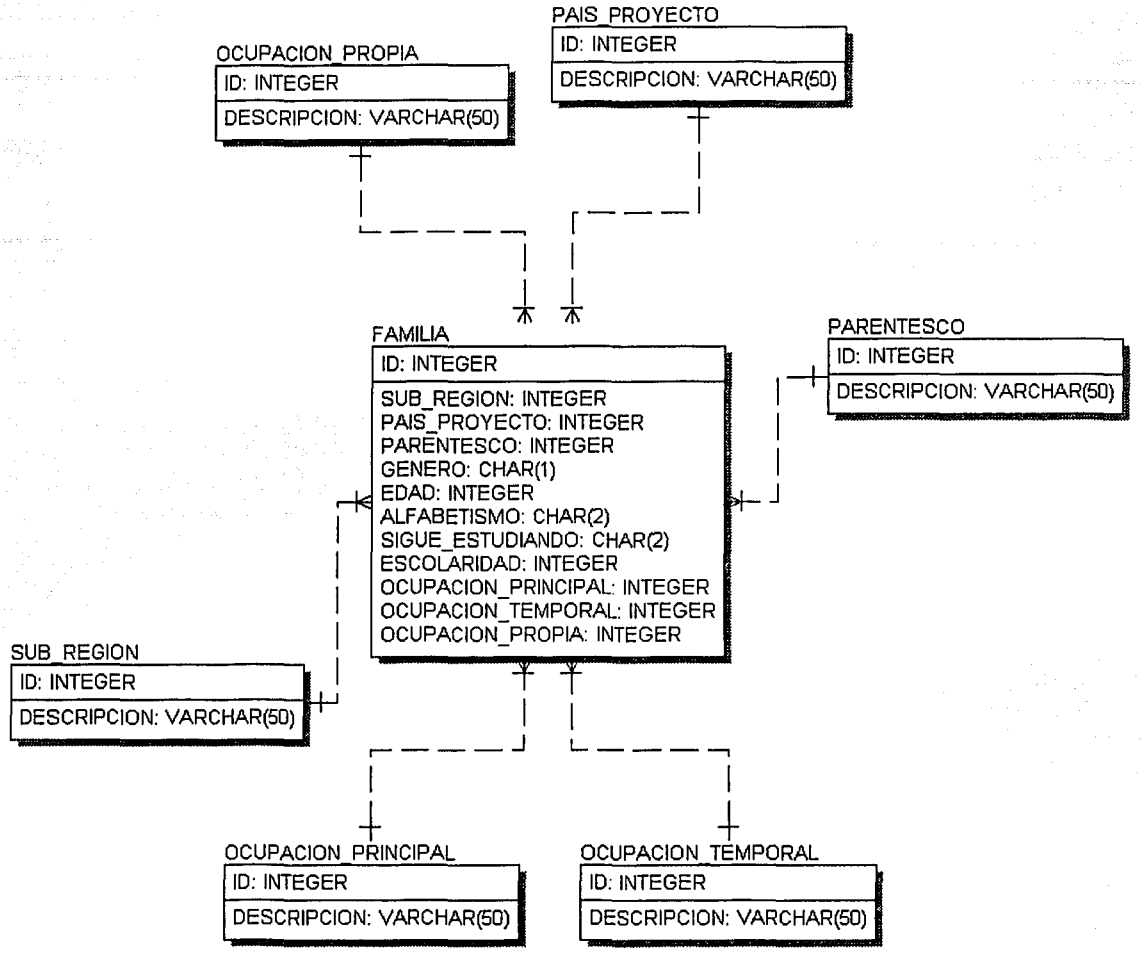


Fig 4.5 Modelo físico 1 de la base de datos

Al observar el modelo Lógico-Físico de la base de datos, se observó que un número considerable de los datos estaban repetidos constantemente, por ejemplo sub_región, pais_proyecto por lo que se normalizo el modelo de datos.

A continuación se muestra el modelo lógico de la base de datos después de normalización (Figura 4.6):

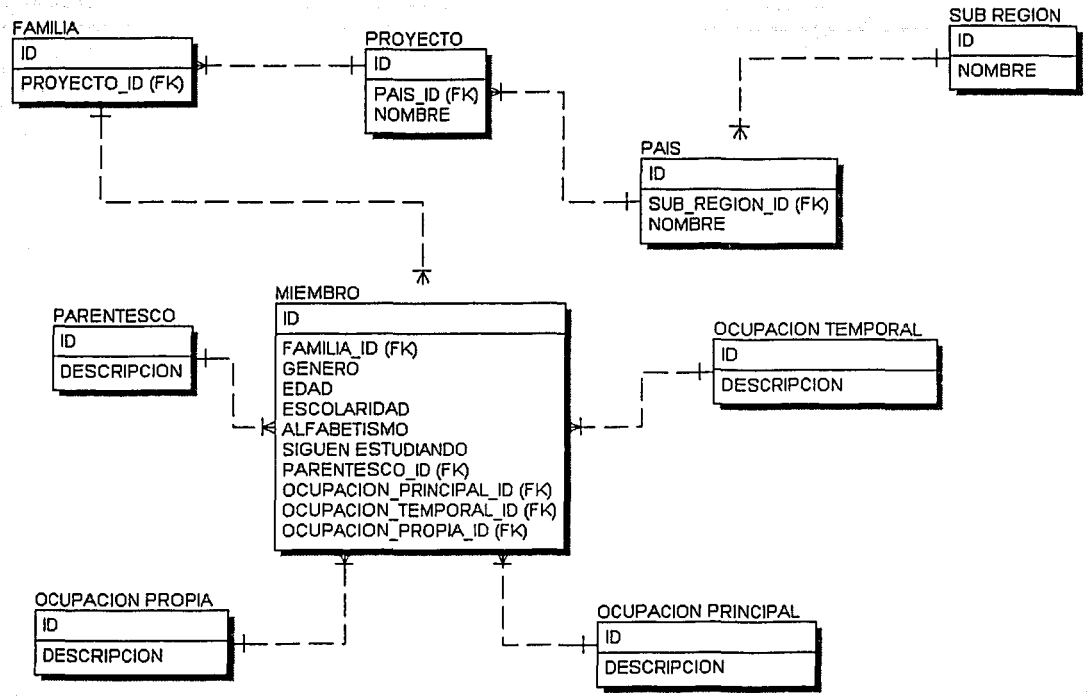


Fig 4.6 Modelo lógico 2 de la base de datos después de la normalización

Aquí se muestran las nuevas entidades que se crearon: Familia, Proyecto, País y sus relaciones entre estas. A continuación se muestra el modelo físico de la base de datos (Figura 4.7).

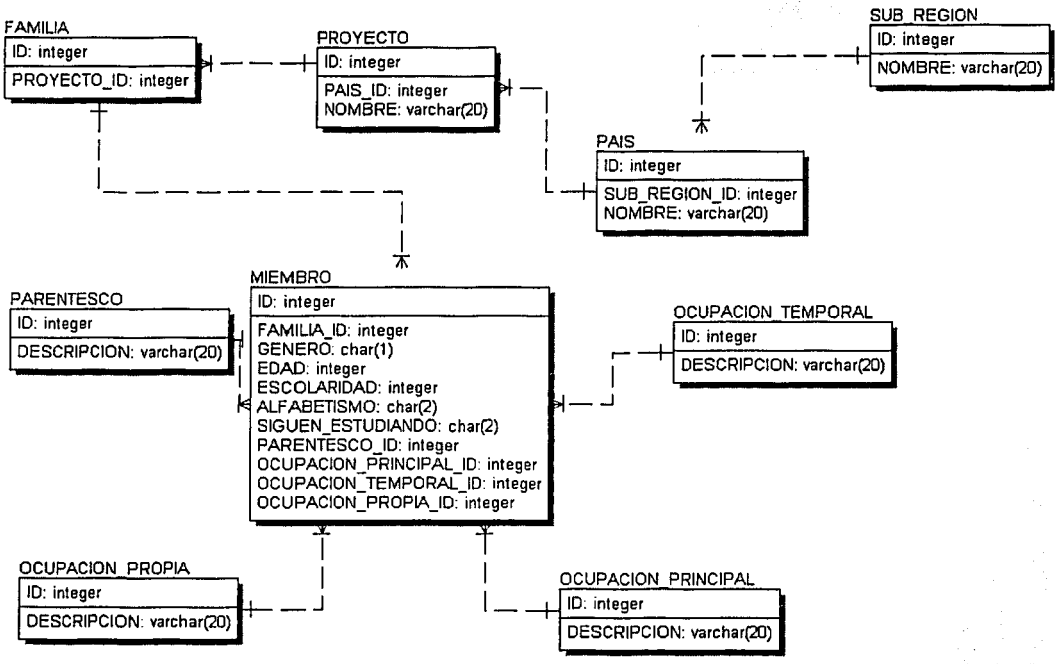


Fig 4.7 Modelo físico 2 de la base de datos después de la normalización

Como se mencionó anteriormente después de la normalización del modelo, se crearon nuevas entidades y relaciones que evitaron la existencia de datos repetidos.

Analizando el modelo 2, tiene algunas desventajas que el modelo 1, ya que existen algunas consultas que requieren de varios productos cartesianos, por ejemplo si se construye la consulta de: NUMERO DE JEFES DE FAMILIA EN LA SUB-REGION MÉXICO, se tendría que hacer el producto cartesiano entre las entidades SUB REGION X PAIS, PAIS X PROYECTO, PROYECTO X FAMILIA, FAMILIA X MIEMBRO y MIEMBRO X PARENTESCO.

Esto dificulta la consulta a la base de datos, ya que se deben hacer varios "match", generando que el manejador trabaje innecesariamente, además la consulta a las bases de datos será a través de Internet por lo que la consulta debe ser lo más sencilla posible.

Sin embargo con el modelo 1, aunque no esta completamente normalizado, las consultas se pueden realizar con mayor rapidez y son relativamente mas sencillas por ejemplo para la consulta NUMERO DE JEFES DE FAMILIA EN LA REGION MEXICO, sólo tiene que hacer los siguientes productos cartesianos: FAMILIA X SUBREGION y FAMILIA X PARENTESCO.

Por lo que entonces se tomo el Modelo 1 como la estructura de la base de datos. A continuación el script que genera el programa Allfusion™ Erwin® Data Modeler para la creación de las tablas en la base de datos MySQL:

```
CREATE TABLE OCUPACION_PRINCIPAL (
    ID              INTEGER,
    DESCRIPCION     VARCHAR(50),
    PRIMARY KEY (ID)
);

CREATE TABLE OCUPACION_PROPIA (
    ID              INTEGER,
    DESCRIPCION     VARCHAR(50),
    PRIMARY KEY (ID)
);

CREATE TABLE OCUPACION_TEMPORAL (
    ID              INTEGER,
    DESCRIPCION     VARCHAR(50),
    PRIMARY KEY (ID)
);

CREATE TABLE PAIS_PROYECTO (
    ID              INTEGER,
    DESCRIPCION     VARCHAR(50),
    PRIMARY KEY (ID)
);


CREATE TABLE PARENTESCO (
    ID              INTEGER,
    DESCRIPCION     VARCHAR(50),
    PRIMARY KEY (ID)
);

CREATE TABLE SUB_REGION (
    ID              INTEGER,
    DESCRIPCION     VARCHAR(50),
    PRIMARY KEY (ID)
);
```

```
CREATE TABLE FAMILIA (
    ID              INTEGER,
    SUB_REGION      INTEGER NOT NULL,
    PAIS_PROYECTO   INTEGER NOT NULL,
    PARENTESCO      INTEGER NOT NULL,
    EDAD            INTEGER,
    ESCOLARIDAD     INTEGER,
    GENERO          CHAR(1),
    OCUPACION_PRINCIPAL  INTEGER NOT NULL,
    ALFABETISMO     CHAR(2),
    OCUPACION_TEMPORAL   INTEGER NOT NULL DEFAULT 0,
    SIGUE_ESTUDIANDO     CHAR(2),
    OCUPACION_PROPIA     INTEGER NOT NULL DEFAULT 0,
    PRIMARY KEY (ID),
    FOREIGN KEY (SUB_REGION)
                REFERENCES SUB_REGION
                ON DELETE RESTRICT,
    FOREIGN KEY (PARENTESCO)
                REFERENCES PARENTESCO
                ON DELETE RESTRICT,
    FOREIGN KEY (PAIS_PROYECTO)
                REFERENCES PAIS_PROYECTO
                ON DELETE RESTRICT,
    FOREIGN KEY (OCUPACION_PRINCIPAL)
                REFERENCES OCUPACION_PRINCIPAL
                ON DELETE RESTRICT,
    FOREIGN KEY (OCUPACION_TEMPORAL)
                REFERENCES OCUPACION_TEMPORAL
                ON DELETE RESTRICT,
    FOREIGN KEY (OCUPACION_PROPIA)
                REFERENCES OCUPACION_PROPIA
                ON DELETE RESTRICT
);

CREATE UNIQUE INDEX PrimaryKey ON FAMILIA
(
    ID
);
```

Como se mencionó anteriormente, este software permite la conexión a la base de datos y por consiguiente la creación de todas las entidades y los objetos ya modelados. Se puede hacer a través de una conexión ODBC a MySQL, para una mayor referencia consultese las páginas *http://www.mysql.com* y *http://www.ca.com*.

En la migración de los datos de las hojas de cálculo a la nueva base de datos, existieron varios procesos entre estos la exportación de los datos a un archivo de texto delimitado por tabuladores y utilizando el lenguaje de programación Perl para llevar a cabo la construcción de varios scripts leyendo e insertando los datos a la base de datos pero no son parte del desarrollo de esta tesina.

4.2.3. Programación de Scripts CGI

Continuando con el proceso, el siguiente paso fue colocar un generador de consultas en la página Web de la Red, que permita realizar consultas generales y específicas con la información de la base de datos socioeconómica.

Para obtener la estructura y sintaxis que se genera al realizar la consulta, se hicieron varias consultas que dieran la pauta para construir la sentencia desde la página Web. Por ejemplo:

- Consulta todos los jefes de familia de la región México

```
SELECT      SUB_REGION.DESCRIPCION, PARENTESCO.DESCRIPCION,
            COUNT (FAMILIA.PARENTESCO)
FROM        PARENTESCO, SUB_REGION, FAMILIA
WHERE       PARENTESCO.ID=FAMILIA.PARENTESCO AND
            SUB_REGION.ID=FAMILIA.SUB_REGION AND
            SUB_REGION.DESCRIPCION='MEXICO' AND
            PARENTESCO.DESCRIPCION='JEFE DE FAMILIA'
GROUP BY    SUB_REGION.DESCRIPCION, PARENTESCO.DESCRIPCION;
```

- Consulta del promedio de edad por genero de las familias participantes en el proyecto Republica Dominicana – Edulelc

```
SELECT      PAIS_PROYECTO.DESCRIPCION, FAMILIA.GENERO, AVG (EDAD),
            COUNT (FAMILIA.GENERO)
FROM        PAIS_PROYECTO, FAMILIA
WHERE       PAIS_PROYECTO.ID=FAMILIA.PAIS_PROYECTO AND
            PAIS_PROYECTO.DESCRIPCION='REPUBLICA_DOMINICANA-EDUDELC'
GROUP BY    PAIS_PROYECTO.DESCRIPCION, FAMILIA.GENERO;
```

Básicamente se necesitan las tablas detalle de la base de datos: Sub_region, Pais_proyecto, Parentesco, etc., y de la tabla maestro Familia las columnas específicas de Edad y Escolaridad para empezar a construir los cuadros de resultados. Además de las funciones de contar (count) y promedio (avg).

Las tablas detalle representan a las opciones de consulta (por ejemplo: Mexico, Jefe de Familia, Masculino, etc.), es decir, consultas por sub_región, pais_proyecto, genero, parentesco y las combinaciones de ellas, como sub_region con genero, pais_proyecto con parentesco y genero, etc; y los datos almacenados en las tablas detalle corresponderán a las restricciones de la consulta por ejemplo: sub_region tendrá sólo a las sub-regiones (México, Centroamérica, El Caribe, etc.), parentesco a los integrantes de la familia (Jefe de familia, Conyuge, Hijos, etc.) y se utilizarán en la sentencia WHERE indicando el parámetro de consulta.

