UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA
LICENCIATURA EN SISTEMAS COMPUTACIONALES

Bases de datos distribuidas con una solución LAMP (Linux, Apache, MySQL y PHP)

M O N O G R A F Í A

QUE PARA OBTENER EL TÍTULO DE LICENCIADO EN SISTEMAS COMPUTACIONALES

P r e s e n t a

PLSC: Daniel Guzmán Reyes.

ASESOR
M. EN C. GONZALO ALBERTO TORRES SAMPERIO

COASESOR
L.C. ISAÍAS PÉREZ PÉREZ

Pachuca de Soto Hidalgo, 2006
México.

A mis padres:

Gracias por darme todo lo que necesite, gracias por esos consejos que siempre esperaba, por todas las sonrisas que fingieron para no preocuparme, por hacerme creer que la felicidad existe. Ahora ya no pensare más en los recuerdos, que fueron los más hermosos y los más infelices de mí vida, a hora solo se que sí en realidad existe un dios quisiera darle las gracias por concederme la dicha de haberlos conocido, el haberme educado y formado, pero sobre todo de tenerlos.

¿Cómo creen que les pagare todo lo que han hecho por mí?
Padres, siento mucho no poder decirles lo que siento día con día, pero son mí aire y mi motivación, se que el día de mañana que concluya mí vida sabré cual es en realidad el porqué y para qué hicieron tan grande sacrificio hacia mí……...

A mí madre:

Mamá, tú que siempre estas con migo, ahora quiero decirte lo mucho que significas para mí, se que tú corazón es bueno y se preocupa por sus hijos, les das consejos y se preocupa por ellos. Quiero decirte antes que nada, porque eres mí madre, por enseñarme a decir la palabra más linda que hoy me gusta decir con amor y cariño "mamá". Siempre he sabido de tus esfuerzos y sufrimientos. Se cuanto sufres cuando lloro, se que hay ocasiones que no necesitamos tantas palabras para expresar lo que sentimos.

"Gracias mamá, por ser un persona tan maravillosa y fuerte en su momento, gracias por demostrarme cuanto me quieres, pero sobre todo por enseñarme el largo camino y sacrificio que se debe de recorrer para poder lograr las metas que se tienen en la vida, gracias por enseñarme que desde niño se tiene que luchara por alcanzar los objetivos y metas"

A mí padre:

Padre, sabes, no he podido levantarme, solo sé que tengo que salir, luchar contra uno o miles, yo sé que tu tratas de darme lo mejor, tratas de transformarme en un ganador, pero padre, algún día tú pensaste que las causas y razones sobrarían, que mí mente estaba afuera desglosando el vasto mundo, enfrentando problemas, y creciendo como ser humano. A hora quiero decirte que te quiero, se que nunca te lo digo tal vez por que me da pena, pero en silencio y en mí mundo siempre pienso en ti y pienso como estarás, se que en ocasiones soy egoísta por encerrarme en mis cosas, pero eso no quiere decir que no te quiero, a lo mejor no se como demostrarlo, solo se que mientras más te veo más te quiero y te admiro.

"GRACIAS padre, por luchar y hacer todo lo posible para darme lo necesario, gracias por la fortaleza de dejarnos y emigrar a otro país, para darnos lo necesario y cumplir nuestras metras. Ante todo y sobre todo gracias por reprimir tus sentimientos, dejándolos a un lado por dar el apoyo, para lograr el sueño de tus hijos."

A mis hijas:

Gracias por existir en mí vida, gracias por ser el orgullo más grande en mí existencia, el pilar que me da fuerza de seguir adelante. Gracias por llegar en el momento, en que mí vida no tenia sentido.

Agradezco:

Al M. en C. Gonzalo Alberto Torres Samperio y al L.C. Isaías Pérez Pérez, por sus sugerencias, correcciones y ayuda constante durante la redacción de este documento, ya qué, ésta monografía nunca habría llegado a su fin sin los siempre acertados consejos y el constante apoyo recibido por parte de ustedes, pero sobre todo, por su amistad invaluable.

Índice de contenido

Índice de Figuras

Capítulo 1: Conceptos básicos de bases de datos distribuidas

Capítulo 2: Sistemas de bases de datos distribuidas

Capítulo 3: El software libre y la distribución Debian

Capítulo 4: Servidor HTTP Apache

Capítulo 5: Gestor de base de datos MySQL

Capítulo 6: Preprocesador de hipertexto (PHP)

Capítulo 7: Conformación de una base de datos distribuida

Índice de Tablas

Capítulo 1: Conceptos básicos de bases de datos distribuidas

Capítulo 3: El software libre y la distribución Debian

Capítulo 5: Gestor de base de datos MySQL

Capítulo 6: Preprocesador de Hipertexto (PHP)

Capitulo 7: Conformación de una base de datos distribuida

1. Antecedentes

La rápida evolución de los ordenadores ha hecho que su aplicación tenga un proceso de continua adaptación durante los últimos 30 años. Inicialmente, las computadoras surgieron como una herramienta de apoyo, después de esto se fue desarrollando el hardware que permitirá automatizar las tareas de los usuarios. La aparición del software dio una mayor flexibilidad al uso de las computadoras, permitiendo el desarrollo individual de programas que posteriormente se introdujeron en las computadoras, con el objetivo de hacer más rápidas las tareas requeridas por el usuario. Mientras se fue incrementando el desarrollo de hardware y software, tanto en el descenso de los costos como en el aumento de la potencia de cálculos en las computadoras, permitiendo ejecutar mas opciones para agilizar el trabajo requerido por los usuarios o empresas distantes entre ellas, haciendo uso de las redes y bases de datos distribuidas mediante el software libre, el cuál ha ido ganando terreno en el tema de las bases de datos.

Una de las principales preocupaciones de los desarrolladores de software destinado a la administración de bases de datos, es la creciente necesidad de incorporar herramientas que permitan acceder y manipular eficientemente las colecciones de datos que distinguen a las nuevas tecnologías de la información. Donde la mayoría de desarrolladores no definían un conjunto de software especifico que diera grandes ventajas o beneficios para lograr una mayor seguridad de las bases de datos, fue hasta finales del año 2000 cuando los integrantes del equipo de desarrolladores de MySQL David Axmark y Monty Widenius vieron al editor de O'Reilly Dale Dougherty, donde le plantearon un nuevo término “LAMP” para dar solución a las bases de datos distribuidas, haciendo uso del software libre. Por otro lado tal término ya era muy conocido en el país de Alemania, el cuál, definía el trabajo en conjunto del software libre **L**inux, **A**pache, **M**ySQL y uno de los lenguajes; “**P**erl, **P**ython o **P**HP”.

Es frecuente que se identifique a primera vista el mundo del software libre con Linux. Eso provoca que muchas veces se ignoren las herramientas que permiten a Linux convertirse en una gran herramienta de desarrollo de software, especialmente de aplicaciones Web. Existen varios casos en los que un producto pasa de ser una curiosidad a una solución adecuada para la empresa, esto es lo que ha ocurrido con la solución para servicios Web llamada LAMP.

En el presente documento podemos hablar de una solución LAMP ya que esta caracterizada como una de las mejores herramientas disponibles para que cualquier organización o individuo pueda emplear o implementar el diseño de un sistema de base de datos distribuido, ya que implica la toma de decisiones sobre la ubicación de los programas que accederán a la base de datos y sobre los propios datos que constituyen esta última. La ubicación de los programas, a priori, no debería suponer un excesivo problema dado que se puede tener una copia de ellos en cada máquina de la red.

Por otra parte se dice que este conjunto de software es creado por separado, debido a que cada una de las tecnologías que lo forman disponen de una serie de

características comunes las cuáles se comentaran conforme se va desarrollando el documento. Lo más interesante es el hecho de que estos cuatro productos pueden funcionar en una amplia gama de hardware, con requerimientos relativamente pequeños sin perder estabilidad. Esto ha favorecido a que se empleen bases de datos distribuidas haciendo uso de LAMP, siendo una de la alternativa más adecuada para pequeñas y medianas empresas.

2. Justificación

La realización de este documento se genera, debido a que en el presente tiempo no existe una fuente que contenga información básica de las bases de datos distribuidas, usando software de código abierto “Open Source”, cabe mencionar que por otro lado, los libros sobre bases de datos no hablan de qué tipo de software se puede emplear para su construcción, de esta manera limitan el conocimiento de las diferentes herramientas que actualmente compiten con el software propietario, además la información de los libros suelen tener información por separado para conformar un sistema de base de datos distribuido.