Sin embargo, Edad y Escolaridad, representan también opciones de consulta, pero no tienen restricciones ya que no existe una tabla detalle para cada una de ellas.

Cabe mencionar que con las opciones de Edad y Escolaridad se pueden trabajar con: promedio, entre dos valores ó superiores e inferiores. Y las opciones restantes sólo se pueden contar los elementos.

Ahora, ya analizado el proceso de consulta a la base de datos, se continua con la construcción del formulario que permitirá crear consultas dinámicas dependiendo de los parámetros de la misma. Para este proceso, se requiere el conocimiento del lenguaje HTML, en especial de formularios que ya se desarrolló en la primera parte de esta tesina.

Para construir el formulario se necesitan los siguientes objetos HTML:

- Botones de radio (Radiobox)
- Casillas de Verificación (Checkbox)
- Listas desplegables
- Botones de envío y reposición

El esquema siguiente muestra la estructura que tendrán los parámetros y las restricciones de consulta (Tabla 28).

TABLA 28. PARÁMETROS Y RESTRICCIONES DE CONSULTA

Parámetro de consulta	Restriccion de consulta
<botón de radio> **Sub Región**	<lista desplegable con la consulta a la base de datos en la tabla sub_region con el detalle de columna Descripcion> *México, Centroamérica y el Caribe, Sudamérica.*
<botón de radio> **País- Proyecto**	<lista desplegable con la consulta a la base de datos en la tabla Pais_proyecto con el detalle de columna Descripcion> *México-Amextra, México- Uruza, etc.*
<casilla de verificación> **Parentesco**	<lista desplegable con la consulta a la base de datos en la tabla Parentesco con el detalle de columna Descripcion> *Jefe de familia, Conyuge, Hijos ó Hijas, etc.*
<casilla de verificación> **Genero**	<lista desplegable con el genero> *Masculino, Femenino*
<casilla de verificación> **Alfabetismo**	<lista desplegable> *Si, No*
<casilla de verificación> **Siguen estudiando**	<lista desplegable> *Si, No*
<botón de radio> **Ocupación principal**	<lista desplegable con la consulta a la base de datos en la tabla Ocupacion_princiapal con el detalle de columna Descripcion> *Agricultor, Ganadero, Comerciante, etc.*
<botón de radio> **Ocupación temporal**	<lista desplegable con la consulta a la base de datos en la tabla ocupacion_temporal con el detalle de columna Descripcion> *Agricultor, Ganadero, Comerciante, etc.*
<botón de radio> **Ocupación propia**	<lista desplegable con la consulta a la base de datos en la tabla ocupacion_propia con el detalle de columna Descripcion> *Agricultor, Ganadero, Comerciante, etc.*
<casilla de verificación> **Edad**	<lista desplegable> *Promedio, Rango (entre 0..100), Especifica (> a n, < a n, = a n, etc.)*
<casilla de verificación> **Escolaridad**	<lista desplegable> *Promedio, Rango (entre 0..100), Especifica (> a n, < a n, = a n, etc.)*

Cabe destacar que en las listas desplegables se tendrá que agregar la restricción TODOS, ya que se necesitará para mostrar el detalle de todas las restricciones. También como complemento para el script, se construyeron algunas funciones en el lenguaje JavaScript, esto nos permitirá activar o desactivar junto con los parámetros de consulta, las restricciones que esta llevará. Por ejemplo: si se elige la opción Parentesco se deberá activar sólo el conjunto de restricciones para esa opción y si no se elige debe estar desactivada. Con la validación de las funciones de Javascript, se controla el flujo de información para construir la consulta a la base de datos.

Para accesar a la base de datos con Perl, se necesita del modulo DBI y DBD-MySQL, además del modulo CGI. Para una mayor referencia de estos módulos consultar la documentación de Perl y el anexo B de esta tesina.

Recordando, que los scripts, deben ser almacenados en el directorio cgi-bin de servidor, en este caso de Apache Web Server.

Con estos análisis previos, ahora se muestra el script que se construyó para generar el formulario de consulta a la base de datos:

La cabecera del script donde se coloca la línea en donde esta ubicado el interprete de Perl.

```
#!c:\perl\bin\perl.exe
```

Se llama al modulo DBI

```
use DBI;
```

Se imprime la cabecera de tipo HTML

```
print "Content-Type:text/html\n\n";
print "<HTML><HEAD>";
```

Función en Javascript que maximiza la pantalla

```
print <<END;
<SCRIPT language=javascript>
<!--
window.moveTo(0,0);
if (document.all) {
        top.window.resizeTo(screen.availWidth,screen.availHeight);
}
else if (document.layers||document.getElementById) {
        if (top.window.outerHeight<screen.availHeight||top.window.outerWidth<screen.availWidth){
                top.window.outerHeight = screen.availHeight;
```

```
        top.window.outerWidth = screen.availWidth;
        }
}
```

Las funciones en Javascript que habilitan y deshabilitan las opciones y restricciones de consulta

```
function disable() {
  if (document.forms[0].elements[0].checked) {
        document.forms[0].elements[1].disabled=false}
        else
        {document.forms[0].elements[1].disabled=true}

        if (document.forms[0].elements[2].checked) {
        document.forms[0].elements[3].disabled=false}
        else
        {document.forms[0].elements[3].disabled=true}

        if (document.forms[0].elements[4].checked) {
        document.forms[0].elements[5].disabled=false}
        else
        {document.forms[0].elements[5].disabled=true}

        if (document.forms[0].elements[6].checked) {
        document.forms[0].elements[7].disabled=false}
        else
        {document.forms[0].elements[7].disabled=true}

        if (document.forms[0].elements[8].checked) {
        document.forms[0].elements[9].disabled=false}
        else
        {document.forms[0].elements[9].disabled=true}

        if (document.forms[0].elements[10].checked) {
        document.forms[0].elements[11].disabled=false}
        else
        {document.forms[0].elements[11].disabled=true}

        if (document.forms[0].elements[12].checked) {
        document.forms[0].elements[13].disabled=false}
        else
        {document.forms[0].elements[13].disabled=true}

        if (document.forms[0].elements[14].checked) {
        document.forms[0].elements[15].disabled=false}
        else
        {document.forms[0].elements[15].disabled=true}

        if (document.forms[0].elements[16].checked) {
        document.forms[0].elements[17].disabled=false}
        else
        {document.forms[0].elements[17].disabled=true}
```

```
        if (document.forms[0].elements[18].checked) {
        document.forms[0].elements[19].disabled=false}
        else
        {document.forms[0].elements[19].disabled=true}

        if (document.forms[0].elements[24].checked) {
        document.forms[0].elements[25].disabled=false}
        else
        {document.forms[0].elements[25].disabled=true}
}

function disabled_1() {
 var list = document.forms[0].Edad_condicion

if (list.options[1].selected){
        document.forms[0].Edad_Rango_condicion_Min.disabled=false;
        document.forms[0].Edad_Rango_condicion_Max.disabled=false;
        document.forms[0].Edad_Rango_condicion_Min.focus()}
        else{
        document.forms[0].Edad_Rango_condicion_Min.disabled=true;
        document.forms[0].Edad_Rango_condicion_Max.disabled=true}

if (list.options[2].selected){
                document.forms[0].Edad_Especifica_condicion.disabled=false;
                document.forms[0].Edad_Especifica_condicion.focus();
                document.forms[0].Edad_Rango_condicion_especifica.disabled=false}
        else{
        document.forms[0].Edad_Especifica_condicion.disabled=true;
        document.forms[0].Edad_Rango_condicion_especifica.disabled=true}

}
function disabled_2() {
 var list_a = document.forms[0].Escolaridad_condicion
if (list_a.options[1].selected){
        document.forms[0].Escolaridad_Rango_condicion_Min.disabled=false;
        document.forms[0].Escolaridad_Rango_condicion_Max.disabled=false;
        document.forms[0].Escolaridad_Rango_condicion_Min.focus()}
        else{
        document.forms[0].Escolaridad_Rango_condicion_Min.disabled=true;
        document.forms[0].Escolaridad_Rango_condicion_Max.disabled=true}

if (list_a.options[2].selected){

document.forms[0].Escolaridad_Especifica_condicion.disabled=false;
document.forms[0].Escolaridad_Especifica_condicion.focus();
document.forms[0].Escolaridad_Rango_condicion_especifica.disabled=false}

        else{
document.forms[0].Escolaridad_Especifica_condicion.disabled=true;
document.forms[0].Escolaridad_Rango_condicion_especifica.disabled=true}

}
//-->
</SCRIPT>
END
```

El titulo de la página

```
print "<TITLE>Consulta a la Base de Datos</TITLE></HEAD>\n";
print "<BODY>";
print "<h2 align='center'>Caracteristicas Socio-demográficas de las familias participantes<h2><BR>";
```

Se abre el formulario de consulta

```
print "<FORM NAME='FORM1' METHOD='POST' ACTION='/cgi-bin/consulta_1.cgi'>";
```

Se abre la conexión a la base de datos familia de MySQL

```
my $dbh=DBI->connect('dbi:mysql:familia:192.168.23.101','root','', {RaiseError => 1});
```

Preparación y ejecución de consultas a la base de datos

```
my $sth1=$dbh->prepare ("SELECT Descripcion FROM SUB_REGION");
$sth1->execute;
my $sth2=$dbh->prepare ("SELECT Descripcion FROM PAIS_PROYECTO");
$sth2->execute;
my $sth3=$dbh->prepare ("SELECT Descripcion FROM PARENTESCO");
$sth3->execute;
my $sth4=$dbh->prepare ("SELECT Descripcion FROM OCUPACION_PRINCIPAL");
$sth4->execute;
my $sth5=$dbh->prepare ("SELECT Descripcion FROM OCUPACION_TEMPORAL");
$sth5->execute;
my $sth6=$dbh->prepare ("SELECT Descripcion FROM OCUPACION_PROPIA");
$sth6->execute;
```

La tabla que contendrá los objetos de radio, check, listas, etc., además se generan dinámicamente las opciones y restricciones con las consultas anteriores. También se hace la llamada a las funciones de habilitar y deshabilitar de Javascript para controlar el flujo de la generación de consultas.

```
print "<TABLE align='Center'>";
print "<TR><TD><INPUT TYPE='radio' NAME='Region_pais' value='Sub_region'
onClick='javascript:disable()'>Subregión</TD>";
print "<TD><SELECT NAME='Subregion_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
while (@row1=$sth1->fetchrow_array) {
print "<OPTION VALUE=$row1[$_]>$row1[$_]</OPTION>";
}
print "</SELECT></TD></TR>";
$sth1->finish();

print "<TR><TD><INPUT TYPE='radio' NAME='Region_pais' value='Pais_proyecto'
onClick='javascript:disable()'>Pais-Proyecto</TD>";
print "<TD><SELECT NAME='Subregion_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
while (@row2=$sth2->fetchrow_array) {
print "<OPTION VALUE=$row2[$_]>$row2[$_]</OPTION>";
}
```

```
print "</SELECT></TD></TR>";
$sth2->finish();

print "<TR><TD><INPUT TYPE='checkbox' NAME='Parentesco' value='Parentesco'
onClick='javascript:disable()'>Parentesco</TD>";
print "<TD><SELECT NAME='Parentesco_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
while (@row3=$sth3->fetchrow_array) {
print "<OPTION VALUE=\'$row3[$_]\'>$row3[$_]</OPTION>";
}
print "</SELECT></TD></TR>";
$sth3->finish();

print "<TR><TD><INPUT TYPE='checkbox' NAME='Genero' value='Genero'
onClick='javascript:disable()'>Genero</TD>";
print "<TD><SELECT NAME='Genero_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
print "<OPTION VALUE='H'>MASCULINO</OPTION>";
print "<OPTION VALUE='M'>FEMENINO</OPTION>";
print "</SELECT></TD></TR>";

print "<TR><TD><INPUT TYPE='checkbox' NAME='Alfabetismo' value='Alfabetismo'
onClick='javascript:disable()'>Alfabetismo</TD>";
print "<TD><SELECT NAME='Alfabetismo_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
print "<OPTION VALUE='S'>SI SABE LEER Y ESCRIBIR</OPTION>";
print "<OPTION VALUE='N'>NO SABE LEER Y ESCRIBIR</OPTION>";
print "</SELECT></TD></TR>";

print "<TR><TD><INPUT TYPE='checkbox' NAME='Siguen_estudiando' value='Siguen_estudiando'
onClick='javascript:disable()'>Siguen_estudiando</TD>";
print "<TD><SELECT NAME='Siguen_estudiando_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
print "<OPTION VALUE='S'>SI SIGUE ESTUDIANDO</OPTION>";
print "<OPTION VALUE='N'>NO SIGUE ESTUDIANDO</OPTION>";
print "</SELECT></TD></TR>";

print "<TR><TD><INPUT TYPE='radio' NAME='Ocupacion' value='Ocupacion_Principal'
onClick='javascript:disable()'>Ocupacion Principal</TD>";
print "<TD><SELECT NAME='Ocupacion_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
while (@row4=$sth4->fetchrow_array) {
print "<OPTION VALUE=$row4[$_]>$row4[$_]</OPTION>";
}
print "</SELECT></TD></TR>";
$sth4->finish();

print "<TR><TD><INPUT TYPE='radio' NAME='Ocupacion' value='Ocupacion_Temporal'
onClick='javascript:disable()'>Ocupacion Temporal</TD>";
print "<TD><SELECT NAME='Ocupacion_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
while (@row5=$sth5->fetchrow_array) {
print "<OPTION VALUE=$row5[$_]>$row5[$_]</OPTION>";
}
print "</SELECT></TD></TR>";
```

```
$sth5->finish();

print "<TR><TD><INPUT TYPE='radio' NAME='Ocupacion' value='Ocupacion_Propia'
onClick='javascript:disable()'>Ocupacion Propia</TD>";
print "<TD><SELECT NAME='Ocupacion_condicion' disabled='disabled'>";
print "<OPTION>TODOS</OPTION>";
while (@row6=$sth6->fetchrow_array) {
print "<OPTION VALUE=$row6[$_]>$row6[$_]</OPTION>";
}
print "</SELECT></TD></TR>";
$sth6->finish();

#Para Edad
print "<TR><TD><INPUT TYPE='checkbox' NAME='Edad' value='Edad'
onClick='javascript:disable()'>Edad</TD>";
print "<TD><select name='Edad_condicion' onChange='javascript:disabled_1()' disabled>";
print "<option value='Promedio'>Promedio";
print "<option value='Rango'>Rango";
print "<option value='Especifica'>Especifica";
print "</select></td></TR>";

print "<TR><TD></TD><TD align='Center'>Entre <select name='Edad_Rango_condicion_Min'
disabled='disabled'>";
for (0..100) {
        print "<OPTION VALUE=$_>$_</OPTION>";
}
print "</SELECT> y ";
print "<select name='Edad_Rango_condicion_Max' disabled='disabled'>";

for (0..100) {
        print "<OPTION VALUE=$_>$_</OPTION>";
}
print "</select></TD>";

print "<TD><select name='Edad_Especifica_condicion' onChange='javascript:disabled_1()' disabled>";
print "<option value='1'>Menor que";
print "<option value='2'>Mayor que";
print "<option value='3'>Distinto de";
print "<option value='4'>Menor ó igual que";
print "<option value='5'>Mayor ó igual que";
print "<option value='6'>Igual de";
print "</select>";

print "<select name='Edad_Rango_condicion_especifica' disabled='disabled'>";
for (0..100) {
        print "<OPTION VALUE=$_>$_</OPTION>";
}
print "</select></TD></TR>";

#Para escolaridad
print "<TR><TD><INPUT TYPE='checkbox' NAME='Escolaridad' value='Escolaridad'
onClick='javascript:disable()'>Escolaridad</TD>";
print "<TD><select name='Escolaridad_condicion' onChange='javascript:disabled_2()' disabled>";
print "<option value='Promedio'>Promedio";
print "<option value='Rango'>Rango";
```

```
print "<option value='Especifica'>Especifica";
print "</select></td></TR>";

print "<TR><TD></TD><TD align='Center'>Entre <select name='Escolaridad_Rango_condicion_Min'
disabled='disabled'>";
for (0..100) {
        print "<OPTION VALUE=$_>$_</OPTION>";
}
print "</SELECT> y ";
print "<select name='Escolaridad_Rango_condicion_Max' disabled='disabled'>";
for (0..100) {
        print "<OPTION VALUE=$_>$_</OPTION>";
}
print "</select></TD>";

print "<TD><select name='Escolaridad_Especifica_condicion' onChange='javascript:disabled_2()'
disabled>";
print "<option value='1'>Menor que";
print "<option value='2'>Mayor que";
print "<option value='3'>Distinto de";
print "<option value='4'>Menor ó igual que";
print "<option value='5'>Mayor ó igual que";
print "<option value='6'>Igual de";
print "</select>";

print "<select name='Escolaridad_Rango_condicion_especifica' disabled='disabled'>";
for (0..100) {
        print "<OPTION VALUE=$_>$_</OPTION>";
}
print "</select></TD></TR>";

print "<TR><TD><BR></TD></TR>";
print "<TR><TD><INPUT TYPE='submit' name='submit' value='Enviar Consulta'></TD>";
print "<TD><INPUT TYPE='reset' NAME='reset' VALUE='Borrar Consulta'></TD></TR>";
```

Se cierra la tabla, la forma y el encabezado del documento HTML

```
print "</TABLE></FORM></BODY></HTML>";
```

A continuación el formulario que se genera a partir del script anterior (Figura 4.8).

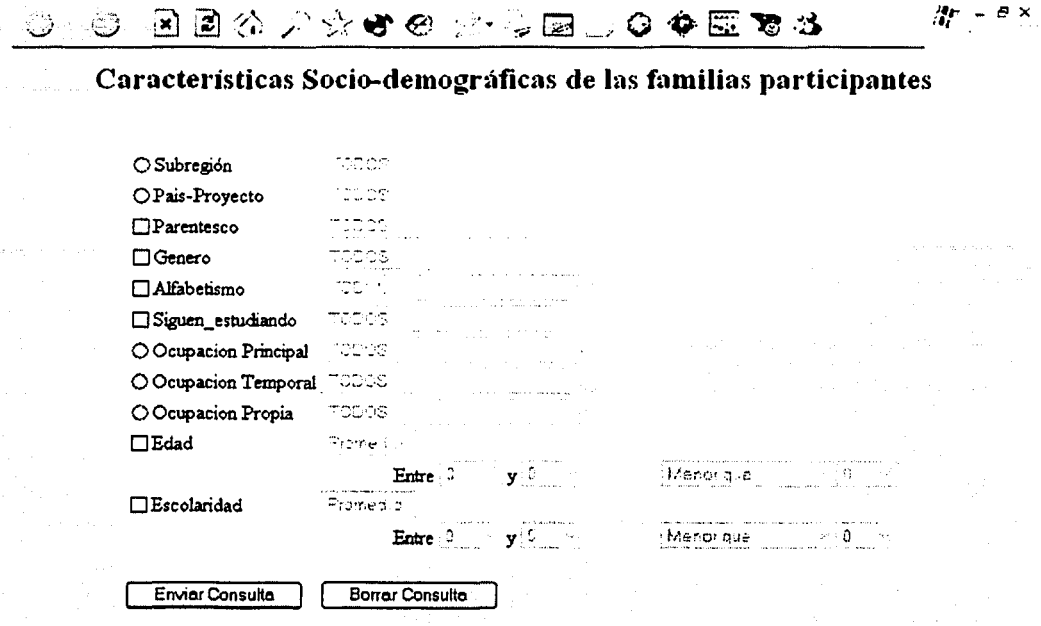


Fig 4.8 Formulario generado por el script base_1.cgi

Hasta ahora se construyó el script en el lenguaje de programación Perl con el nombre base_1.cgi, que genera dinámicamente el formulario para construir una consulta a la base de datos. Ahora se construirá el script que accesa a la base de datos y muestra la información deseada de acuerdo a los parámetros y restricciones antes seleccionados con el nombre consulta_1.cgi. Este script cuenta con algunos parámetros extras, tales como totales y porcentajes que se necesitan calcular y mostrar en un cuadro de resultado en la página Web.

La cabecera del script donde se coloca la línea en donde esta ubicado el interprete de Perl.

#!c:\perl\bin\perl.exe

Se cargan los módulos CGI para trabajar con archivos HTML y DBI para bases de datos

use CGI;
use DBI;

Se crea el objeto CGI

```
$query = new CGI;
```

Se utilizan las propiedades del objeto CGI

```
print $query->header;
print $query->start_html("Consulta");
```

Función en Javascript que maximiza la pantalla

```
print <<JAVA;
<script language='JavaScript1.2'>
<!--
window.moveTo(0,0);
if (document.all) {
        top.window.resizeTo(screen.availWidth,screen.availHeight);
}
else if (document.layers||document.getElementById) {
        if (top.window.outerHeight<screen.availHeight||top.window.outerWidth<screen.availWidth){
        top.window.outerHeight = screen.availHeight;
        top.window.outerWidth = screen.availWidth;
        }
}
//-->
</script>
JAVA
```

La captación de parámetros, en este caso las tablas

```
$region_pais=$query->param('Region_pais');
$parentesco=$query->param('Parentesco');
$genero=$query->param('Genero');
$alfabetismo=$query->param('Alfabetismo');
$siguen_estudiando=$query->param('Siguen_estudiando');
$ocupacion=$query->param('Ocupacion');
$edad=$query->param('Edad');
$escolaridad=$query->param('Escolaridad');
```

La captación de restricciones

```
$condicion_sub_region=$query->param('Subregion_condicion');
$condicion_parentesco=$query->param('Parentesco_condicion');
$condicion_genero=$query->param('Genero_condicion');
$condicion_alfabetismo=$query->param('Alfabetismo_condicion');
$condicion_siguen_estudiando=$query->param('Siguen_estudiando_condicion');
$condicion_ocupacion=$query->param('Ocupacion_condicion');
```

Captación de restricciones para Promedio y Rango

```
$Edad_condicion=$query->param('Edad_condicion');
$Edad_rango_condicion_min=$query->param('Edad_Rango_condicion_Min');
$Edad_rango_condicion_max=$query->param('Edad_Rango_condicion_Max');
$Edad_condicion_especifica=$query->param('Edad_Especifica_condicion');
```

```
$Edad_rango_condicion_especifica=$query->param('Edad_Rango_condicion_especifica');
$Escolaridad_condicion=$query->param('Escolaridad_condicion');
$Escolaridad_rango_condicion_min=$query->param('Escolaridad_Rango_condicion_Min');
$Escolaridad_rango_condicion_max=$query->param('Escolaridad_Rango_condicion_Max');
$Escolaridad_condicion_especifica=$query->param('Escolaridad_Especifica_condicion');
#Promedio,Rango y Especifica
$Escolaridad_rango_condicion_especifica=$query->param('Escolaridad_Rango_condicion_especifica');
```

Almacenamiento de las tablas y restricciones seleccionadas

```
@tablas_temp=($region_pais, $parentesco, $genero, $alfabetismo, $siguen_estudiando, $ocupacion);
@restricciones_temp=($condicion_sub_region,$condicion_parentesco,
$condicion_genero,$condicion_alfabetismo,$condicion_siguen_estudiando, $condicion_ocupacion);
```

Almacenamiento de operadores de comparación

```
@operadores=("","<",">","<>","<=",">=","=");
```

Declaración de variables de arreglo

```
my @tablas_descripcion;
my @tablas_id;
my @tablas_red_lac;
my @tablas_count;
my @tablas_match;
my @tablas;
my @tablas_restricciones;
my @restricciones;
my @count;
my @num;
```

Procedimiento que almacena las tablas seleccionadas y prepara las tablas que se necesitarán para la consulta

```
foreach (@tablas_temp) {
        if ($_ =~ /\w/g) {
                if ($_ =~ /Genero|Alfabetismo|Siguen_estudiando/) {
push(@tablas_descripcion, $tablas_descripcion="FAMILIA" . "\." . $_);
push(@tablas_count, $tablas_count="Count(FAMILIA" . "\." . $_ . ")");

                                }else {
push(@tablas_descripcion, $tablas_descripcion=$_ . "\." . "Descripcion");
push(@tablas_id,$tablas_id=$_ . "\." . "id");
push(@tablas_red_lac, $tablas_red_lac="FAMILIA" . "\." .$_);
push(@tablas_count, $tablas_count="Count(FAMILIA" . "\." . $_ . ")");
push(@tablas,$tablas=$_);
                                }
                }
}
```

A continuación se genera la cadena para la sentencia SELECT

```
foreach (reverse(@tablas_descripcion)) {
        $proyeccion=$_ . "\, " . $proyeccion;
}
$proyeccion =~ s/\, $//g;

if ($proyeccion_edad =~ /AVG\(Edad\)/g) {
        $proyeccion=$proyeccion . "," . $proyeccion_edad;
}

if ($proyeccion_escolaridad =~ /AVG\(Escolaridad\)/g) {
        $proyeccion=$proyeccion . "," . $proyeccion_escolaridad;
}

foreach (((@tablas_count)) {
        if (($_ =~ /Pais_proyecto|Sub_region/g) || ($n==1)) {
                next;
        }else{
                push(@count,$count=$_);
                $n=1;
        $proyeccion=$proyeccion . ", " . $_
}
```

Ahora la cadena para la sentencia FROM

```
foreach (@tablas) {
        $proyeccionde=$_ . ", " . $proyeccionde;
}
$proyeccionde=$proyeccionde . "FAMILIA";

#Procedimiento para colocar el signo de igual a las tablas
for(0..(@tablas_id)-1) {
        $tablas_match[$_]=$tablas_id[$_] . "=" . $tablas_red_lac[$_];
}
```

El procedimiento para convertir en una cadena los "match" necesarios y las restricciones para generar la sentencia WHERE

```
for (0..@tablas_temp) {
        if ($tablas_temp[$_] =~ /\w/g) {
                push (@tablas_restricciones,$tablas_restricciones=$restricciones_temp[$_]);
        }
}

for (0..(@tablas_restricciones)-1) {
        if ($tablas_restricciones[$_] =~ /TODOS/g) {next;
        }
        else {
                $restricciones[$_]=$tablas_descripcion[$_] . "=" . "\'$tablas_restricciones[$_]\'"
        }
}
```

```
foreach (reverse(@restricciones)) {
        if ($_ =~ /^$/) {next;
        }
        else{$detalle=$_ . " AND " . $detalle}
}

$detalle =~ s/\ AND $//g;

if ($edad =~ /\w/g) {
        if ($Edad_condicion =~ /Promedio/g) {
                $proyeccion_edad=" AVG(". $edad .")"
        }elsif ($Edad_condicion =~ /Rango/g)
        {
        $restriccion_edad=$edad . " BETWEEN " . $Edad_rango_condicion_min . " AND " .
$Edad_rango_condicion_max
        }
        elsif ($Edad_condicion =~ /Especifica/g)
        {
                $restriccion_edad=$edad . $operadores[$Edad_condicion_especifica] .
                $Edad_rango_condicion_especifica
        }
}

if ($escolaridad =~ /\w/g) {
        if ($Escolaridad_condicion =~ /Promedio/g)
        {
                $proyeccion_escolaridad=" AVG(". $escolaridad .")"
        }elsif ($Escolaridad_condicion =~ /Rango/g)
        {
                $restriccion_escolaridad=$escolaridad . " BETWEEN " .
                $Escolaridad_rango_condicion_min . " AND " .
                $Escolaridad_rango_condicion_max
        }
        elsif ($Escolaridad_condicion =~ /Especifica/g)
        {
                $restriccion_escolaridad=$escolaridad .
                $operadores[$escolaridad_condicion_especifica] .
                $escolaridad_rango_condicion_especifica
        }
}

foreach (@tablas_match) {
        $condicion=$_ . " AND " . $condicion;
}

if ($detalle =~ /\w/g) {
        $condicion=$condicion . $detalle
}else{$condicion =~ s/\ AND $//g;}

if ($restriccion_edad =~ /\w/g) {
        $condicion=$condicion . " AND " . $restriccion_edad;
}

if ($restriccion_escolaridad =~ /\w/g) {
        $condicion=$condicion . " AND " . $restriccion_escolaridad;}
```

Finalmente generar la cadena para la sentencia GROUP BY

```
foreach (reverse(@tablas_descripcion)) {
        $agruparen=$_ ."\, " . $agruparen;
}

$agruparen =~ s/\, $//g;
```

Se continúa con la construcción de la cadena final de consulta

```
$post_consulta=uc("SELECT " . $proyeccion . "\nFROM " . $proyeccionde . "\nWHERE " . $condicion .
"\nGROUP BY " . $agruparen . ";");
```

En caso de que el usuario necesite generar otra consulta

```
print "<P><BR><H4 align='center'>Para otra consulta presione <a href=\"/cgi-
bin/base_1.cgi\">aqui</A></H4><BR><BR>";
```

Almacenamiento de los títulos de las columnas seleccionadas

```
@num=(@tablas_descripcion);

if ($proyeccion_edad =~ /\w/) {
        push(@num,$num="Edad")
}

if ($proyeccion_escolaridad =~ /\w/) {
        push(@num,$num="Escolaridad")
}

push (@num,@count);
push (@num,$num="Porcentaje");
```

Se abre la tabla que contendrá los títulos de las columnas y los datos regresados por la consulta

```
print "<TABLE BORDER=1 align='center'>\n";
foreach (@num) {
        $_ =~ s/\.Descripcion|\_/ /g;
        $_ =~ s/^Count.*/ n /;
        print "<TD align='CENTER'bgcolor='#990000'><H3>
<font color='#FFFFFF'>$_</font></H3></TD>";
}
```

Continúa con la preparación y ejecución de la consulta a la base de datos familia de MySQL

```
my $dbh = DBI->connect('dbi:mysql:familia:192.168.23.101','root','', {RaiseError => 1});
my $sth = $dbh->prepare ($post_consulta);
$sth->execute();
```

Almacena los datos consultados en la variable $table

```
my $table = $sth->fetchall_arrayref;
my($i, $j);
```

Obtiene la suma de la última columna de registros

```
for $i ( 0 .. $#{$table} ) {
                $num_total=($table->[$i][-1])+$num_total;
}
```

A continuación se muestran los datos obtenidos de la consulta en la tabla y se genera el promedio (Figura 4.3)

```
for $i ( 0 .. $#{$table} ) {
        print "<TR>";
  for $j ( 0 .. $#{$table->[$i]} ) {
     print "<TD>$table->[$i][$j]</TD>";
                $promedio=$table->[$i][$j];
  }
        $promedio=($promedio*100)/$num_total;
        $promediogral=$promediogral+$promedio;
        printf ("<TD align='CENTER'>%.2f %",$promedio);
        printf "</TD>";
}
print "</TR>";
```

Muestra el total de elementos y el promedio de cada valor

```
print "<TR>";
foreach (@num) {
        if ($_ =~ / n /) {
                print "<TD align='CENTER'bgcolor='#990000'><H4><font
color='#FFFFFF'>$num_total</font></TD>";
        }elsif ($_ =~ /Porcentaje/) {printf ("<TD align='CENTER'bgcolor='#990000'><H4><font
color='#FFFFFF'>%.2f %</font>",$promediogral); print"</TD>"}
        else{print "<TD> </TD>";next}
}
print "</TR>";
```

Se cierra la tabla

```
print "</TABLE>";
print "<P>";
```

Se finaliza la conexión a la base de datos

```
$sth->finish();
$dbh->disconnect || warn "\nFallo al desconectar.\nError: $DBI::errstr\n";
```

Función Javascript para imprimir la página HTML generada en una impresora conectada al equipo.

```
print "<BR><h4 align='center'>Para imprimir presione <a href=\"#\" onClick=\"javascript:window.print()\">aqui</A></h4>";
```

Termina el documento HTML

```
print $query->end_html;
```

Las siguientes pantallas muestran las consultas generadas dinámicamente a través de la página.