Otro punto importante que se debe de mencionar es que la mayoría de los documentos que se pueden consultar acerca del software de código abierto “Open Source”, se encuentra escrito en un lenguaje diferente al español, esto hace que sea una gran limitante para el usuario común que esté interesado en esté tipo de información.

3. Objetivos

3.1. Objetivo general

La presente investigación proporciona información relacionada con los conceptos y aspectos fundamentales de bases de datos distribuidas, gestores de bases de datos y herramientas Open Source, y hacer entender y apreciar que el software Open Source, como Linux, Apache, MySQL, PHP son una herramienta poderosa para la construcción de bases de datos distribuidas, para cualquiera que pretenda dar una solución a cualquier tipo de problema que tengan las empresas o instituciones relacionadas con bases de datos.

3.2. Objetivos específicos

- ✓ Expresar los conceptos básicos de bases de datos distribuidas.
- ✓ Mencionar las características de los sistemas de bases de datos distribuidas.
- ✓ Explicar qué es el software libre, la distribución Debian y los pasos para instalar Debian GNU/Linux.
- ✓ Mostrar la historia y características del servidor HTTP Apache, así como las directivas necesarias para su configuración básica.
- ✓ Dar a conocer las características más importantes del gestor de base de datos MySQL.
- ✓ Proporcionar información acerca del Preprocesador de Hipertexto (PHP).
- ✓ Por último, se redacta como puede ser conformada una base de datos distribuida.

4. Alcances y limitaciones

En la presente monografía sólo se redactan las características principales de una base de datos, bases de datos distribuidas y sistemas gestores de bases de datos, así como componentes Open Source, para la construcción de bases de datos distribuidas, entre las cuáles se encuentran:

- ✓ Sistema operativo Debian GNU/Linux.
- ✓ Servidor HTTP Apache.
- ✓ Gestor de base de datos MySQL.
- ✓ Preprocesador de Hipertexto PHP.

Es importante mencionar que sólo se redactarán los temas necesarios para realizar bases de datos distribuidas; así como el uso del software de código abierto (Open Source).

5. Introducción

Conforme va pasando el tiempo la computadora, los sistemas de telecomunicaciones y las bases de datos, son los medios y herramientas didácticas que utilizan muchas personas de diferente nivel de estudio y organizaciones, de tal forma que en la actualidad las bases de datos distribuidas y sus componentes notan una gran importancia en las empresas e instituciones; por lo tanto requieren una mayor seguridad en el manejo y el guardado de su información que es de vital importancia.

Aprovechando que el sistema operativo Debian GNU/Linux es de código libre y proporciona mayor seguridad que otros sistemas operativos, se hará uso de este sistema debido a que proporciona varias ventajas como sistema completo con características multiusuario, multitarea, implementación completa del protocolo TCP/IP, tiene un sistema completo de correo electrónico, etc. Por tal razón en el presente escrito, sólo se hablara del software de código abierto, así como software que se pueda adquirir sin tener que pagar cantidades monetarias altas, para lograr la implementación, diseño y construcción de bases de datos distribuidas, seguras y de bajo costo.

Otro de los puntos importantes que se realizan son las características del Servidor HTTP Apache, posteriormente se hablará del sistema gestor de base de datos MySQL, sistema gestor que en la actualidad es muy usado para el manejo y uso de las mejores bases de datos; así como la facilidad de descargarlo desde Internet. Más adelante se mostrarán las ventajas y características del Preprocesador de Hipertexto (PHP), como interfaz entre la base de datos y el usuario que desee hacer consultas de la misma.

5.1. El porqué se elige este software

Para iniciar con esta descripción del porqué se decide usar este tipo de software para implementación o uso de bases de datos distribuidas. Primero, es por que este software está basado en la licencia GPL. El segundo punto, se debe de mencionar el significado del término LAMP. LAMP es un término utilizado para definir cómo MySQL puede ser utilizado en conjunción con Linux, Apache y cualquiera de los lenguajes de scripts cuyo nombre empieza por 'P': Perl, Python y PHP.

LAMP = Linux + Apache + MySQL + (PHP, Perl, Python).

En general, el término LAMP, hace referencia a cualquier combinación de herramientas open source para desarrollo Web e interfaz de bases de datos como:

- ✓ Linux podría reemplazarse por OpenBSD.
- ✓ MySQL por PostGreSQL.
- ✓ La 'P' podría traducirse en PHP, Perl, Python o Ruby

Porqué la distribución Debian: Al realizar un exhaustivo estudio de las distribuciones para identificar que distribución era mejor para implementarla en este

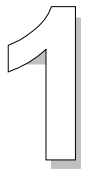
trabajo, se concluye que la distribución Debian se elige por ser una distribución 100% libre.

Porqué Apache: Básicamente con el servidor Apache HTTP o Apache no hay ningún problema, debido a que esté forzosamente debe de ser instalado.

Porqué MySQL y no PostGreSQL: MySQL esta basado bajo la licencia GPL y PostGreSQL, tiene la licencia BSD de tipo simple y permisible sin copyleft con el defecto de la cláusula publicitaria de BSD que ocasiona problemas prácticos.

Porqué PHP y no otro lenguaje: PHP es más sencillo para gente nueva en la escritura de scripts, es fácilmente extensible, entre otras; Perl, Python son de propósito general, donde se pueden hacer cosas como en C, Java, Visual Basic, etc.

Capítulo



Conceptos básicos de bases de datos distribuidas

El crecimiento de la globalización y las situaciones más competitivo ha hecho que las empresas nacionales e internacionales trabajen de una nueva manera, con el objetivo de maximizar sus asociaciones entre diferentes unidades de negocios, ingeniería y proyectos alrededor del mundo. Con la explosiva popularidad de la Internet y el World Wide Web (WWW) hay una necesidad de crecimiento rápido para suministrar acceso sin precedente a bases de datos distribuidas globalmente a través de la Internet. La integración de los datos dispersos en diferentes sitios para ser accedidos a través de la Web, requieren de nuevas arquitecturas y herramientas de software para el desarrollo de estos sistemas. Esta necesidad ha creado una fuerte demanda por capacidades de acceso a bases de datos a través de Internet.

En este capítulo se muestra la definición de una base de datos, las ventajas y desventajas de éstas, así como una breve explicación de las características de las bases de datos distribuidas como la arquitectura, diseño, consultas, transacciones, concurrencia, entre otras.

1.1. Origen de las bases de datos

Desde el principio de la humanidad se ha tenido una gran importancia de organizar y almacenar grandes cantidades de información para después recuperarla en forma eficaz, por tal motivo se crearon métodos para la organización de información.

Métodos antiguos para la organización de datos:

- ✓ ***Pergaminos:*** Los datos se escribían en una larga hoja de papel que se podía enrollar. La desventaja evidente es que los datos que aparecen al final del rollo no pueden recuperarse a menos que se desenrolle todo el pliego. Sin embargo era un método de archivo compacto para grandes cantidades de datos.

- ✓ ***Libros encuadernados:*** La rapidez de acceso puede mejorarse si el rollo se divide en páginas y luego estas se encuadernan formando un libro. Esta innovación permite un rápido acceso a los datos almacenados en cada página. Los datos en cada página se buscan ordenadamente. **[1]**

Esta situación permitió acumular gran cantidad de información que posteriormente fue transmitida de una generación a otra, a pesar de no contar en esa época con los mejores medios de almacenamiento.

Actualmente con los adelantos tecnológicos en el campo de la comunicación y la industria de la información se ha hecho posible que todos los conocimientos publicados puedan ser difundidos por diferentes medios entre los que podemos citar libros, publicaciones periódicas, revistas y bases de datos. **[2]**

Una Base de datos, es una colección de datos organizados por su historia, estructura y acceso a la computadora. Los datos pueden ser textos, números, imágenes, etc. El gobierno, el ejército, las empresas e instituciones tienen bases de datos donde registran su información.

Las bases de datos comenzaron a comercializarse desde 1960, pero su disponibilidad fue limitada. Fue entonces cuando los desarrolladores de los códigos de bases de datos decidieron que estos deberían estas disponibles (por primera vez en 1970). Años anteriores de la aparición de las técnicas de la base de datos, ya existía un elevado nivel de redundancia en los sistemas de procesamiento de datos. Aún con las técnicas de las bases de datos a medida que crecientes volúmenes de información se combinan para formar bases de datos integradas, se crea una mayor posibilidad de redundancia. **[3]**

El manejo de los datos es el uso de procedimientos computarizados para la captura de los datos, almacenamiento y recuperación, después de esto los suprime o los archivan en un sistema de almacenamiento. De esta forma el manejo de los datos es una manera más técnica, para el procesamiento de información.

1.2. Definición de base de datos (BD)

"*Colección o depósito de datos, donde los datos están lógicamente relacionados entre si, tienen una definición, una descripción común y están estructurados de una forma particular. Una base de datos es también un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones*".

Una base de datos es una colección y conjunto de datos almacenados en un servidor informático. Los datos o información están estructurados e interrelacionados entre sí basados en un modelo preparado para guardar el máximo contenido semántico. También cabe mencionar que la redundancia que existe en la base de datos debe ser controlada, para evitar duplicidad de información innecesaria y que las redundancias físicas sean tratadas por el sistema para no ocasionar inconsistencia en la misma.

Las bases de datos tienen como objetivo ayudar a la organización en la que se encuentren implementadas, manipulando los datos como recurso. Por tal motivo, las bases de datos deben atender a distintos usuarios y aplicaciones al mismo tiempo. Uno de los puntos importante de las bases de datos es la independencia física como la lógica, entre datos, para hacer mas optimo el manejo de la información. **[A]**

1.3. Beneficios e inconvenientes de usar bases de datos

Las bases de datos, surgen como respuesta al nuevo planteamiento de los sistemas orientados hacia los datos, mejorando las prestaciones de los sistemas informáticos e incrementando su rendimiento. En la presente redacción se hace mención de las principales características del uso de una base de datos, como son las ventajas y desventajas del uso de estas. Cabe mencionar que estas características dependen también del sistema gestor de base de datos que se este usando.

1.3.1. Beneficios

Independencia de los datos respecto a los tratamientos y viceversa: Esta independencia lleva a que los cambios (inserción, borrado, actualización) de los datos no impongan un nuevo diseño lógico o físico de la base de datos, dando una flexibilidad; la cuál es muy importante para concebir sin excesivo costo la continua adaptación de la base de datos, con respecto a la organización.

Coherencia de los resultados: Debido a que la información de la base de datos es agregada y guardada, una sola vez, ocasionando que las consultas accedan a los mismos datos; por lo que, los resultados de todas las consultas deben ser coherentes y compatibles.

Mejor disponibilidad de los datos para el conjunto de usuarios: Aquí los usuarios ya no son propietarios de los datos, debido a que se comparten entre el conjunto de aplicaciones, dando como resultado la disponibilidad de los datos,

siempre y cuando se cuente con los permisos necesarios. También existe mayor transparencia de la información, ya que los datos se encuentran relacionados en un diccionario de datos.

Mayor valor informativo: Gracias a que la base de datos es un reflejo del mundo real, donde todos los elementos se encuentran interrelacionados, el valor de su conjunto es superior a la suma del valor informativo de los elementos separados que lo forman.

Mejor y más normalizada documentación de la información, la cuál esta integrada con los datos: Aquí la estandarización no existe, debido a que los datos se encuentran separados de su contenido semántico, los primeros se almacenan en ficheros y su descripción se hace mediante un lenguaje de programación. La documentación de los datos, realizada por el programador, es en general insuficiente y en ocasiones no existe.

Mayor eficiencia en la recopilación, validación y entrada de los datos al sistema: Los datos son guardados y validados una sola vez; incrementando el rendimiento de todo el proceso previo al almacenamiento.

Reducción del espacio de almacenamiento: La disminución o desaparición de la redundancia; así como la aparición de técnicas de comparación, dan como resultado que los sistemas de bases de datos usen menos espacio de almacenamiento en disco.

Ventajas de las bases de datos, referidos a datos, resultados y usuarios:
Los datos:

- ✓ Independencia de los datos respecto a los tratamientos y viceversa.
- ✓ Mejor disponibilidad de los datos para el conjunto de usuarios.
- ✓ Mayor eficiencia en la recopilación, validación y entrada de datos al sistema.

Los resultados:

- ✓ Mayor coherencia.
- ✓ Mayor valor informativo.
- ✓ Mejor y más normalizada documentación de la información.

Los usuarios:

- ✓ Acceso más rápido y sencillo de los usuarios finales.
- ✓ Más facilidad de compartir los datos por el conjunto de los usuarios.
- ✓ Mayor flexibilidad para atender a demandas cambiantes. **[B]**

1.3.2. Inconvenientes

Instalación costosa: La instalación de la base de datos puede ser costosa, en equipo físico (nuevas instalaciones o ampliación de estas) como en el lógico (sistema operativo, programas, compiladores, etc.).

Personal especializado: Los conocimientos, que son imprescindibles para un uso adecuado y eficaz de la administración de la base de datos, originan una contratación y formación de nuevo personal.

Implantación larga y difícil: La implantación de la base de datos puede ser tardada y laboriosa, sí no se cuenta con el personal indicado.

Falta de rentabilidad a corto plazo: La implantación de un sistema de base de datos, tanto en costo de personal y equipo, como el tiempo que tarda en estar funcionando, no resulta favorable a corto plazo. Para que el sistema sea rentable debe funcionar a mediano o largo plazo.

Ausencia real de normas: No existe una estandarización, que facilite a los usuarios el manejo de los sistemas de bases de datos.

Desfase entre teoría y práctica: En ocasiones los usuarios, especialmente los directivos, se engañan respecto a las prestaciones reales que pueden proporcionar los sistemas gestores de bases de datos actuales, creyendo que ya son una realidad ciertos aspectos que aun son teóricos, dando origen, riesgos de frustración en los usuarios de los sistemas de bases de datos, que pueden hacer olvidar las ventajas que ofrecen estos sistemas.

Desventajas de las bases de datos, con respecto a implantación y usuarios:
Relativas a la implantación:

- ✓ Costo en equipo.
- ✓ Ausencia de estándares.
- ✓ Larga y difícil implantación.
- ✓ Rentabilidad a mediano plazo.

Relativa a los usuarios:

- ✓ Personal especializado.
- ✓ Desfase entre teoría y práctica. **[B]**

1.4 Modelos de bases de datos

Según Flory (1982), "modelar consiste en definir un mundo abstracto y teórico, tal que las conclusiones que se puedan sacar de él, coinciden con las manifestaciones aparentes del mundo real". Siendo un modelo, "un conjunto de conceptos que permiten construir una representación organizacional de la empresa".

Según Tsichritzis y Lochovsky (1982), "un modelo de datos es un dispositivo de abstracción que nos permite ver el bosque (información contenida en los datos) en opción a los árboles (valores individuales de los datos)"

Modelo entidad/interrelación (ME/R): Este modelo E/R fue propuesto por Meter P. Chen en sus dos artículos, Chen (1976) y Chen (1977). Según Chen (1976), "el modelo E/R, puede ser usado como una base para una vista unificada de los datos",

adoptando “el enfoque mas natural del mundo real, que consiste en entidades e interrelaciones”.

El modelo E/R, esta basado en dos conceptos fundamentales: el de entidades y el de interrelación. Donde la entidad es cualquier objeto real o abstracto sobre el cuál se quiere tener información en la base de datos y que tiene existencia por si mismo. La interrelación, es la asociación o correspondencia entre entidades.

Modelo entidad/interrelación extendido: La existencia de este modelo se origina, debido a que varios autores han considerado diversas extensiones del modelo E/R definido por Chen, dando lugar a los que algunos denominan E/R extendido (EE/R). Este modelo, define el contenido semántica de las interrelaciones, como:

- ✓ Cardinalidad: Número máximo y mínimo de ocurrencia de un tipo de entidades que puede estar interrelacionada con otra ocurrencia.
- ✓ Dependencia en existencia: Cuando las ocurrencias de un tipo de entidades no pueden existir sin desaparecer la ocurrencia de la entidad.
- ✓ Abstracción.

Modelo en red general y sistema Codasyl: Representa la entidad en forma de nodos de un grafo y las asociaciones o interrelaciones entre estas, mediante arcos que unen a los nodos. Esta representación no impone ninguna restricción al tipo, número de arco, permite modelar estructuras de datos muy complejas.

Modelo Codasyl: Definición de datos a nivel lógico: Modelo encargado de mostrar las diferentes sentencias del lenguaje de definición de datos a nivel lógico.

- ✓ Estructura general del lenguaje de definición (LDDE): Encargado de realizar la descripción de todos los elementos que forman parte de la base de datos (áreas, registros, esquemas y conjuntos).
- ✓ Entrada de esquema: Especifica el nombre con el que se va a conocer el esquema y todas las características relativas a la seguridad y confidencialidad de la base de datos.
- ✓ Entrada de área: Las áreas pueden ser páginas de disco, cilindros, etc.; es decir, parte del soporte físico que tiene el sistema. Por lo tanto los elementos que se asignan a cada área de la base de datos son en base a criterios de rendimiento.
- ✓ Entrada de registro: Es usado para dar nombre a cada uno de los tipos de registros existentes en la base de datos y especificar sus elementos de datos; así como, otras características importantes del modelo.
- ✓ Subentrada de datos: Se utiliza para describir las características de los electos de datos que ha de contener el registro.
- ✓ Entrada de conjunto: Esta entrada especifica los registros que forman parte del conjunto, el nombre que recibe este y otras características.