— Consulta 1. Promedio de Edad y número total de miembros por sexo masculino en la región México (Figura 4.9 y 4.10)

— Consulta 2. Numero de Jefe de Familia de genero masculino y femenino que tienen la Ocupación principal de Agricultor y una Escolaridad entre 5 y 10 años por cada uno de los Proyectos (Figura 4.11 y 4.12)

— Consulta 3. Alfabetismo de todos los miembros de la familia en la región de Centroamérica (Figura 4.13 y 4.14)

En estas consultas son una muestra de la capacidad, flexibilidad y de robustez que tiene el lenguaje de programación Perl para interactuar con bases de datos e incluso otros lenguajes de programación como Javascript y HTML.

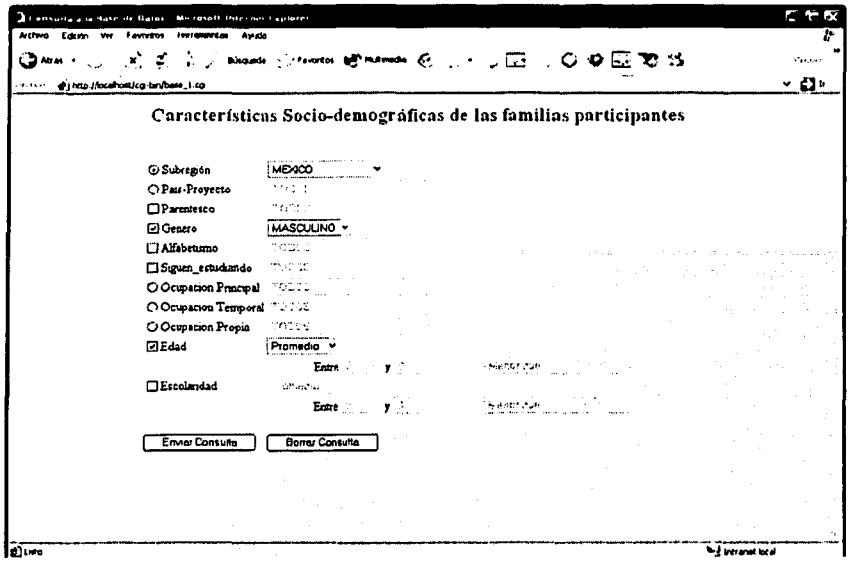


Fig 4.9 Consulta 1. Promedio de edad y número total de miembros por sexo masculino en la región México

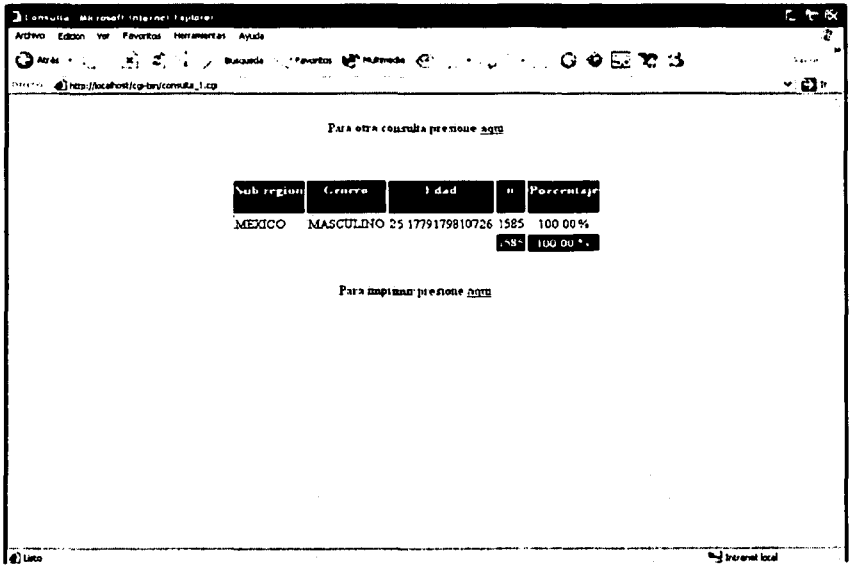


Fig 4.10 Datos generados por la consulta 1

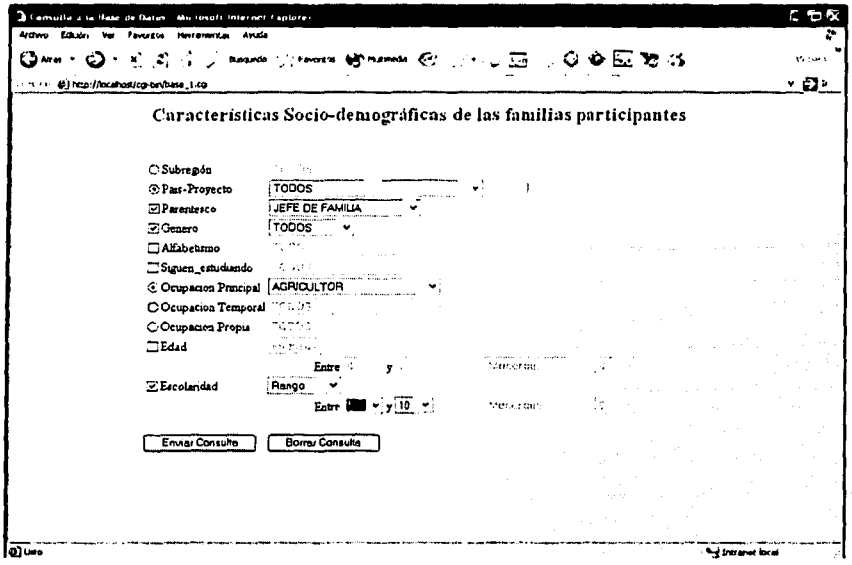
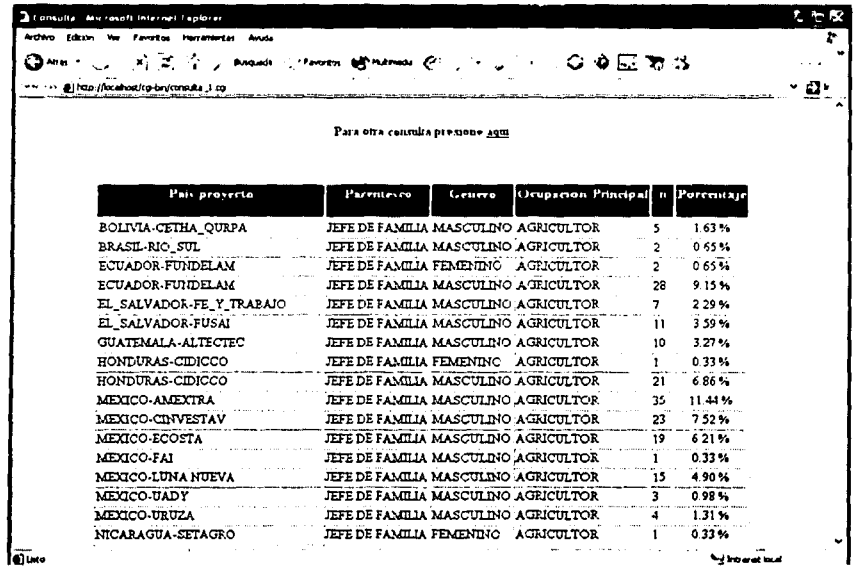


Fig 4.11 Consulta 2. Numero de jefes de familia de género masculino y femenino que tienen la ocupación principal de agricultor y una escolaridad entre 5 y 10 años por cada uno de los proyectos



Para otra consulta presione aquí

País_proyecto	Parentesco	Genero	Ocupacion Principal	n	Porcentaje
BOLIVIA-CETHA_QURPA	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	5	1.63 %
BRASIL-RIO_SUL	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	2	0.65 %
ECUADOR-FUNDELAM	JEFE DE FAMILIA	FEMENINO	AGRICULTOR	2	0.65 %
ECUADOR-FUNDELAM	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	28	9.15 %
EL_SALVADOR-FE_Y_TRABAJO	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	7	2.29 %
EL_SALVADOR-FUSAI	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	11	3.59 %
GUATEMALA-ALTECTEC	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	10	3.27 %
HONDURAS-CIDICCO	JEFE DE FAMILIA	FEMENINO	AGRICULTOR	1	0.33 %
HONDURAS-CIDICCO	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	21	6.86 %
MEXICO-AMEXTRA	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	35	11.44 %
MEXICO-CINVESTAV	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	23	7.52 %
MEXICO-ECOSTA	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	19	6.21 %
MEXICO-FAI	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	1	0.33 %
MEXICO-LUNA NUEVA	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	15	4.90 %
MEXICO-UADY	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	3	0.98 %
MEXICO-URUZA	JEFE DE FAMILIA	MASCULINO	AGRICULTOR	4	1.31 %
NICARAGUA-SETAGRO	JEFE DE FAMILIA	FEMENINO	AGRICULTOR	1	0.33 %

Fig 4.12 Datos generados por la consulta 2

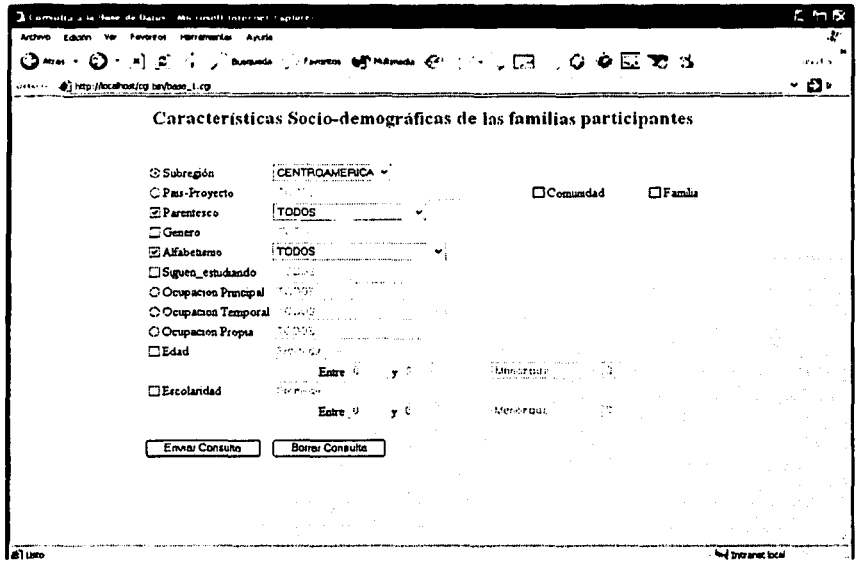
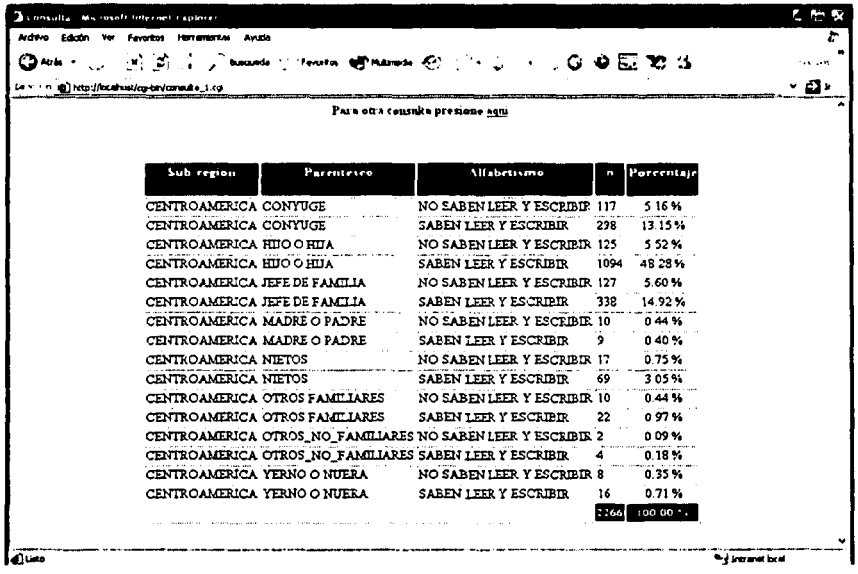


Fig 4.13 Consulta 3. Alfabetismo de todos los miembros de la familia en la región de Centroamérica



Para otra consulta presione aqui

Sub region	Parentesco	Alfabetismo	n	Porcentaje
CENTROAMERICA	CONYUGE	NO SABEN LEER Y ESCRIBIR	117	5.16 %
CENTROAMERICA	CONYUGE	SABEN LEER Y ESCRIBIR	298	13.15 %
CENTROAMERICA	HIJO O HIJA	NO SABEN LEER Y ESCRIBIR	125	5.52 %
CENTROAMERICA	HIJO O HIJA	SABEN LEER Y ESCRIBIR	1094	48.28 %
CENTROAMERICA	JEFE DE FAMILIA	NO SABEN LEER Y ESCRIBIR	127	5.60 %
CENTROAMERICA	JEFE DE FAMILIA	SABEN LEER Y ESCRIBIR	338	14.92 %
CENTROAMERICA	MADRE O PADRE	NO SABEN LEER Y ESCRIBIR	10	0.44 %
CENTROAMERICA	MADRE O PADRE	SABEN LEER Y ESCRIBIR	9	0.40 %
CENTROAMERICA	NIETOS	NO SABEN LEER Y ESCRIBIR	17	0.75 %
CENTROAMERICA	NIETOS	SABEN LEER Y ESCRIBIR	69	3.05 %
CENTROAMERICA	OTROS FAMILIARES	NO SABEN LEER Y ESCRIBIR	10	0.44 %
CENTROAMERICA	OTROS FAMILIARES	SABEN LEER Y ESCRIBIR	22	0.97 %
CENTROAMERICA	OTROS_NO_FAMILIARES	NO SABEN LEER Y ESCRIBIR	2	0.09 %
CENTROAMERICA	OTROS_NO_FAMILIARES	SABEN LEER Y ESCRIBIR	4	0.18 %
CENTROAMERICA	YERNO O NUERA	NO SABEN LEER Y ESCRIBIR	8	0.35 %
CENTROAMERICA	YERNO O NUERA	SABEN LEER Y ESCRIBIR	16	0.71 %
			2266	100.00 %

Fig 4.14 Datos generados por la consulta 3

CONCLUSIONES

La tecnología esta avanzando de un modo muy acelerado, cada día aparecen nuevos lenguajes de programación, otros se perfeccionan, y también se tienen que adaptar a las nuevas condiciones tecnológicas: las aplicaciones Web.

El lenguaje de programación Perl, aunque es un lenguaje relativamente nuevo, se ha desarrollado de una manera impresionante ya que existen desarrolladores en todo el mundo, los cuales han cooperado con la evolución de este lenguaje.

Perl es un lenguaje pensado para la manipulación de cadenas de caracteres, archivos y procesos. Esta manipulación se va simplificando por el importante número de operadores a disposición del usuario. El lenguaje Perl se percibe habitualmente como un lenguaje intermedio entre los *shell scripts* y la programación en "C".

El lenguaje Perl no es precompilado, pero aún así es más rápido que la mayoría de lenguajes interpretados. Esto se debe a que los programas en Perl son analizados e interpretados antes de su ejecución.

Estas características hacen que el mantenimiento y la depuración de un programa en Perl sean mucho más sencilla que el mismo programa escrito en lenguaje "C".

Cuando se planteó este trabajo de tesina, se había hecho de una manera muy teórica, pero al tener la oportunidad de desarrollarlo en forma práctica en un proyecto del Colegio de Postgraduados fue un gran reto para mi, ya que comprobaría si en verdad mi trabajo de investigación podría resolver un problema real y considero que si lo resolvió.

Durante la parte de la aplicación tuve que hacer un análisis de fondo ya que se construyeron bases de datos y se migraron las hojas de cálculo a estas bases. El proceso fue difícil ya que se necesitaron otros conocimientos aparte de los utilizados en este trabajo, pero la mayor parte de ellos tuvieron que ser implementados auxiliándose de herramientas informáticas y del uso del Internet

La conjunción de los temas anteriores permitieron generar un proyecto que tiene bajo costo, ya que todos los lenguajes de programación, manejador de base de datos, servidor Web, etc., son de distribución gratuita por lo que sólo se necesita el conocimiento que es adquirido en este tipo de trabajos y en artículos en Internet para poder construir aplicaciones sencillas pero a la vez robustas y confiables.

GLOSARIO

Applet	Un applet es una pequeña aplicación software, normalmente en un lenguaje de programación Java.
Bookmark	Es la dirección en internet (URL) de una página web, que se desea volver a ver. En Netscape se guardan los URLs en un archivo llamado bookmark. En Internet Explorer se guardan en el directorio llamado Favoritos (Favorites).
CGI	Abreviación de Common Gateway Interface, el CGI es un programa de interfaz que permite al servidor de Internet utilizar programas externos para realizar una función específica. También denominado pasarelas o CGI "scripts", estos programas consisten generalmente de una serie de instrucciones escritas en un lenguaje de programación como "C" o PERL que procesan la petición de un navegador, ejecutan un programa y formatean los resultados en HTML de manera que puedan ser presentados en el navegador. Se utilizan para añadir interactividad a una página web al permitir a los usuarios llenar y enviar formularios que podrán ser procesados (como un catálogo en línea), acceder a bases de datos por medio de una búsqueda, y obtener acceso a un sitio protegido escribiendo una contraseña.
Caché	Cuando se descarga una página web, el dato es "ocultado", lo que significa que es almacenado temporalmente en la computadora. La próxima vez que se desea esa página, en lugar de pedir el archivo al servidor, el navegador accede a ella a partir del caché, de manera que la página aparezca rápidamente. Pero si la página es actualizada frecuentemente, como lo son las páginas de noticias, de resultados deportivos o de datos financieros, no se verán las informaciones más recientes. Se ha de usar el botón de Recargar del navegador para descargar del servidor los datos más recientes.
Contraseña	Una contraseña es un código o una palabra que se utiliza para acceder a datos restringidos de una computadora. Mientras que las contraseñas crean una seguridad contra los usuarios no autorizados, el sistema de seguridad sólo puede confirmar que la contraseña es válida, y no si el usuario está autorizado a utilizar esa contraseña.
Descargar	En inglés: download. Descargar es el método mediante el cual los usuarios acceden y guardan programas u otros archivos en sus computadoras a partir de computadoras remotas, normalmente por medio de un módem.

Dirección IP	Una dirección IP es un código numérico que identifica a una computadora específica en Internet. Las direcciones de Internet son asignadas por un organismo llamado InterNIC. El registro incluye un nombre (whitehouse.gov), nombre de dominio, y un número (198.137.240.100), dirección o número IP.
Driver	Manejador ó Controlador de un dispositivo
Hipertexto	Hipertexto se refiere a cualquier texto disponible en el World Wide Web que contenga enlaces con otros documentos. Utilizar el hipertexto es una manera de presentar información en la cual texto, sonido, imágenes y acciones están enlazadas entre sí de manera que se pueda pasar de una a otra en el orden que se desee.
Host	Un host, literalmente anfitrión, es una computadora directamente conectada a una red y que efectúa las funciones de un servidor, y alberga servicios, como correo electrónico, grupos de discusión Usenet, FTP, o World Wide Web, accesibles por otras computadoras de la red.
Host Name	Nombre de sistema central. Toda computadora que está conectado directamente a Internet tiene una identificación numérica, denominada dirección IP, y un nombre, llamado host name. La mayoría de la gente que utiliza el Internet no necesita saber el host name de una computadora para conectarse a ella. Todo lo que se necesita conocer es los URLs y las direcciones de correo electrónicos
HTML	Siglas de Hypertext Markup Language. El HTML es el lenguaje informático utilizado para crear documentos hipertexto. El HTML utiliza una lista finita de etiquetas, o tags, que describe la estructura general de varios tipos de documentos enlazados entre sí en el World Wide Web.
http	Http son las siglas de HyperText Transfer Protocol, el método utilizado para transferir archivos hipertexto por Internet. En el World Wide Web, las páginas escritas en HTML utilizan el hipertexto para enlazar con otros documentos. Al pulsar en un hipertexto, se salta a otra página web, archivo de sonido, o imagen.
Java	Java es un lenguaje de programación por objetos creado por Sun Microsystems, Inc. que permite crear programas que funcionan en cualquier tipo de computadora y sistema operativo. Se usa el Java para crear programas especiales denominados applets, que pueden ser incorporados en páginas web para hacerlas interactivas. Los applets Java requieren que el navegador utilizado sea compatible con Java.

JavaScript	JavaScript es un lenguaje scripting que permite hacer que los documentos HTML sean dinámicos, por ejemplo haciendo que el relieve de un botón cambie al posicionar el cursor sobre éste.
Login Name	Un login name es el identificador del usuario requerido al acceder a un sistema operativo. También es conocido como nombre o identificador del usuario.
MIME	Siglas de Multipurpose Internet Mail Extension. Sistema que permite integrar dentro de un mensaje de correo electrónico archivos binarios (imágenes, sonido, programas ejecutables, etc.).
Navegador	Un navegador es un programa software que permite ver e interactuar con varios tipos de recursos de Internet disponibles en el World Wide Web.
Nombre usuario	El nombre de usuario es el mismo que el login name. Es el nombre por del el cual el usuario y su buzón de correo electrónico son identificados en linea.
Página web	Una página web es un documento creado en formato HTML (Hypertext Markup Language) que es parte de un grupo de documentos hipertexto o recursos disponibles en el World Wide Web. Una serie de páginas web componen lo que se llama un sitio web. Los documentos HTML, que estén en Internet o en el disco duro de la computadora, pueden ser leídos con un navegador. Los navegadores leen documentos HTML y los visualizan en presentaciones formatedas, con imágenes, sonido, y video en la pantalla de una computadora. Las páginas web pueden contener enlaces hipertexto con otros lugares dentro del mismo documento, o con otro documento en el mismo sitio, o con documentos de otros sitios web.
Protocolo	Un protocolo es una serie de reglas que utilizan dos computadoras para comunicar entre sí. Cualquier producto que utilice un protocolo dado debería poder funcionar con otros productos que utilicen el mismo protocolo.
Script	Un script es un tipo de programa que consiste de una serie de instrucciones que serán utilizadas por otra aplicación.
Servidor	Un servidor es una computadora que trata las peticiones de datos, el correo electrónico, la transferencia de archivos, y otros servicios de red realizados por otras computadoras (clientes).

TCP/IP

TCP/IP son las siglas de Transmission Control Protocol/Internet Protocol, el lenguaje que rige todas las comunicaciones entre todas las computadoras en Internet. TCP/IP es un conjunto de instrucciones que dictan cómo se han de enviar paquetes de información por distintas redes. También tiene una función de verificación de errores para asegurarse que los paquetes llegan a su destino final en el orden apropiado.

IP, Internet Protocol, es la especificación que determina hacia dónde son encaminados los paquetes, en función de su dirección de destino. TCP, o Transmission Control Protocol, se asegura de que los paquetes lleguen correctamente a su destino. Si TCP determina que un paquete no ha sido recibido, intentará volver a enviarlo hasta que sea recibido correctamente.

URL

Siglas de Uniform Resource Locator. Es la dirección de un sitio o de una fuente, normalmente un directorio o un archivo, en el World Wide Web y la convención que utilizan los navegadores para encontrar archivos y otros servicios distantes.

World Wide Web

Literalmente "tela de araña mundial", más conocida como web. Existen tres descripciones principales: Serie de recursos (Gopher, FTP, http, telnet, Usenet, WAIS, y otros) a los que se puede acceder por medio de un navegador. Serie de archivos hipertexto disponibles en servidores del web y Serie de especificaciones (protocolos) que permiten la transmisión de páginas web por Internet.

Se puede considerar el web como una serie de archivos de texto y multimedia y otros servicios conectados entre sí por medio de un sistema de documentos hipertexto.

ANEXO A. TAGS HTML

El HTML esta compuesto de una serie de elementos que definen un documento y dirigen su visualización. Los siguientes tags son básicos para la construcción de una página Web. Si se desea mayor información consultese la referencia bibliografica de esta tesina.

Un elemento del HTML puede incluir un nombre, algunos atributos y cierto texto ó hipertexto y aparecerá dentro del documento HTML de la siguiente forma:

<tag> texto </tag>

<tag nombre_atributo=argument> texto </tag>

ó solamente <tag>

Por ejemplo:

<title> Mi pagina Personal </title>

* texto *

Un documento HTML esta compuesto de un solo elemento:

<html> ... </html>

es decir, turnadamente de una cabecera y un cuerpo de elementos

<head> ... </head>

y

<body> ... </body>

Elementos puestos generalmente entre el elemento head

<title> </title>

Especifica el título del documento. Este título aparecerá en la barra superior de la ventana que lo identifica con el contendido mostrado

<base href="URL">

Especifica el nombre del archivo en relación con la ruta donde se encuentra.

<link rev="RELATIONSHIP" rel="RELATIONSHIP" href="URL">

El tag link permite definir relaciones entre los documentos conteniendo el tag de liga y el documento especificado en el URL. El atributo rel especifica la relación entre el archivo HTML y el URL. El atributo rev especifica la relación entre el URL y el archivo HTML.

Elementos puestos generalmente entre el elemento body

Elementos de Texto:

<p>
El final de un párrafo que será formateado antes que se visualice en pantalla.

<pre> ... </pre>
Identifica el texto que ha sido formateado (preformateado) por un cierto sistema y debe ser mostrado tal como esta.

<blockquote> ... </blockquote>
Incluye una sección de texto obtenida de otra fuente.

Hipervínculos o Anclas

* ... *
Define la localización de una etiqueta dentro del documento.

* ... *
Define la liga para la localización de una etiqueta dentro del propio documento.

* ... *
Es la liga para otro archivo o recurso.

* ... *
Define una liga a una etiqueta en otro documento.

Títulos

* ... *	Título más destacado
* ... *	
* ... *	
* ... *	
* ... *	
* ... *	Título menos destacado

Estilos lógicos

* ... *	Énfasis
<code> ... </code>	Despliega el texto como una directiva HTML
<cite> ... </cite>	Muestra el texto en forma de cita

Estilos físicos

* ... *	Tipo de letra en negritas
* ... *	Tipo de letra en itálica
<u> ... </u>	Tipo de letra subrayada

Lista desordenada

<ul>
<li> Primer elemento en la lista
<li> Siguiente elemento en la lista
*
*

Lista ordenada

<ol>
<li> Primer elemento de la lista
<li> Segundo elemento de la lista
*
*

Formularios HTML

Los siguientes tags son implementados en los formularios

- *<form> . . . </form>*
- *<input>*
- *<select> . . . </select>*
- *<option>*
- *<textarea> . . . </textarea>*

En el punto 1.6.3. Formularios HTML de esta tesina, se encuentran los atributos y los argumentos de los tags anteriores.

Tablas

<table>...</table>

Las tablas son posiblemente la manera más clara de organizar la información. Sus atributos son:

border

Especifica el grosor del borde que se dibujará alrededor de las celdas. Por defecto es cero, lo que significa que no dibujará borde alguno.

cellspacing

Define el número de pixels que separarán las celdas.

cellpadding

Especifica el número de pixels que habrá entre el borde de una celda y su contenido.

width

Especifica la anchura de la tabla. Puede estar tanto en pixels como en porcentaje de la anchura total disponible para él ("100%" indica que ocupará todo el ancho de la ventana del navegador).

align

Alinea la tabla a izquierda (LEFT), derecha (RIGHT) o centro (CENTER).

Definir las filas de la tabla

<tr>... </tr>

Cada fila se define con una etiqueta TR, que tiene los siguientes atributos:

align

Alinea el contenido de las celdas de la fila horizontalmente a izquierda (LEFT), derecha (RIGHT) o centro (CENTER).

valign

Alinea el contenido de las celdas de la fila verticalmente arriba (TOP), abajo (BOTTOM) o centro (MIDDLE).

Definir las celdas de la tabla

<td> ... </td> <th> ... </th>

Las etiquetas TD y TH son equivalentes, pero la última se utiliza para encabezados, de modo que su interior se escribirá por defecto en negrita y centrado. Estos son los atributos de ambas:

align

Alinea el contenido de la celda horizontalmente a izquierda (LEFT), derecha (RIGHT) o centro (CENTER).

valign

Alinea el contenido de la celda verticalmente arriba (TOP), abajo (BOTTOM) o centro (MIDDLE).

width

Especifica la anchura de la celda. También se puede especificar tanto en pixels como en porcentaje, teniendo en cuenta que, en este último caso, será un porcentaje respecto al ancho total de la tabla (no de la ventana del navegador).

nowrap

Impide que, en el interior de la celda, se rompa la línea en un espacio.

colspan

Especifica el número de celdas de la fila situadas a la derecha de la actual que se unen a ésta (incluyendo la celda en que se declara este parámetro). Es por defecto uno. Si se pone igual a cero, se unirán todas las celdas que queden a la derecha.

rowspan

Especifica el número de celdas de la columna situadas debajo de la actual que se unen a ésta.

Tags Misceláneos

<!-- Texto -->

Comentarios dentro de los documentos HTML.

<address> . . . </address>

Muestra información de la dirección.

*
*

Salto de línea.

<hr>

Dibuja una línea.

<img src="URL" alt="Texto Alterno">

Coloca una imagen en del documento. Tiene los siguientes atributos:

src

Especifica la localización física de la imagen.

alt

Permite mostrar una cadena de texto en lugar de colocar la imagen y funciona en los navegadores que no pueden mostrar imágenes.

align

Especifica la relación con el texto adyacente. El argumento para align puede ser top, middle o bottom.

ismap

Especifica un mapa de imágenes. Es decir, diferentes partes de ella llevan a diferentes recursos.

Métodos del modulo DBI

- **connect (conectar)**

```
$dbh = DBI->connect("DBI:mysql:$database", undef, undef);
$dbh = DBI->connect("DBI:mysql:$database", $username, $password);
$dbh = DBI->connect("DBI:mysql:$database:$hostname", $username, $password);
$dbh = DBI->connect("DBI:mysql:$database:$hostname:$port",
$username, $password);
```

Se usa el método connect para crear una conexión con la base de datos a la fuente de datos. La variable $data_source debe empezar con 'DBI:nombre_del_driver:'. Si el username (nombre de usuario) y/o el password están indefinidos, entonces DBI usará los valores de las variables de entorno DBI_USER, DBI_PASS respectivamente. Si no se especifica un nombre de host (hostname), se utilizará por defecto "localhost".

- **available_drivers (drivers disponibles)**

```
@drivers = DBI->available_drivers;
@drivers = DBI->available_drivers($quiet);
```

Este método devuelve un array con los drivers disponibles, buscándolos en los directorios DBD::* .

- **data_sources (fuentes de datos)**

```
@databases = DBI->data_sources($driver);
```

Este método devuelve un array de bases de datos disponibles para el driver dado (variable $driver). Si dicha variable $driver se omite, se utiliza la variable de entorno DBI_DRIVER

- **trace (seguimiento)**

```
DBI->trace($trace_level);
DBI->trace($trace_level, $trace_file);
```

La información en DBI puede seguirse (trace) para todos los manejadores que usen éste método. Para habilitarlo para un manejador específico hay que usar el método $dbh->trace similar para el manejador. Poniendo la variable $trace_level a 2 se pude ver información más detallada, si se pone a 0

desaparece la información de seguimiento (trace). Si el archivo $trace_file está especificado, entonces toda la información se añade a dicho archivo.

Funciones de utilidades de DBI

- **neat**

  $str = DBI::neat($value, $maxlen);

  Devuelve una representación limpia (neat) de la variable ($value) que se le pasa, es decir, formateada para uso humano. Esta función se usa internamente por el DBI para el seguimiento (trace). No debe ser usada para formatear valores para usarlos en la base de datos.

- **neat_list**

  $str = DBI::neat_list(\@listref, $maxlen, $field_sep);

  Llama a DBI::neat para cada elemento de @listref y devuelve una cadena con los resultados unidos por $field_sep. No debe ser usada para formatear valores para usarlos en la base de datos.