Modelo Codasyl: Dinámica: Explica la parte dinámica del modelo; es decir, la manipulación de datos en dicho modelo.

- ✓ Selección: Consiste en localizar un registro de la base de datos.
- ✓ Acción: Con el registro seleccionado se puede hacer acciones, como actualizar, insertar, borrar, modificar, etc.

Modelo Codasyl: Esquema de almacenamiento: Este esquema es usado para proporcionar:

- ✓ Independencia del esquema respecto a los aspectos físicos de la base de datos.
- ✓ Independencia del sistema operativo.
- ✓ Eficiencia en los accesos y en el almacenamiento de los datos.
- ✓ Posibilidad de reestructurar la base de datos.

Modelo jerárquico como un caso particular de los modelos en red: Este caso surge, debido a que los árboles como instrumento para la representación de estructuras de datos, presentan problemas con su poca flexibilidad, lo que da origen a una falta de adaptación a muchas organizaciones reales. Por lo tanto el modelo jerárquico se puede definir como:

- ✓ Un conjunto de tipos de entidades (segmentos, grupos repetitivos, registros, etc.) $E_1, E_2, \ldots\ldots, E_n$ (nodos de grafo).
- ✓ Un conjunto de interrelaciones, R_{ij} que une las entidades E_i y E_j (arcos del grafo).
- ✓ Un conjunto de relaciones inherentes que provienen de la estructura jerárquica, que ya han sido mencionadas anteriormente.

Modelo relacional: Estática: Según las palabras de E. F. Codd (1970), "la vista relacional de los datos… parece ser superior al modelo en grafos o en red… Proporciona un medio de describir datos con su estructura natural únicamente; es decir, sin superponer ninguna estructura adicional con el propósito de su representación en la maquina".

El trabajo publicado por E. F. Codd en ACM, Codd (1970), presentaba un modelo de datos que percibe los objetivos:

- ✓ Independencia lógica: Al anexar, eliminar o modificar objetos de la base de datos no repercute en los programas y usuarios.
- ✓ Flexibilidad: Facilidad de presentar los datos a cada usuario en la forma que prefiera.
- ✓ Independencia física: El modo en que se almacenan los datos no influye en su manipulación lógica.
- ✓ Uniformidad: La estructura lógica de los datos presenta un aspecto uniforme, facilitando la concepción y manipulación de la base de datos.
- ✓ Sencillez: Facilidad de comprender y utilizar el modelo de datos relacional. **[B]**

1.5. Bases de datos distribuidas (BDD)

El crecimiento de las bases de datos distribuidas es ocasionado por razones organizacionales, las cuales necesitan grandes capacidades para que los datos sean anexados a las bases de datos, como la integración de información desde diferentes lugares donde se encuentra la organización distribuida, y así poder hacer una consulta desde distintas localidades o lugares.

Debido a que las bases de datos distribuidas constan de una colección de nodos o sitios y cada uno de ellos representa una computadora; además con el desarrollo de las tecnologías de comunicación permiten que se pueda tolerar y vincular datos con aplicaciones que se hallan en zonas distintas y remotas; por ejemplo las transacciones bancarias realizadas en cajeros que se localizan en centros comerciales, empresas, escuelas, etc. **[C]**

1.5.1. Definición de base de datos distribuida

Una base de datos distribuida***, se refiere a un conjunto de nodos o localidades, en cada uno de estos nodos tienen una base de datos, por tal motivo cada nodo procesa transacciones locales; así cada nodo puede participar en la ejecución de transacciones globales***.

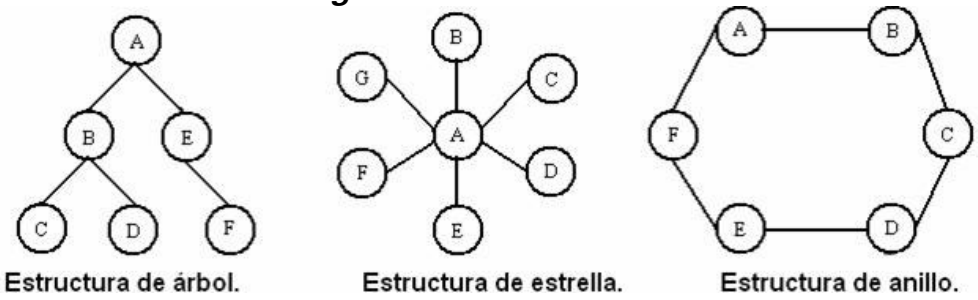


Figura 1.1: Comunicación de nodos. **[D]**

La manipulación de transacciones globales necesitan estar conectadas entre sí a través de distintos elementos de comunicación; como cables de red de alta velocidad usando Internet y líneas telefónicas, estas no comparten memoria RAM ni la sincronización de los reloj de cada computadora donde se encuentran las bases de datos.

Los nodos pueden comunicarse físicamente de muchas formas, estas configuraciones se pueden representar usando grafos (ver Figura 1.1). Las diferencias entre las configuraciones que puedan implementarse en las localidades son las siguientes:

- ✓ Costo de implementación: Costo de conexión física entre localidades.
- ✓ Costo de comunicación*:* Costo de tiempo y dinero en enviar información entre los nodos.
- ✓ Fiabilidad: Periodicidad con la que falla la comunicación entre nodos.

- ✓ Disponibilidad: Acceso a la información aunque existan fallas entre los nodos. **[D]**

1.5.2. Características de las BDD

Autonomía local: Las operaciones en un nodo dado están controladas por este nodo; así como los datos locales son obtenidos y administrados localmente, todos los datos pertenecen "realmente" a una base de datos local, aun cuando estén accesibles desde otro nodo distante; por lo tanto la seguridad, integridad y la representación de almacenamiento de los datos locales están bajo el control y jurisdicción del nodo local.

No dependencia de un nodo central: Los nodos deben ser tratados de mera igual, para que no exista alguna dependencia de un nodo "maestro" central para un servicio central. Cabe mencionar que este nodo central puede ser un cuello de botella, ocasionando que el nodo sea vulnerable y falle el sistema.

Operación continua: Da mayor confiabilidad y disponibilidad, ya que un sistema de base de datos nunca debe de estar fuera de servicio, porqué debe de soportar backups en línea; así que debe de tener un soporte para que pueda recuperar rápidamente la base de datos cuando exista una falla.

Independencia de ubicación: Los usuarios no deben de conocer el lugar donde son guardados los datos, demostrando como si los datos estuvieran almacenados en el lugar donde se encuentran los usuarios, esta independencia es importante para simplificar los programas de usuario y las actividades terminales.

Independencia de fragmentación: Los datos pueden ser almacenados en el lugar donde son usados con mayor frecuencia, para que la mayoría de las operaciones sean locales y se reduzca el tráfico en la red.

Independencia de replicación: Un fragmento puede ser representado por muchas copias distintas o replicas, guardados en lugares distintos.

Procesamiento de consulta distribuida: La conexión de la consulta debe ser independiente del nodo o sitio donde se realice la consulta.

Administración de transacciones distribuidas: En este punto sobresale el control de la recuperación y el control de la concurrencia, para hacer las actualizaciones en muchos lugares, el control de la concurrencia esta basado en el ***bloqueo.*** El control de la recuperación, asegura que una transacción sea atómica (todo o nada) en el ambiente distribuido.

Independencia del hardware: Facilidad de ejecutar el mismo sistema gestor de base de datos en las diferentes plataformas de hardware y hacer que participen de forma igual.

Independencia del sistema operativo: Facilita que el sistema gestor de base de datos se ejecute en diferentes sistemas operativos (Unix, NT).

Independencia de la red: Garantiza que el sistema distribuido soporte varios tipos de redes de comunicación.

Independencia del sistema gestor de base de datos: Todos los sistemas gestores de bases de datos soportan la misma interfaz en diferentes nodos. **[E]**

1.5.3. Ventajas de las BDD

Una de las ventajas más importantes de las bases de datos distribuidas es la capacidad de compartir y acceder a la información de una forma segura y eficaz.