- **dump_results**

  $rows = DBI::dump_results($sth, maxlen, $lsep, $fsep, $fh);

  Esta función extrae todas las tuplas de una manejador de sentencia ($sth), llama a la función DBI::neat_list para cada tupla y escribe el resultado en el descriptor de archivo ($fh), que por defecto es STDOUT. El separador de línea ($lsep) es por defecto '\n', el separador de campos ($fsep) es por defecto ',' y la máxima longitud ($maxlen) es 35.

Atributos dinámicos de DBI

Todos los atributos dinámicos descritos a continuación son volátiles, es decir, tienen una vida corta debido a que siempre están asociados al último manejador usado. El motivo es que tiene que ser usados inmediatamente después de llamar el método que los activa. También se han encontrado problemas con multi-hebras en Perl 5.005. Es mejor usar los atributos equivalentes que se menciona a continuación

- **$DBI::err**

  Equivalente a $dbh->err

- **$DBI::errstr**

  Equivalente a $dbh->errstr

- **$DBI::state**

  Equivalente a $dbh->state

- **$DBI::rows**

  Equivalente a $dbh->rows

Métodos comunes a todos los Manejadores

- **err (error de la BD)**

  $err = $dbh->err;

  Este método devuelve el código de error nativo de la base de datos, provocado a la última función del driver llamada.

- **errstr (cadena de error de la BD)**

  $errstr = $dbh->errstr;

  Este método devuelve el mensaje de error nativo de la base de datos, debido a la última función del driver llamada.

- **state(estado)**

  $state = $dbh->state;

- **trace (seguimiento)**

  $dbh->trace($trace_level);

  Muy similar a DBI::trace con una expresión. El seguimiento (trace) sólo está asociado con el manejador específico con el que está siendo usado.

- **func**

  $dbh->func(@func_arguments, $func_name);

  Este método se usa para llamar a métodos privados que están implementados por el driver (DBD::*)

Atributos comunes a todos los Manejadores

La mayoría de estos atributos son heredables, es decir, cualquier manejador hijo los heredará de sus padres. (Por ejemplo los manejadores de sentencias los heredarán de los manejadores de la BD que los crea).

```
$dbh->{AttributeName} = 1;
$value = $dbh->{AttributeName};
```

- **Warn (booleano, heredado)**

  Activa mensajes de aviso útiles. Activado por defecto.

- **CompatMode (booleano, heredado)**

  Usado para emulación de capas para activar el modo compatible en el drivers. No aplicables al driver de mySQL.

- **InactiveDestroy (booleano)**

  Este atributo es usado para inhabilitar los efectos de destruir un manejador. Está específicamente diseñado para aplicaciones en Unix que tienen procesos hijos creados con fork. EL padre o el hijo deben poner este atributo en todos sus manejadores, pero no los dos.

- **PrintError (booleano, heredado)**

  Este atributo se usa para forzar a los errores a generar mensajes de aviso además de los códigos de error. Por defecto DBI->connect tiene activado PrintError.

- **RaiseError (booleano, heredado)**

  Este atributo es usado para obligar a los errores a crear excepciones justo antes de devolver un código de error. Por defecto, no está activado.

- **ChopBlanks (booleano, heredado)**

  Este atributo se usa para controlar los espacios en los campos de anchura fijos. Por defecto es falso.

- **LongReadLen (entero, heredado)**

  Este atributo se usa para controlar la máxima longitud del los campos blob (binarios). Una valor de 9 significa que automáticamente no se extraen datos

largos (la extracción debe devolver undef para el blob cuando está a 0. Por defecto es, normalmente, 0, pero puede haber diferencias entre drivers. Este atributo debe ser activado antes que la sentencia esté en la etapa de preparación.

- **LongTruncOk (booleano, heredado)**

  Este atributo se usa para controlar como se extrae (fetch) manejadores con campos blob (binarios) que han sido truncados (debido a ser el valor del campo más largo que el atributo LongReadLen). Por defecto es falso y provocará que la extracción (fetch) falle. Muchos drivers permiten continuar extrayendo más tuplas cuando este atributo esta puesto a falso.

Métodos de los Manejadores de BD

- **prepare (preparación)**

  $sth = $dbh->prepare($statement);
  $sth = $dbh->prepare($statement, \%attr);

  Este método prepara una sentencia simple de SQL para su ejecución y devuelve una referencia al manejador de la sentencia ($sth) para usarlo con el objetivo de coger los atributos de la sentencia cuando se ejecute la sentencias. Algunos drivers sólo guardan la sentencia en el manejador y sólo puede dar información útil después de que el método execute haya sido llamado.

- **do**

  $rc = $dbh->do($statement);
  $rc = $dbh->do($statement, \%attr);
  $rc = $dbh->do($statement, \%attr, @bind_values);

  Este método prepara y ejecuta una sentencia. Devuelve el número de tupas afectadas o -1 si no es conocido, o undef si hay un error. Este método es usualmente usado para sentencias no-select (que no son consultas), las cuales no necesitan ser preparadas o ejecutadas por separado.

- **commit**

  $dbh->commit; Este método hace persistente una operación realizada en la BD (No soportado por mySQL).

- **rollback**

  $dbh->rollback; Deshace un commit (No soportado por mySQL).

- **disconnect**

  $dbh->disconnect;

  Este método cierra la conexión entre la BD y el manejador de la base de datos. Normalmente se suele usar justo antes de terminar la aplicación. Si se llama a este método mientras hay manejadores de sentencias activos se obtendrá un mensaje de aviso. Hay que usar el método finish para cada manejador de sentencia definida.

- **ping**

  $result = $dbh->ping;

  Este método mira si el servidor de BD sigue funcionando y la conexión sigue funcionando. No se suele usar. En Apache::DBI se pude ver un ejemplo de uso.

- **quote**

  $sql = $dbh->quote($string);

  Este método escapa caracteres especiales (comillas, etc.) en las cadenas y añade las comillas externas. No es aplicable a todos los tipos de entrada (ejemplo, datos binarios).

Atributos de los Manejadores de Bases de Datos

- **AutoCommit**

  $dbh->{AutoCommit} = 1;
  $value = $dbh->{AutoCommit}; (No suportado por mySQL)

Métodos de los Manejadores de Sentencias

- **bind_param**

  $sth->bind_param($param_num, $bind_value);
  $sth->bind_param($param_num, $bind_value, \%attr);
  $sth->bind_param($param_num, $bind_value, $bind_type);

  Este método se usa para asignar valores a un parámetro en una sentencia preparada. %attr puede ser usado para especificar el tipo de dato del parámetro.

- **bind_param_inout**

  $sth->bind_param_inout($param_num, \$bind_value, $max_len);
  $sth->bind_param_inout($param_num, \$bind_value, $max_len, \%attr);
  $sth->bind_param_inout($param_num, \$bind_value, $max_len, $bind_type); (no soportado por mySQL)

- **execute**

  $rc = $sth->execute;
  $rc = $sth->execute(@bind_values);

  Este método ejecuta una sentencia preparada y devuelve verdadero si hay éxito y undef si ocurre un error. Para las sentencias que no son SELECT (UPDATE, INSERT, etc.) el valor de retorno es el número de tuplas afectadas. Cero tuplas son devueltas como '0E0' que Perl trata como '0' pero que es tratado como verdadero. Para sentencias SELECT la ejecución comienza la maquinaria de consulta. Se necesitan usar uno de los métodos de extracción (fetch), explicados a continuación, para obtener datos. Si se omite algún argumento entonces el método execute llama a bind_param para cada valor, y pone el tipo a SQL_VARCHAR.

- **fetchrow_arrayref**

  $row = $sth->fetchrow_arrayref;

  Este método extrae la siguiente tupla con datos y devuelve una referencia a un array con los valores de los campos. Si no hay más tuplas que obtener, devuelve undef. Cuando es usado con el método bind_columns, es el modo más rápido de extraer datos.

- **fetchrow_array**

  @row = $sth->fetchrow_array;

  Es igual a fetchrow_arrayref excepto que devuelve un array con los valores de los campos, en lugar de una referencia a un array.

- **fetchrow_hashref**

  $row_hash = $sth->fetchrow_hashref;

  Otro método alternativo para extraer tuplas de datos. Este método devuelve una referencia a una memoria asociativa (hash) conteniendo para cada campo el nombre y el valor. Las claves de la memoria asociativa (hash) son los mismos que los nombres de los campos devueltos por el método *$sth->{NAME}*. Este método es menos eficiente que el anterior.

- **fetchall_arrayref**

  $table = $sth->fetchall_arrayref;

  Este método es usado para obtener todos los datos (tuplas) devueltos por la sentencia SQL. Devuelve una referencia a un array de arrays de referencias.

- **finish**

  $sth->finish;

  Este método es usado cuando no se va a extraer más datos de un manejador de sentencia, antes de que sea preparado de nuevo o destruido. Este método es el más usado para housekeeping interno (liberar recursos como cuando se bloquea la lectura). No hace falta llamar a este método si vas a destruirlo rehusar el manejador de sentencia. De todos modos es bueno tomar el hábito de usarlo.

- **rows**

  $rc = $sth->rows;

  Este método devuelve el número de tuplas afectadas por la última sentencia no-Select ejecutada.

- **bind_col**

  $sth->bind_col($column_number, \$var_to_bind);
  $sth->bind_col($column_number, \$var_to_bind, \%attr);

  Este método asigna una columna (campo) a una variable. Cuando una columna es captada (fetched) la correspondiente variable es automáticamente actualizada. Este hace que la extracción se muy eficiente.

- **bind_columns**

  $sth->bind_columns(\%attr, @list_of_refs_to_vars_to_bind);

  Este método llama a bind_col para cada columna de la sentencia SELECT.

Atributos para Manejadores de Sentencias

La mayoría de estos atributos son de sólo lectura. Algunos drivers no proporcionan los valores hasta después de ejecutar el método que ha sido llamado.

- **NUM_OF_FIELDS**

  $num_fields = $sth->{NUM_OF_FIELDS};

  Este atributo guarda el número de campos que la sentencia ya preparada devolverá

- **NUM_OF_PARAMS**

  $num_params = $sth->{NUM_OF_PARAMS};

  Este atributo almacena el número de parámetros (placeholders) que están preparados en la sentencia.

- **NAME**

  $names = $sth->{NAME};

  Este atributo devuelve una referencia a un array de nombres de campos para cada columna.

- **NULLABLE**

  $nulls = $sth->{NULLABLE};

  Este atributo devuelve una referencia a un array indicando cual es devuelto como NULL (verdadero/falso).

- **CursorName**

  $cursor_name = $sth->{CursorName}; (no soportado por mySQL)

  Devuelve el nombre del cursor asociado con el manejador de sentencia.

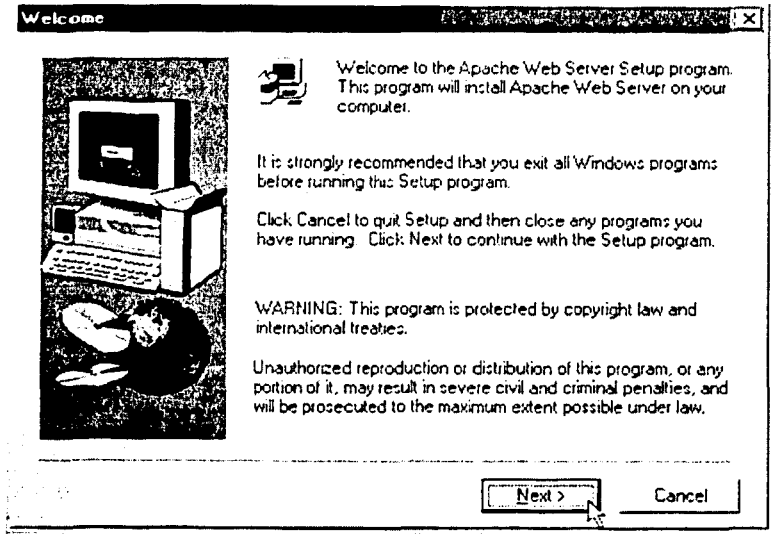
Apache Web Server

Apache Web Server esta diseñado para trabajar en Windows NT 4.0 y Windows 2000. El instalador del software sólo funcionará con procesadores de la familia x86, como los procesadores Intel. El instalador se puede descargar de la siguiente dirección en Internet: *http://www.apache.org/dist/httpd/binaries/win32/*

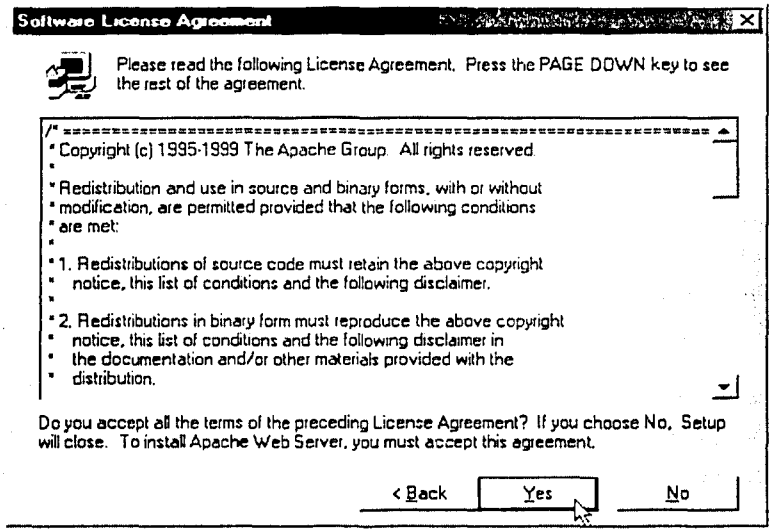
El servidor Apache Web Server también puede trabajar en Windows 95, Windows 98. Para estos casos debe estar instalado el protocolo de red TCP/IP. Para más información consultese la página *http://www.apache.org*. Si el servidor se instala en Windows 95, se debe actualizar la librería "Winsock2" antes de ejecutar Apache.

Para la versión de Apache para Windows se debe descargar un archivo binario con la extensión exe. Este es un archivo único y auto extraíble, que contiene al servidor apache listo para instalarse, por lo que se debe hacer dos veces click sobre el archivo binario (exe) y seguir los siguientes pasos:

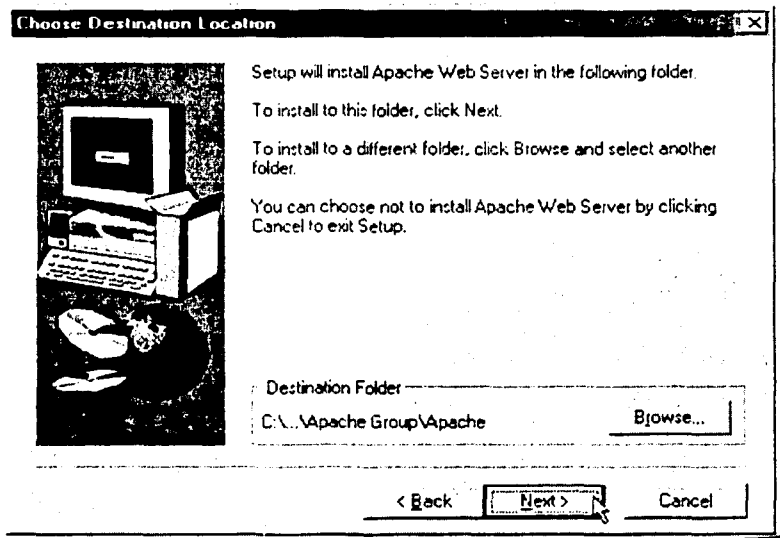
- Una vez comenzada la instalación, el programa desplegará la bienvenida, además avisa que se deben cerrar todos los programas que estén ejecutándose en ese momento y que está protegido por copyright (derechos de autor). Presionar Next (siguiente).



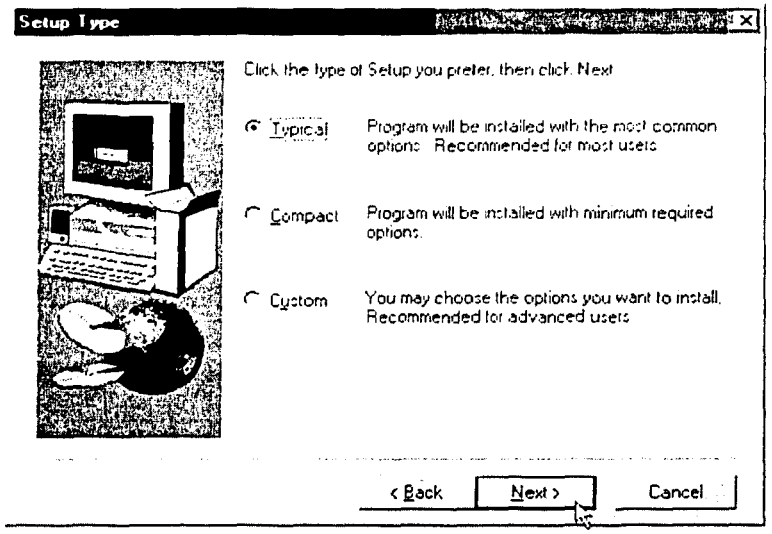
➢ A continuación aparecerá una ventana con las características de la Licencia, es muy importante leer estas condiciones. Cuado se este con la plena seguridad de utilizar este programa presionar el botón Yes (Si).



➢ El programa advierte que esta versión tiene algunos problemas de bugs conocidos y de estabilidad y que no es como la de Unix. Presionar Next. Ahora el programa pregunta donde se instalará el servidor, pudiéndose elegir un directorio diferente o el que se propone por default. Presionar Next.

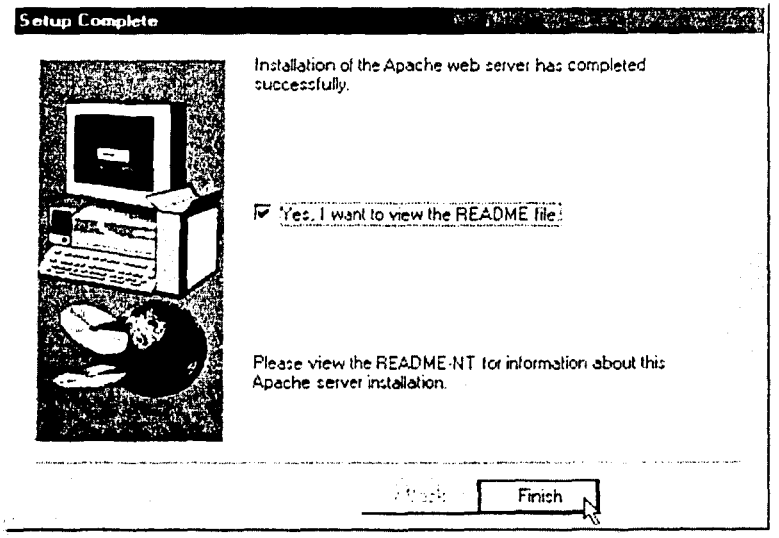


- Ahora pregunta el tipo de instalación que se requiere. La opción recomendada para la mayoría de los usuarios es la opción típica (Typical). Presionar Next .

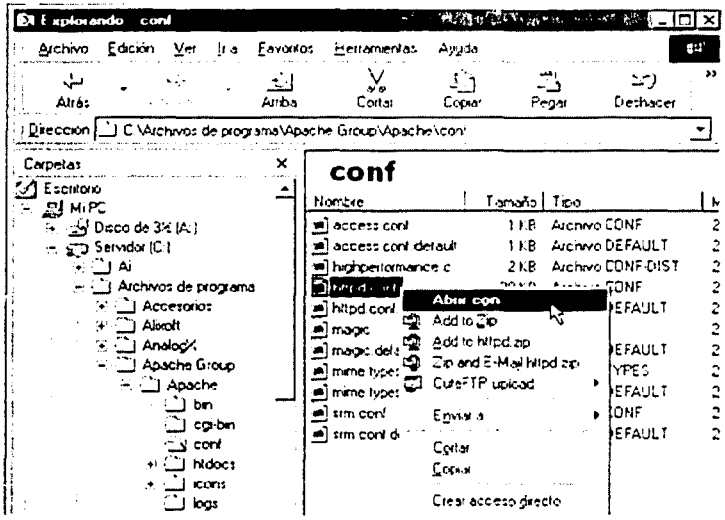


- También solicita el programa de instalación en qué grupo de programas dentro del Menú de Inicio de Windows aparecerá. A continuación Next.

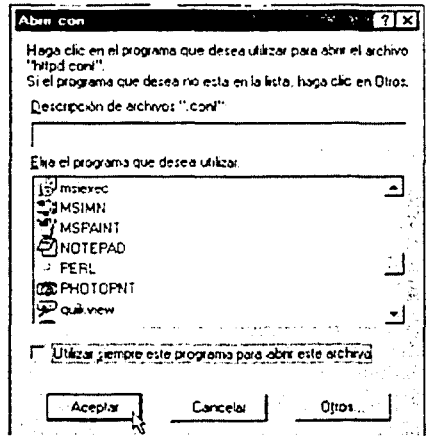
- Si todo va bien hasta aquí comenzará la instalación del programa y al término pulsar el botón Finish (Finalizado) para concluir la instalación.



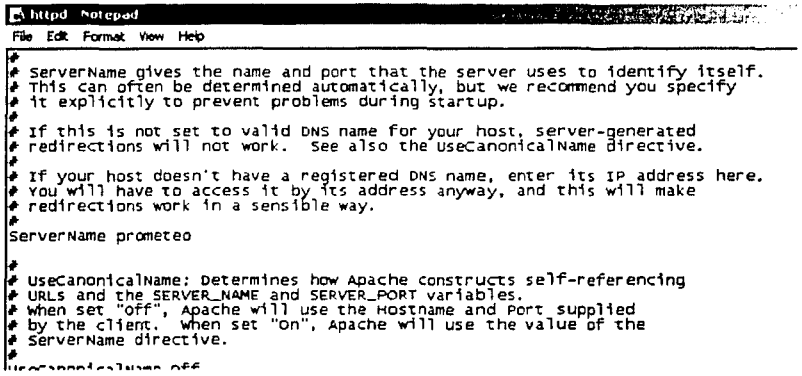
- Para las versiones de Windows queda una pequeña modificación por hacer en la configuración del programa para empezar a utilizar Apache Web Server. Abrir el explorador de Windows y localizar el fichero httpd.conf (se encuentra, en Archivos de Programa → Apache group → Apache → conf ó en el directorio Program Files → Apache group → Apache → conf). Una vez localizado, presionar el botón derecho del ratón sobre el archivo y Windows desplegará un menú emergente con varias opciones.



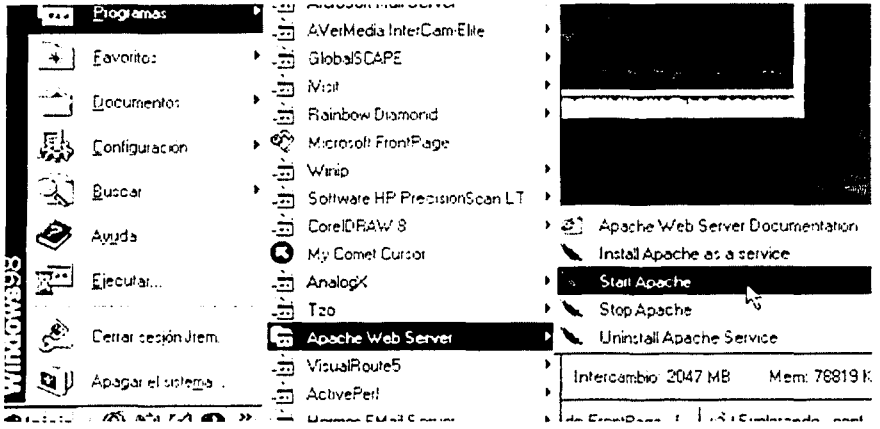
- Escoger la opción "Abrir con" y después seleccionar el programa NOTEPAD, ya que este archivo debe modificarse con algún editor de textos.



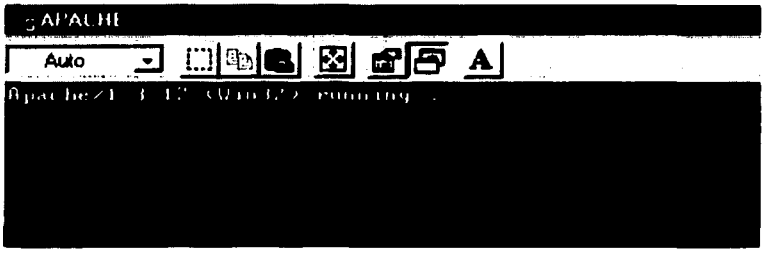
- Ahora se debe insertar una línea que contenga el texto *ServerName mi_nombre_de_servidor*, como por ejemplo: *Servername prometeo*. Después de colocar la línea guardar los cambios realizados en el archivo.



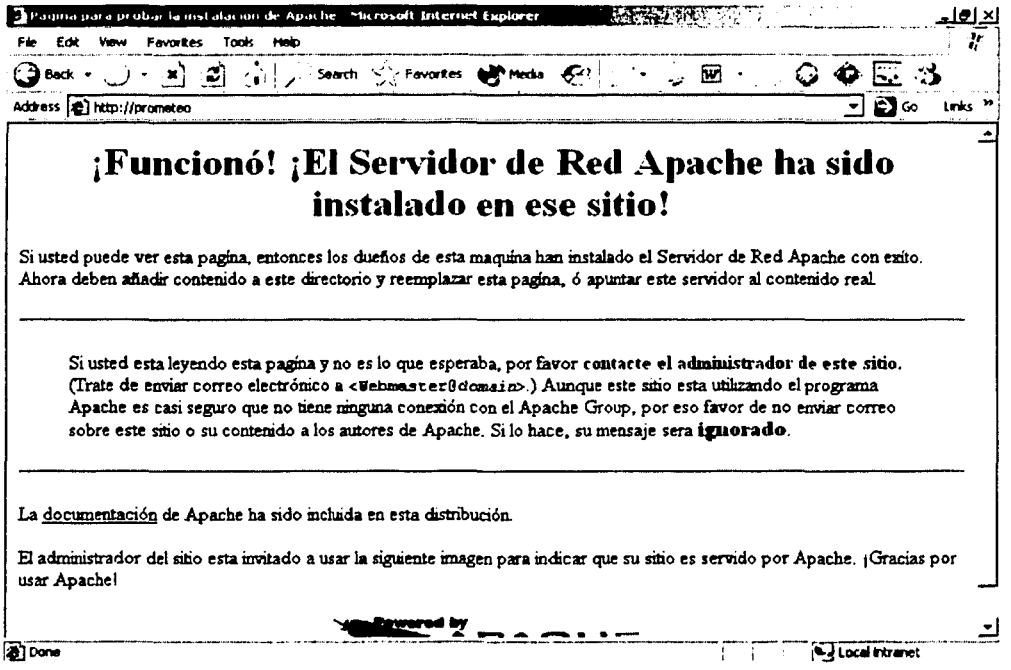
- El servidor esta listo para trabajar. Presionar el botón de INICIO → PROGRAMAS → APACHE WEB SERVER → START APACHE (Es conveniente que cuando se desconecte el servidor se haga mediante la opción STOP APACHE)



- Al iniciar aplicación se abrirá una ventana MS-DOS que indica que el servidor está funcionando



- Para verificar que todo este funcionando realizar la siguiente prueba: con el navegador Web (Internet Explorer ó Netscape Navigator) escribir en la barra de direcciones el nombre del servidor *http://mi_nombre_de_servidor* ó *http://prometeo* y en el navegador se mostrará la siguiente página Web.



- La página principal (*index.html*) del sitio debe ir colocada en la carpeta *htdocs* del directorio donde se instaló Apache Web Server.

- Para poder ejecutar los scripts cgi de Perl, estos deben ir almacenados en la carpeta *cgi-bin* del directorio donde se instaló Apache Web Server.

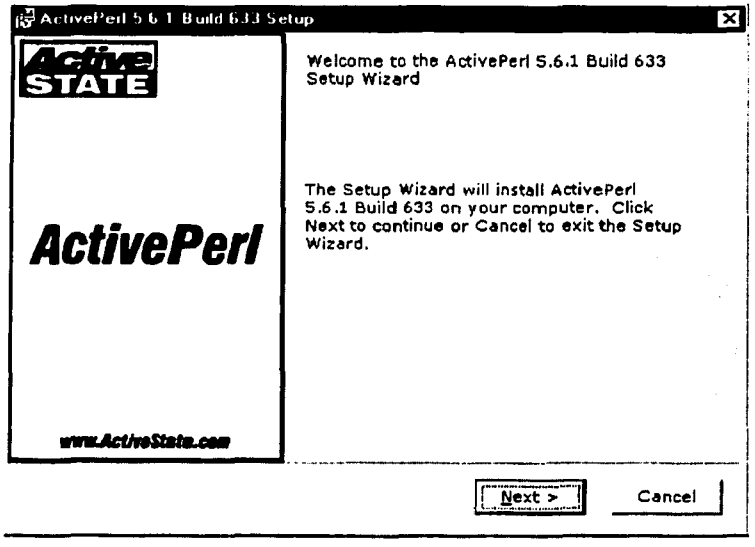
Perl y Módulos

Para ejecutar el archivo de instalación de Perl se debe contar con Microsoft Windows Installer 1.1+ (disponible en

http://download.microsoft.com/download/platformsdk/wininst/1.1/NT4/EN-S/InstMsi.exe)

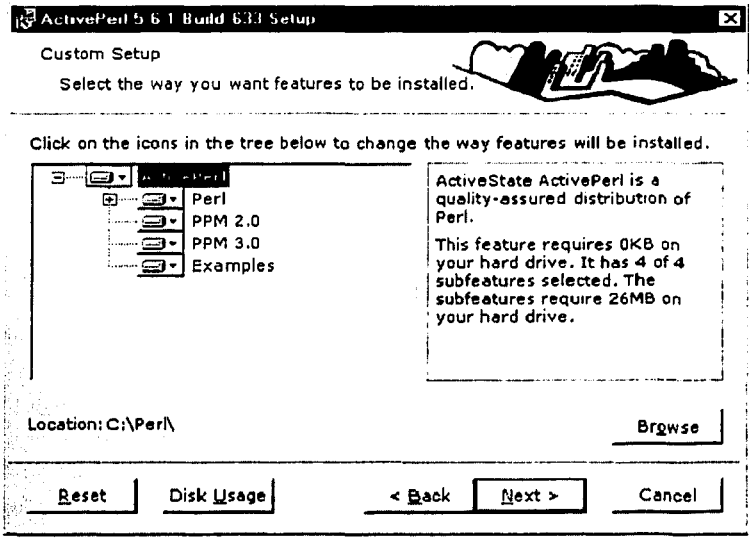
El archivo de instalación de Perl se puede bajar de la siguiente dirección de Internet: *www.activeperl.com*. Para iniciar el asistente de instalación que guiará este proceso ejecutar el paquete de instalación *ActivePerl-5.6.0.6xx.msi* dando dos veces click sobre de él. Cabe mencionar que este tipo de archivos .msi son archivos de instalación preparados para que sean ejecutados mediante Windows Installer. Si este software ya estaba instalado en el sistema o si se ha instalado ahora, el tipo de fichero msi será reconocido.

- La primera pantalla da la bienvenida a la instalación del software. Presionar Next.