Uso compartido de los datos y distribución del control: Si existen varias localidades distintas interconectadas entre sí; en distintos lugares, esto origina que algún usuario ajeno a cierta localidad pueda acceder a datos que este en otra localidad, siempre y cuando esta este disponible para el usuario.

La ventaja principal de compartir datos a través de la distribución, es que las localidades pueden controlar cierto número de datos almacenados en la misma localidad. En un sistema centralizado, el administrador de bases de datos de la localidad central controla la base de datos. En un sistema distribuido coexiste un administrador global de la base de datos que se encarga de todo el sistema. Parte de responsabilidad es asignada a los administradores de cada base de datos.

Confiabilidad y disponibilidad: Cuando existe una falla en alguna de las localidades el sistema de base de datos distribuido sigue funcionando. Si los datos se llegan a repetir en cierta localidad, esto no impide que el usuario pueda realizar una consulta en la base de datos, la podrá hacer sin ningún problema. De esta manera cuando ocurre una falla en un nodo no es necesario desconectar el sistema.

El sistema debe ser capaz de detectar fallas en las localidades y debe de evaluarlas para recuperarse de la falla. El sistema debe desconectar la localidad que fallo para no seguir utilizándola. Después de esto; cuando se recupere la localidad, conviene contar con dispositivos para restablecerla al sistema con los mínimos problemas.

Sin embargo la recuperación de fallas en sistemas distribuidos es más compleja que en los centralizados, la ventaja que posee el sistema de continuar trabajando, a pesar de la falla de una de las localidad se deriva una mayor disponibilidad. Consiguientemente la disponibilidad es imprescindible para los sistemas de bases de datos que manejan aplicaciones en tiempo real.

Asignación del proceso de consultas: Cuando una consulta contiene datos de distintas localidades; es posible dividir la consulta en diversas subconsultas que se hagan en paralelo en diferentes localidades.

1.5.4. Inconvenientes de las BDD

La desventaja de los sistemas de bases de datos distribuidos, poseen una complejidad mayor que es necesaria para avalar una correcta coordinación entre las localidades que forman el sistema distribuido.

Costo de desarrollo de software: El costo se deriva de la dificultad para constituir un sistema de base de datos distribuido, por esta razón su costo es elevado.

Mayor posibilidad de errores: Esto se origina por que las localidades del sistema distribuido funcionan en paralelo; ocasionando que los algoritmos no sean correctos en su totalidad, esto da como resultado errores muy sutiles.

Mayor tiempo de procesamiento: El intercambio de datos y cálculos que se necesitan para poder coordinar los procesos de las diferentes localidades ocasionan mayor tiempo de procesamiento.

Cuando el diseñador de bases de datos distribuidas escoge o realiza el diseño debe tener presente las ventajas y desventajas de la distribución de los datos, para tener mejor seguridad en el manejo de estos. **[D]**

1.6. Arquitectura de las BDD

Dicha arquitectura, se forma mediante las bases de datos que conforman la base de datos distribuida, el sistema gestor de bases de datos y los programas de consulta, como se muestra en la figura 1.2. Cabe mencionar que el procesamiento en las bases de datos distribuidas, como la ejecución de las transacciones, la recuperación y actualización de los datos se realiza entre dos ó más computadoras que se encuentran en diferente lugar. **[4]**

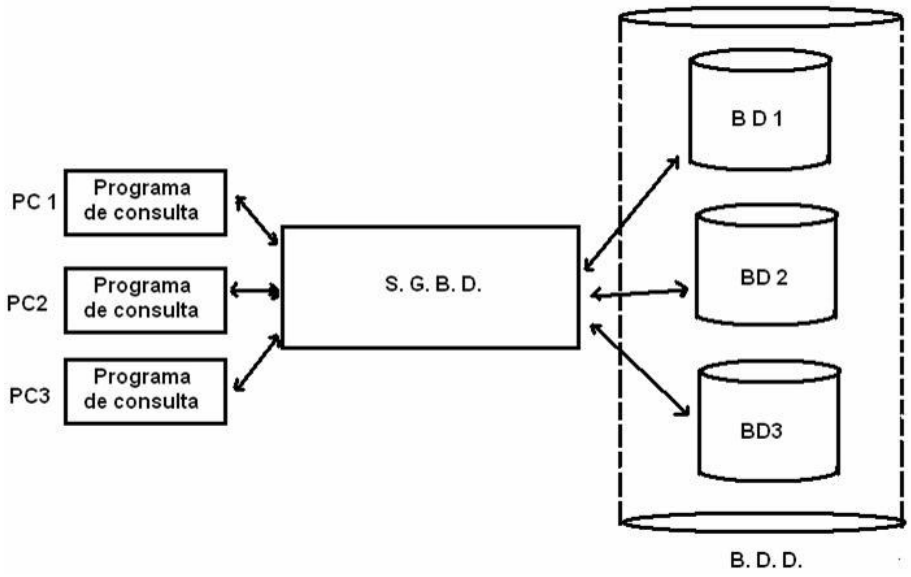


Figura 1.2: Arquitectura de las bases de datos distribuidas. **[4]**

1.7. Diseño de las BDD

Cuando se inicia con el diseño de una base de datos distribuida se debe de tener en cuenta el problema de cómo va a ser distribuida la información en los diferentes lugares, como la fragmentación y la asignación de fragmentos en los diferentes lugares de la red, para que no exista diferentes copias de la misma información. Otro de los puntos importantes es el uso de directorios globales, locales o en dado caso la combinación de ambos, esto dependerá del tipo de usuarios que existan.

1.7.1. Estrategias

Para que un diseño de base de datos distribuida se logre de manera correcta se debe hacer uso de las estrategias top-down y bottom-up; las cuales garantizan un mejor funcionamiento, debido a que se necesitan uno al otro.

1.7.1.1. Top-down

Proceso que inicia con el análisis de requerimientos, para definir el diseño conceptual, las vistas de usuario, posteriormente se define un esquema conceptual global y los esquemas externos necesarios, se continúa con la fragmentación de la base de datos e inmediatamente se asignan los fragmentos en los distintos sitios del sistema, creando las imágenes físicas. Para finalizar se elabora un diseño físico para cada lugar (ver Figura1. 3).

1.7.1.2. Bottom-up

Por lo regular Bottom-up es usada para unir muchas bases de datos centralizadas, esto se logra a partir de varios esquemas conceptuales para hacer un solo esquema global.

Etapas del proceso descendentes:

Análisis de requerimientos: Aquí se fijan los requisitos para obtener los datos, como el procesamiento de los usuarios; así como establecer los requisitos del sistema, rendimiento, seguridad, disponibilidad y flexibilidad con respecto a factores económicos. Cuando se termina con esta etapa se habrán cumplido los requisitos para poner en marcha el diseño conceptual y diseño de vistas.

Diseño de vistas: En esta etapa se definen las aplicaciones que serán empleadas para usar la base de datos, como datos estáticos de acceso a las aplicaciones de las tablas que faciliten tener información para optimizar y crear un diseño conceptual eficiente. En otras palabras, aquí se define la interfaz entre usuario y sistema.

Diseño conceptual: Etapa encargada de la unión de las vistas de usuario, logrado mediante un esquema global, información de acceso y esquemas externos.

Distribución del diseño: En esta etapa se obtienen varios esquemas conceptuales mediante el esquema conceptual global y la información de acceso. Aquí se debe de tomar en cuenta la fragmentación y la asignación.

Diseño físico: Este diseño se logra mediante esquemas conceptuales locales y la información de acceso reunidos en las etapas anteriores.

Monitorización y observaciones o ajustes: Usado para tener un control de proceso, con el objetivo de reparar los errores que vallan apareciendo.

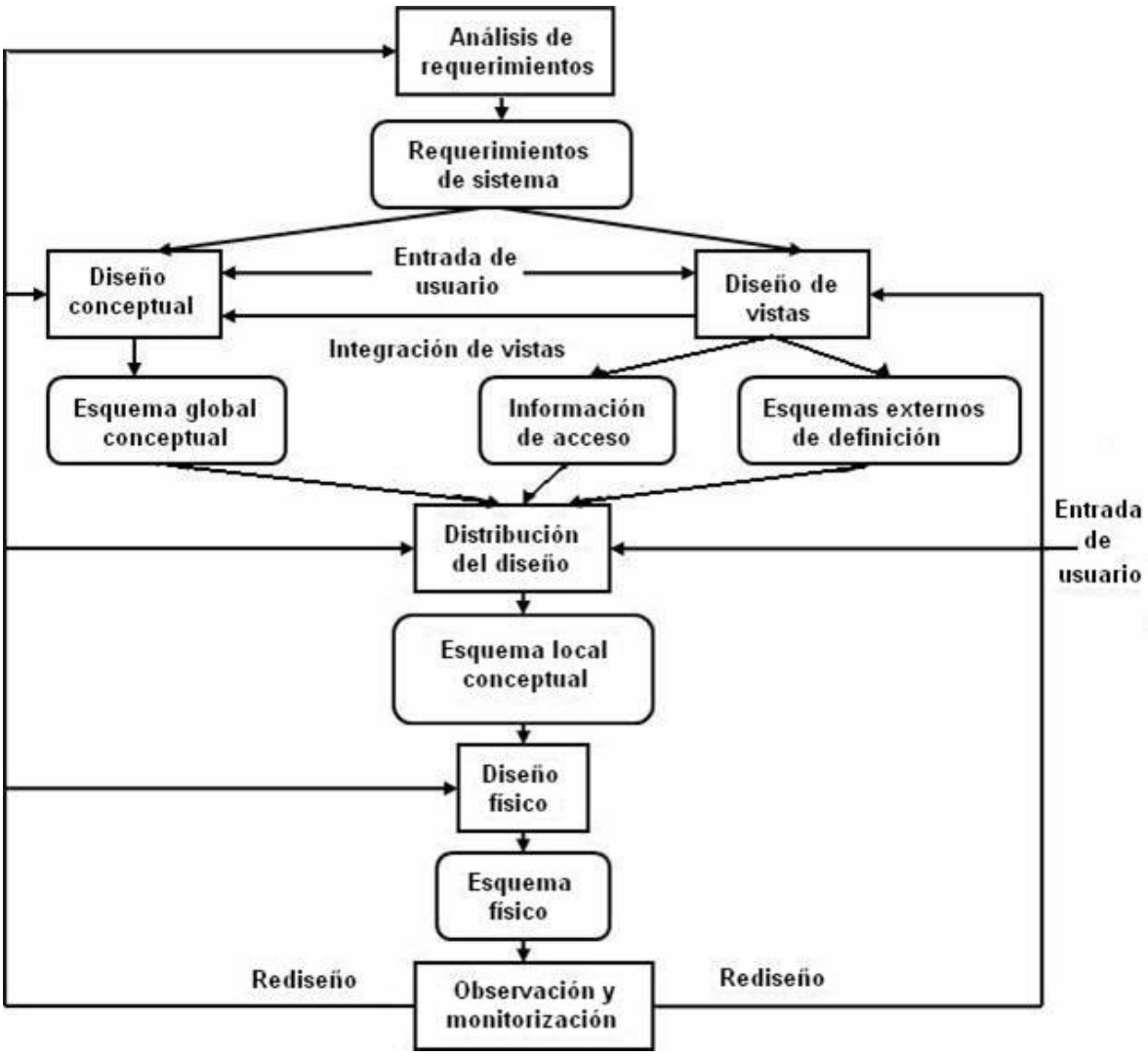


Figura 1.3: Proceso del diseño top-down. **[F]**

1.7.2. Fragmentación

El proceso de fragmentación consiste en dividir la base de datos en varias partes guardadas en distintos lugares. Para lograr este proceso se cuenta con la fragmentación horizontal, vertical e hibrida.

1.7.2.1. Razones e inconvenientes de fragmentación

Razones para fragmentar:

- ✓ Eficiencia: Los datos son guardados en el lugar donde serán usados, de esta manera no existe duplicación de datos.
- ✓ Paralelismo: División de transacciones en subconsultas que funcionan con fragmentos.
- ✓ Seguridad. Los datos que no son ocupados no se guardan localmente, evitando que sean usados por los usuarios sin permisos.

Inconvenientes de fragmentar:

- ✓ Las vistas que tengan mas de un fragmento pueden tener bajo redimiendo.
- ✓ Las vista de usuario no se pueden realizar con un solo fragmento, esto ocasiona un rendimiento lento.
- ✓ Complejidad de mantenimiento de la integridad referencial.

1.7.2.2. Corrección en la fragmentación

Las siguientes reglas se tienen que cumplir para que no ocurran cambios semánticos en la base de datos.

Integridad: Si una relación R se descompone en una serie de fragmentos R_1, R_2…, R_n elemento de datos que pueda encontrarse en R, se deben de localizar en uno o más fragmentos R_i. Esta propiedad garantiza que los datos de la relación global sean proyectados sobre los fragmentos sin pérdida alguna.

Corrección: Si una relación R se descompone en una serie de fragmentos R_1, R_2…, R_n, puede definirse un operador relacional ∇ tal que:

$$R = \nabla R_i, \forall R_i, \in F_R$$

El operador ∇debe ser distinto dependiendo de la fragmentación que se aplique.

Disyunción: Esta regla garantiza que los fragmentos horizontales sean disjuntos. Si una relación R se descompone horizontalmente en varios fragmentos R_1, R_2…, R_n y un elemento de datos se encuentra en algún fragmento R_j entonces no se localiza en otro fragmento R_k ($K \neq j$). **[F]**

1.7.2.3. Tipos de fragmentación

Debido a que anteriormente se explico que es la fragmentación, a hora se continúa con la explicación de cada tipo de fragmentación, haciendo uso del siguiente ejemplo:

Las entidades a ser modeladas son ingenieros y proyectos. Para cada ingeniero, se quiere conocer su número de empleado (ENO), su nombre (ENOMBRE), puesto ocupado dentro de la compañía (TITULO), el salario (SAL), la identificación de los

nombres se los proyectos en los cuáles están trabajando (JNO),la responsabilidad que tienen dentro del proyecto (RESP), y la duración de su responsabilidad en meses (DUR). Igualmente para cada proyecto se quiere conocer el número de proyecto (JNO), el nombre del proyecto (JNOMBRE), el presupuesto asignado al proyecto (PRESUPUESTO) y el lugar donde se desarrolla el proyecto (LUGAR) (ver Figura 1.4).

E

ENO	ENOMBRE	TITULO
E1	J. Doe	Ingeniero Eléctrico
E2	M. Smith	Analista de Sistemas
E3	A. Lee	Ingeniero Mecánico
E4	J. Miller	Programador
E5	B. Casev	Analista de Sistemas
E6	L. Chu	Ingeniero Eléctrico
E7	R. Dávis	Ingeniero Mecánico
E8	J. Jones	Analista de Sistemas

J

JNO	JNOMBRE	PRESUPUESTO
J1	Instrumentación	150000
J2	Desarrollo de bases de datos	135000
J3	CAD/CAM	250000
J4	Mantenimiento	310000

S

TITULO	SALARIO
Ingeniero Eléctrico	40000
Analista de Sistemas	34000
Ingeniero Mecánico	27000
Programador	24000

G

ENO	JNO	PUESTO	DUR
E1	J1	Administrador	12
E2	J1	Analista	24
E2	J2	Analista	6
E3	J3	Consultor	10
E3	J4	Ingeniero	48
E4	J2	Programador	18
E5	J2	Administrador	24
E6	J4	Administrador	48
E7	J3	Ingeniero	36
E8	J3	Administrador	40

Figura 1.4: Tablas de la base de datos de una compañía. **[F]**

Cabe decir que un ingeniero puede participar en más de un proyecto pero su salario corresponde únicamente al puesto que ocupa en la compañía. Después de aplicar la normalización se obtienen las relaciones *E* para ingeniero, *J* para proyectos, *S* para los salarios asignados a los puestos y *G* para los ingenieros asignados a cada proyecto (ver Figura 1.5). **[F]**

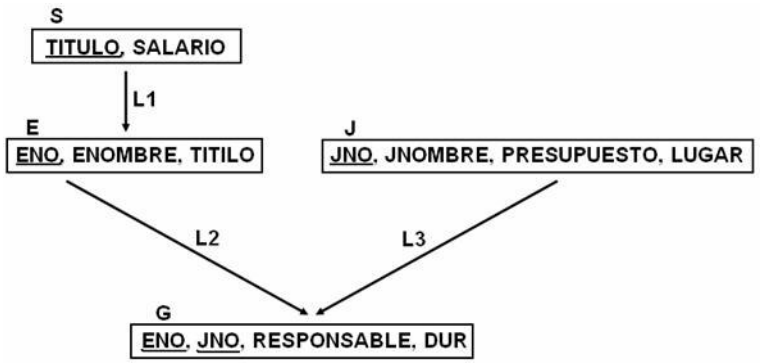


Figura 1.5: Modelo conceptual de la base de datos de la figura 1.4. **[F]**

1.7.2.3.1. Horizontal

Este tipo de fragmentación consiste en dividir una relación global en subconjuntos, donde cada subconjunto puede contener datos que satisfagan una condición y se puedan definir expresando cada fragmento como una operación de selección sobre la relación global. En otras palabras se dice que para construir una fragmentación se necesitara *información de la base de datos* y las *aplicaciones* que se utilizan. La información consiste en contar con las relaciones que conforman la base de datos, la cardinalidad de cada relación card(R), las dependencias entre relaciones e información cualitativa y cuantitativa, donde la información cualitativa se encargara de guiar la fragmentación, mientras que la cuantitativa se necesitará en los modelos de asignación. La principal información de carácter cualitativo son los predicados empleados en las consultas de usuario.