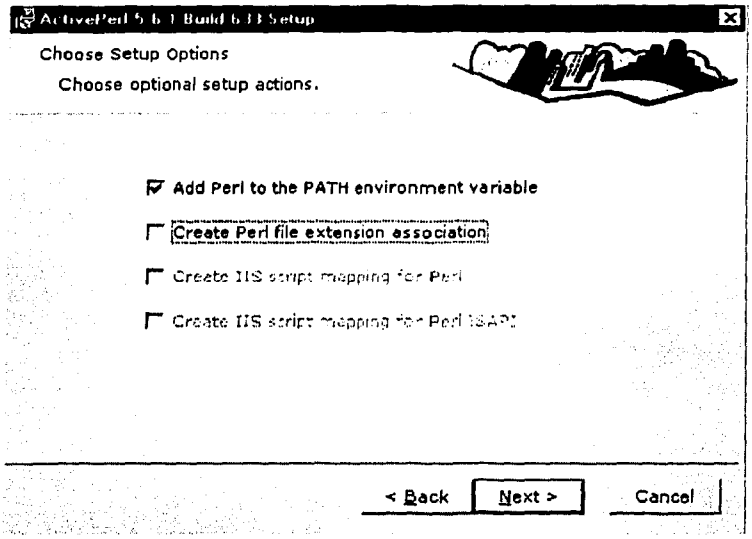


- En la siguiente pantalla el programa pregunta si se aceptan las condiciones de uso. De preferencia se deben leer las condiciones de uso y si se esta de acuerdo marcar la primera opción y presionar el botón Next.

- Por defecto el directorio donde se instalaran los archivos es *C:\Perl*, si se desea instalar el paquete en otro directorio diferente presionar el botón Browse y seleccionar el directorio deseado. El resto de opciones que aparecen en este cuadro de diálogo dejarlas tal y como están ya que se refieren a los ejecutables y documentación que serán instalados. Presionar el botón Next.



- A continuación se deben configurar algunos de los parámetros de la instalación. La primera casilla hará que el intérprete de Perl pueda ser ejecutado desde cualquier directorio donde sea invocado, esto es, será añadido a la variable PATH del sistema operativo actual. La segunda casilla pregunta si los scripts se deben ejecutar desde una consola MS-DOS y no haciendo dos veces clic sobre ellos. Las dos siguientes casillas requieren la instalación previa del Servidor Web Internet Information Server (IIS). Presionar Next.



- Una vez concluido el proceso de instalación, Presionar el botón en Finish para terminar la instalación

Para poder ejecutar los scripts en Windows abrir una consola de sistema (o ventana de MS-DOS) y teclear la siguiente sintaxis:

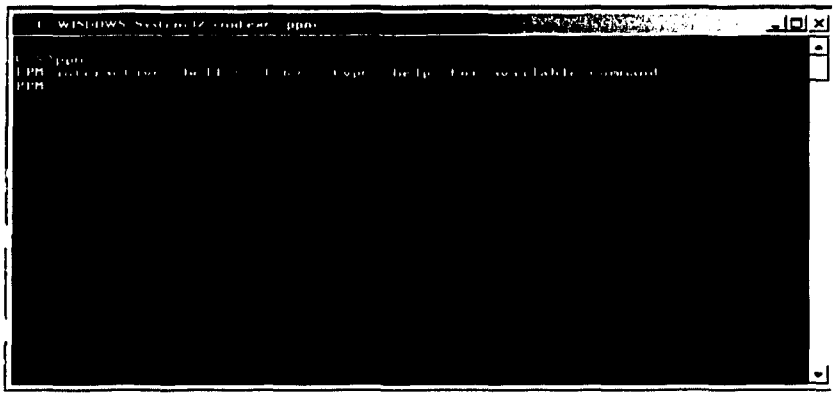
C:\>perl miScript.pl

Donde, obviamente, miScript.pl es el nombre del script que se va a ejecutar.

Instalación de módulos

Para instalar módulos en Perl, se requiere de una aplicación llamada PPM (Perl Packpage Manager) que provee una interfase de línea de comando para manipular los módulos Perl y extensiones (paquetes). PPM permite acceder a los repositorios del paquete, instala y quita los paquetes del sistema, y pone al día los paquetes que previamente fueron instalados usando las últimas versiones.

Para iniciar la aplicación abrir una consola MS-DOS y escribir "ppm".



Y para comenzar a instalar los módulos, se necesita:

— Tener una conexión a internet activa o

— Tener los módulos en la unidad de disco del equipo.

Para la primera opción consultese la documentación del Perl en la parte ActivePerl Components > PPM, para configurar la conexión de Firewall o Proxy en caso de ser necesario.

Para la segunda opción, los módulos se pueden descargar de Internet en la página *http://www.activestate.com/PPMPackages/zips/6xx-builds-only/.*

Para instalar los módulos con la primera opción escribir lo siguiente:

PPM> install nombre_modulo

En el caso de la segunda opción, crear un directorio en c:\perl llamado "paquetes" donde se va a descomprimir los módulos (ya que vienen en archivos comprimidos en formato zip). Para descomprimir los archivos zip en Windows 95,98 y NT/2000 se necesita el programa de Winzip que se puede descargar de la página *http://www.winzip.com*. Consultese documentación del mismo programa.

Cuando se extraen todos los archivos que vienen dentro del zip al directorio *c:\perl\paquetes*, verificar que vengan acompañados de varios archivos y directorios, entre estos un archivo con el nombre del modulo con la extensión pm.

El siguiente paso es instalar el modulo. Escribir en la ventana de línea de comandos (MSDOS).

PPM> install --location c:\perl\paquetes nombre_modulo

Con este proceso se pueden instalar todos los modulo que sean necesarios, como los por ejemplo los módulos ASP, Apache, HTML, CGI, etc.

Para mayor información acerca de la instalación, búsqueda, actualización entre otras funciones, teclear PPM> help, este comando mostrará en pantalla los comandos que existen en el PPM y posteriormente para la saber como utilizar estos comandos teclear

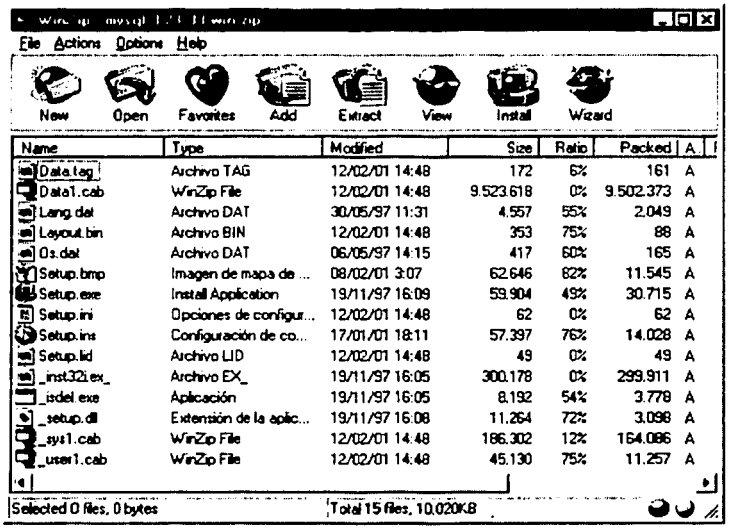
PPM>help nombre_comando

MySQL

Para comenzar la instalación de mysql descargar el archivo del programa de instalación de la siguiente dirección electrónica:

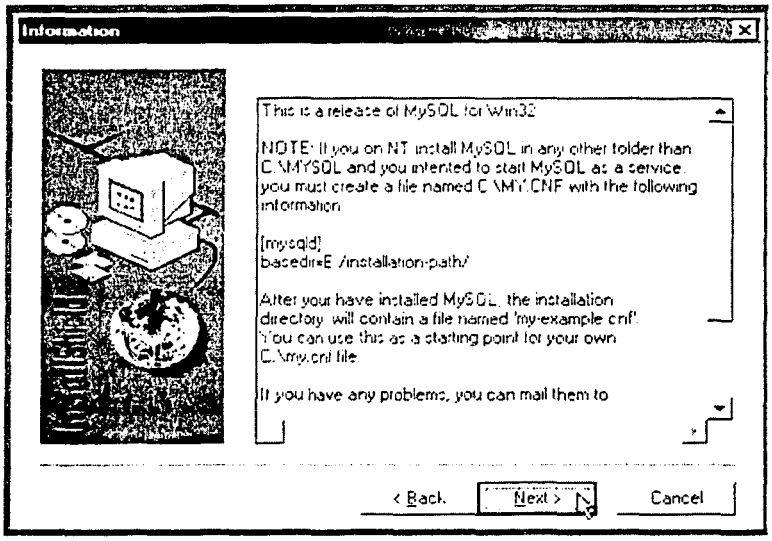
http://www.mysql.com/downloads/index.html

- En esta dirección electrónica se encuentra el archivo binario ó ejecutable para Windows y otros sistemas operativos. Este se desempaqueta en un directorio como por ejemplo *c:\Temp* con el programa Winzip.

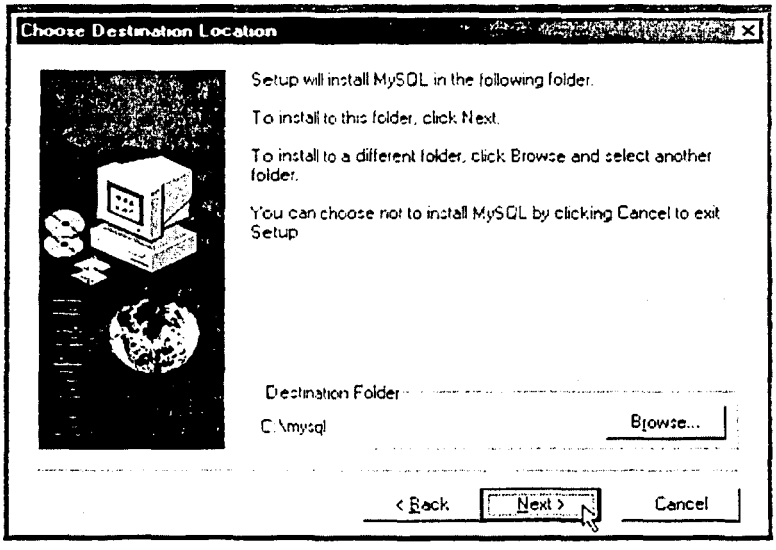


- Posteriormente buscar en el directorio antes mencionado el archivo setup.exe y hacer dos veces click sobre de él para comenzar la instalación. La primera pantalla es la Bienvenida que da el programa y recomienda cerrar todas las aplicaciones que se esten ejecutando en ese momento. A continuación Next

- Continua con una pantalla que contiene la información de la versión que se está instalando, para el caso de Windows NT si se desea instalar MySQL como un servicio se debe crear un archivo con el nombre de my.cnf que deberá incluir dos líneas con las siguientes características : *[mysqld]* y basedir=*C:/mysql/*.Presionar el botón Next.



- El directorio de instalación por default es c:\mysql. Si se desea instalar en otra ubicación presionar el botón Browse y elegir la carpeta deseada. Presionar botón Next.

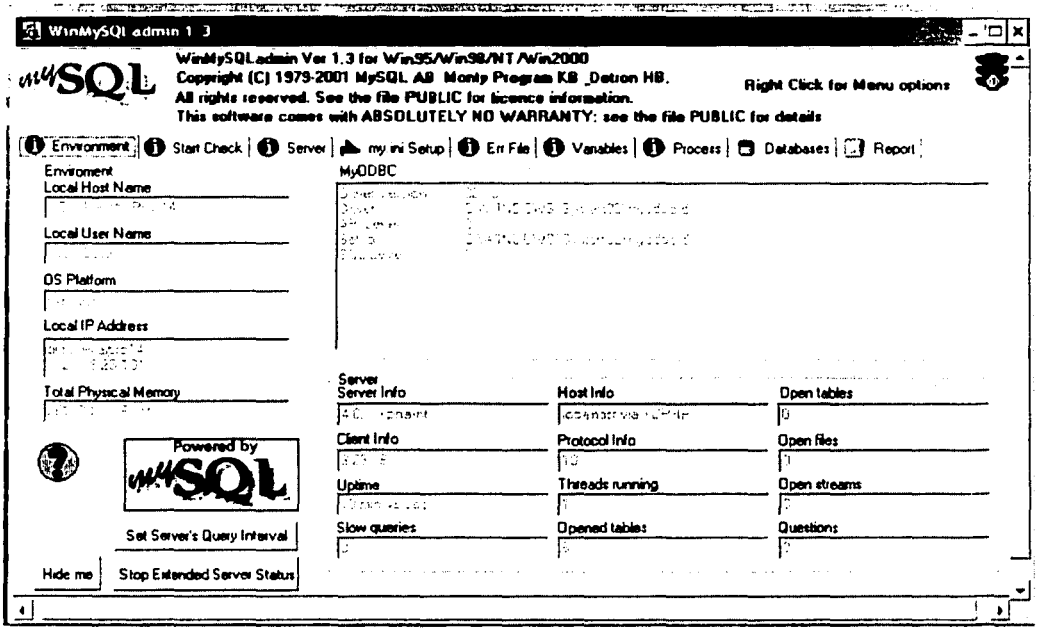


- El programa pregunta el tipo de instalación que se hará, en este caso la opción Typical es la recomendable. Presionar Next.

- El programa copia los archivos necesarios y termina la instalación. En este caso el programa no creo íconos de acceso directo en el escritorio de Windows. Pero se encuentra instalado en el directorio *C:\Mysql* ó el que se haya especificado.

- Para manejar MySQL existe una aplicación y esta localizada en el directorio *c:\mysql\bin\winmysqladmin.exe*. Cuando se ejecuta por primera vez solicita un nombre de usuario y un password para crear un archivo my.ini.

- Para arrancar MySQL existen tres archivos dentro del directorio *c:\mysql\bin*:
  - *mysqld.exe* que es el genérico de los sistemas 386 en adelante.
  - *mysqld-opt.exe* para equipos pentium y
  - *mysqld-nt.exe* para los basados en NT.



- Para detener MySQL se realiza mediante el comando *mysqlshutdown.exe.*

REFERENCIAS BIBLIOGRÁFICAS

Libros:

- Barkakati, N (1998), *Manual Fundamental de Perl 5*, Anaya Multimedia, Madrid España, 467 pp
- Castaño, A, Piattini, M (1993), *Concepción y Diseño de Bases de Datos. Del Modelo E/R al Modelo Relacional*, Addison-Wesley, E.U.A., 979 pp
- Date, CJ (1986), *Introducción a los Sistemas de Bases de Datos*, Addison-Wesley, México, 365 pp
- Descartes, A, Bunce, T (2000), *Programming the Perl DBI*, O'Reilly, E.U.A., 346 pp
- Guelich, S, Gundavaram, S, Birznieks, G (2000), *CGI Programming with Perl*, 2da ed, O'RELLY, E.U.A, 451 pp
- Kline, K, Kline, D (2001), *SQL in a nutshell. A Desktop Quick Reference*, O'Reilly, United States of America, 224 pp
- Medinets, D (1997), *Perl 5 a través de ejemplos*, Prentice Hall, México, 670 pp
- Phillips, A (1998), *Descubre HTML 4*, Prentice Hall, España, 584 pp
- Rocher, G (1999) *Traducción de Queries en Prolog a SQL.* Tesis Licenciatura, *Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas* Puebla, México

Artículos:

- Codd, E (1970), *A Relational Model of Data for Large Shared Data Banks*, **Communications of the ACM, 13**(6), 377-387.
  http://www.acm.org/classics/nov95/toc.html

En Internet:

- *http://hoohoo.ncsa.uiuc.edu/cgi/*
- *http://www.monografias.com*
- *http://www.activestate.com*
- *http://www.infase.es/FORMACION/HTML/cgi.html*
- *http://www.cpan.org*
- *http://www.perl.org*
- *http://www.perl.com*
- *http://html.programacion.net*
- *http://programacion.net*
- *http://www.banesto.es/banesto/manuhtml/castella/curso.htm*
- *http://www.recursoscgi.com*
- *http://www.bolnet.bo/cursohtml*
- *http://www.w3.org/Protocols/*
- *http://geneura.ugr.es/~javi/dbi/index.htm*
- *http://www.mysql.com/documentation/mysql/bychapter/index.html*
- *http://www.internautas.org/*
- *http://www.cc.ukans.edu/~acs/index.shtml*
- *http://www.learnthenet.com/spanish/glossary/glossary.htm*
- *http://www.linuxfocus.org/Castellano/January2002/article226.shtml*
- *http://www.mundojavascript.com*
- *http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm*
- *http://www.wall.org/~larry/perl.htm*