Una combinación especialmente interesante es la conjunción de predicados simples, al predicado resultante se le denomina predicado minitérmino. Partiendo de que siempre es posible transformar una expresión lógica en su forma normal conjuntiva.

Sobre la información cuantitativa necesaria para realizar las aplicaciones, será necesarios definir los siguientes conjuntos de datos:

Selectividad minitérmino: Es el número de tuplas de una relación a las que accede una consulta de acuerdo a un predicado minitérmino dado. Se especificara la selectividad de un minitérmino m_i como sel(m_i).

Frecuencia de acceso: Frecuencia con la que un usuario accede a los datos. Si Q = {q1, q2, ..., qn} es un conjunto de consultas de usuario, acc(qi) indica la frecuencia de acceso a la consulta q_i en un periodo dado.

Dada una relación *R(A1, A2, …, An)*, donde *Ai* es un atributo definido sobre el dominio *Di*, un predicado simple *pj* definido en *R* tiene la forma:

$$p_j: A_i \ \theta \ \text{Valor}$$

Donde $\theta \in \{ =, <, \neq, \leq, >, \geq \}$ y $Valor \in D_i$. Para la relación *R* se define un conjunto de predicados simples como $Pr = \{ p_1, p_2, \ldots, p_m \}$.

Ejemplo de predicado simple:

JNOMBRE = "Mantenimiento"
PRESUPUESO < 200000

Dado la relación *R* y el conjunto de predicados simples $Pr = \{ p_1, p_2, \ldots, p_m \}$, se define el conjunto de ***predicados minitérmino*** como $M = \{ m_1, m_2, \ldots, m_r \}$ como:

$$M = \{ m_i \mid m_i = \wedge_{pj \in Pr} p_j^* \}, \ 1 \leq j \leq m, \ 1 \leq i \leq z$$

Donde, pj* = pj o pj* = ¬ (pj).

Ejemplo de minitérmino de la relación *J*:

m_1: JNOMBRE == "Mantenimiento" $\wedge$ Presupuesto $\leq$ 200000
m_2: NOT(JNOMBRE == "Mantenimiento") $\wedge$ Presupuesto $\leq$ 200000
m_3: JNOMBRE == "Mantenimiento" $\wedge$ NOT (Presupuesto $\leq$ 200000)
m_4: NOT(JNOMBRE == "Mantenimiento") $\wedge$ NOT(Presupuesto $\leq$ 200000)

Con respecto a la información cuantitativa de las aplicaciones de usuario, es necesario contar con los siguientes dos conjuntos de datos:

- ✓ Selección de los minitérminos: Indicada como *sel*(m_i), representa el número de tuplas de la relación que serán accedida por una consulta de usuario definida de acuerdo a un predicado minitérmino dado.

- ✓ Frecuencia de acceso: Expresada como *acc*(q_i), indica la frecuencia con la cuál una consulta de usuario q_i es accedida en un periodo de tiempo. En otras palabras la frecuencia de minitérmino se puede especificar como *acc*(m_i). **[F]**

La fragmentación horizontal primaria: Se define mediante una selección en las relaciones de un esquema de la base de datos. La fragmentación de una relación *R* esta dada por:

$$R_j = \sigma_{Fj}(R),\ 1 \leq j \leq w$$

F_j es una fórmula de selección, la cuál es preferiblemente un predicado minitérmino. Por lo tanto, un fragmento horizontal R_i de una relación *R* consiste de todas la tuplas de *R* que satisfacen un predicado minitérmino m_i. Lo anterior involucra que proporcionado un conjunto de predicados minitérmino *M*, hay muchos fragmentos horizontales de *R* como minitérminos. El conjunto de fragmentos horizontales en ocasiones son llamados *fragmentos minitérminos*.

En la fragmentación horizontal es necesario un algoritmo que tenga como entrada una relación *R* y el conjunto de predicados simples *Pr,* arrojando como resultado un conjunto de fragmentos $R = \{ R_1, R_2, \ldots, R_m \}$ el cuál cumple con las reglas de fragmentación. Los predicados deben ser completos y minimales.

Un conjunto de predicados simples *Pr* se dice que es *completo* si y solo si los accesos a las tuplas de los fragmentos minitérminos definidos en *Pr* requieren que dos tuplas del mismo fragmento tengan la misma probabilidad de ser accedidos por cualquier aplicación.

Si un predicado interviene en la fragmentación, significa que un segmento *f* se fragmenta mas en f_i y f_j solo a si se tendría una consulta que accede a f_i y f_j de manera diferente. Para que *Pr* sea mínimo, los predicados de un conjunto *Pr* deben ser relevantes.

Relevancia: Sean m_i y m_j dos predicados minitérminos definidos exactamente igual excepto que m_i contiene a p_i y m_j contiene a p_j. También, sean f_i y f_j los dos fragmentos definidos de acuerdo a m_i y m_j, respectivamente. Entonces, p_i es relevante si y solo si: **[F]**

$$acc(m_i)/card(f_i) \neq acc(m_j)/card(f_j)$$

1.7.2.3.1.1. Algoritmo COM_MIN

Algoritmo encargado de producir un conjunto completo y mínimo de predicados *Pr'* dado un conjunto de predicados simple *Pr.* Cabe mencionar que este algoritmo hace uso de la regla completes y minimalidad (una relación o fragmento es dividido por lo menos en dos partes las cuáles se acceden de distinta forma por al menos una consulta de usuario).

El algoritmo inicia, encuentra un predicado que divide la relación de entrada. Continúa anexando predicados al conjunto, dando minimalidad a cada proceso, resultando *Pr* completo y mínimo.

El segundo término en el proceso de diseño de fragmentación horizontal primaria es derivar el conjunto de predicados minitérminos que pueden ser definidos en los predicados del conjunto *Pr'*. Esos minitérminos definen los fragmentos que serán usados como candidatos en el paso de asignación. **[5]**

1.7.2.3.1.2. Algoritmo primario (PHORIZONTAL)

Este algoritmo tiene como entrada *Rj* la cuál se le aplicará la fragmentación primaria y Pr_i, es el conjunto de predicados simples que son establecidos por las consultas definidas en la relación R_i.

Los predicados simples que serían usados para particionar una relación S son:
p1 : SAL ≤ 30000
p2 : SAL > 30000

Cuando se emplea el algoritmo COM_MIN se comprueba que $Pr = \{ P_1, P_2 \}$ esta completo y minimal, *Pr' = Pr*. Haciendo referencia a la base de datos conceptual anterior (ver Figura 1.3), con esto se pueden crear los siguientes predicados minitérminos como elementos de *M*.

m_1: (SAL ≤ 30000) ∧ (SAL > 30000)
m_2: (SAL ≤ 30000) ∧ NOT (SAL > 30000)
m_3: NOT (SAL ≤ 30000) ∧ (SAL > 30000)
m_4: NOT (SAL ≤ 30000) ∧ NOT (SAL > 30000)

Tomando en cuenta que el dominio de SALARIO se puede dividir en dos, como p_1 y p_2, las consiguientes implicaciones son equivalentes:

i_1: (SAL $\leq$ 30000) $\Rightarrow$ NOT (SAL > 30000)
i_2: NOT (SAL $\leq$ 30000) $\Rightarrow$ (SAL > 30000)
i_3: (SAL > 30000) $\Rightarrow$ NOT (SAL $\leq$ 30000)
i_4: NOT (SAL > 30000) $\Rightarrow$ (SAL $\leq$ 30000)

De acuerdo a i_1, m_1 y i_2, m_4 son contradictorios; por esta razón se elije $M = \{m_2, m_3\}$, definiendo los fragmentos $F_s = \{S_1, S_2\}$ con respecto a M. **[5]**

S_1

TITULO	SALARIO
Ingeniero Mecánico	27000
Programador	24000

S_1

TITULO	SALARIO
Ingeniero Eléctrico	40000
Analista de Sistemas	34000

1.7.2.3.1.3. Corrección de la fragmentación horizontal

***Completitud*:** Si $P_{r'}$ es completo y mínimo, los predicados de selección son completos.

***Reconstrucción*:** Si la relación R es fragmentada en $F_R = (R_1, R_2, ..., R_r)$, entonces:

$$R = \bigcup \forall_{Ri \in FR} R_i$$

***Fragmentos disjuntos*:** Los predicados minitérminos que forman la base de la fragmentación deben ser mutuamente exclusivos. **[F]**

1.7.2.3.2. Vertical

Esta fragmentación divide un conjunto de atributos en partes pequeñas. Los fragmentos se logran proyectando la relación global sobre cada grupo, para que esta fragmentación sea correcta, los atributos deben ser mapeados en al menos un atributo del fragmento. La fragmentación vertical de una relación R origina fragmentos R_1, R_2,..., R_r, estos fragmentos contienen un subconjunto de atributos y llaves primarias de R. El objetivo principal de esta fragmentación es subdividir una relación en conjuntos pequeños, para que las aplicaciones de usuarios sean ejecutadas sobre un fragmento, reduciendo el tiempo de ejecución de una consulta de usuario.

Debido a que la fragmentación vertical cuenta con varias alternativas para su realización, se hará uso de la alternativa heurísticas para hacer la división de los

fragmentos. A continuación se hace mención de los enfoques básicos con los que cuenta la alternativa heurística:

- ✓ Agrupamiento: Da un atributo a cada fragmento, si los fragmentos cumplen con los requisitos, se creara un solo fragmento.
- ✓ División: Comienza con una relación para hacer una división basada en el comportamiento de acceso de las consultas sobre los atributos.

Requerimientos de información para la fragmentación vertical: Para llevar a cabo el particionamiento vertical, se debe de contar con información de dicha fragmentación, debido a que la fragmentación vertical ubica los atributos en fragmentos que se acceden juntos, por este motivo será necesario de un orden que relacione la igualdad de los atributos; así como identificar el grado de relación entre ellos. Esto se logra mediante datos primitivos.

Dado un conjunto de consultas $Q = \{ q_1, q_2, ..., q_q \}$ que serán aplicadas a la relación $R [A_1, A_2, ..., A_n]$, se define la función:

$$use(q_i, A_j) = \begin{cases} 1 \text{ Si el atributo } A_j \text{ es referido por la consulta } q_i \\ 0 \text{ en caso contrario} \end{cases}$$

Si se conocen las aplicaciones que se usarán en la base de datos, será sencillo definir los vectores *use* (q_i, ●).

Ejemplo: Suponiendo que las siguientes consultas se definen sobre la relación *J*:

q_1: Encontrar el presupuesto de un proyecto dado su número de identificación.

```
SELECT PRESUPUESTO
FROM J
WHERE JNO=valor
```

q_2: Encontrar los nombres y presupuestos de todos los proyectos.

```
SELECT JNOMBRE, PRESUPUESTO
FROM J
```

q_3: Encontrar los nombres de los proyectos en una ciudad dada.

```
SELECT JNOMBRE
FROM J
WHERE LUGAR=valor
```

q_4: Encontrar el presupuesto total de los proyectos en cada ciudad.

```
SELECT SUM(PRESUPUESTO)
FROM J
WHERE LUGAR=valor
```

Sean A_1=*JNO*, A_2=*JNOMBRE*, A_3=*PRESUPUESTO*, A_4=*LUGAR*. La función *use* se representar con la siguiente matriz:

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

La igualdad entre los atributos A_i y A_j de una relación R [A_1, A_2, ..., A_n] con respecto al conjunto de consultas Q = { q_1, q_2, ..., q_q } se define a continuación:

$$\text{aff}(A_i,A_j) = \Sigma_{\text{las consultas que accesan Ai y Aj}} \Sigma_{\text{Sl}} (\text{ref}_l(q_k)\ \text{acc}_l(q_k))$$

Donde, $ref_l(q_k)$ es el número de accesos a los atributos (A_i, A_j) para cada ejecución de la consulta q_k en el sitio S_l y $acc_l(q_k)$ es la frecuencia de acceso de la consulta previamente definida y modificada para incluir las frecuencias en sitios diferentes.

La afinidad de los atributos A1 y A3 se puede medir como:

$$aff(A_1, A_3) = \sum_{k=1}^{1}\sum_{l=1}^{3} acc_l(q_k) = acc_1(q_1) + acc_2(q_1) + acc_3(q_1) = 45$$

Ya que la única aplicación que accede a los dos atributos es q_1. La matriz de afinidades entre atributos, *AA*, es: **[F]**

$$\begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{bmatrix} \end{array}$$

1.7.2.3.2.1. Algoritmo de agrupamiento (Clustering)

Este algoritmo se encarga de tomar la matriz de afinidades entre los atributos (AA) y reorganizar su orden formando un grupo con alta afinidad entre ellos. **[F]**

1.7.2.3.2.2. Algoritmo de energía acotada (BEA)

Se encarga de encontrar un ordenamiento de los atributos, para poder maximizar la siguiente *medida de afinidad global* (AM):

$$AM = \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)\left[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1}) + aff(A_{i-1}, A_j) + aff(A_i, A_{j-1})\right]$$

Donde,

$$aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) = aff(A_i, A_{n+1}) = 0$$

Entrada: La matriz de afinidades entre atributos *AA*.
Salida: La matriz de afinidades agrupada, *CA*, la cuál es una perturbación de *AA*.

Iniciación: Colocar y fijar una de las columnas de *AA* en *CA*.
Iteración: Colocar las restantes *n-i* columnas en las restantes *i*+1 después en la matriz *CA*. Para cada columna, se elije el lugar que ayude a la medida de afinidad global.
Ordenamiento de renglones: Ordenar los renglones de acuerdo al orden de las columnas.

Para poder definir un lugar, se precisa la contribución de una ubicación. **[F]**

$$cont(A_i, A_k, A_j) = 2bond(A_i, A_k) + 2bond(A_k, A_l) - 2bond(A_i, A_j)$$

Donde,

$$bond(A_x, A_y) = \sum_{z=1}^{n} aff(A_z, A_x)aff(A_z, A_y)$$

1.7.2.3.2.3. Algoritmo de particionamiento

Dedicado a la localización de atributos que son accedidos de manera única por un conjunto de consultas. Si se fija un punto a lo largo de la diagonal, se identifican dos conjuntos de atributos. El conjunto *arriba es* $\{A_1, \ldots, A_i\}$ ubicado en la esquina superior izquierda y el conjunto *abajo* $\{A_{i+1}, \ldots, A_n\}$ se ubica en la esquina inferior derecha (ver la Figura 1.6).

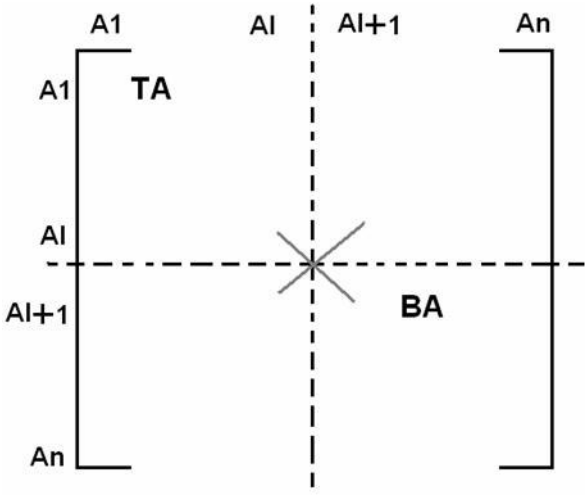


Figura 1.6: Localización del punto de división. **[F]**

Tomando el conjunto de consultas $Q = \{ q_1, q_2, ..., q_q \}$ y definiendo el conjunto de aplicaciones que acceden a *TA*, *BA*, o ambas. Aquí el problema radica en localizar el punto que maximice la función objetivo a lo largo de la diagonal.

$$z = CTQ * CBQ - COQ^2$$

CTQ: Número total de accesos a atributos por aplicaciones que acceden a TA.
CBQ: Número total de accesos a atributos por aplicaciones que acceden a BA.
COQ: Número total de accesos a atributos por aplicaciones que acceden a TA y BA.

Esta expresión permite equilibrar las cargas de procesamiento cuando existe una distribución de fragmentos, ya que CTQ y CBQ tienen valores equivalentes.

Hay dos dificultades de usar esta sintaxis:

- ✓ El particionamiento en ocasiones se forma a la mitad de la matriz *CA*, cuando surge este problema solo se tiene que hacer un corrimiento de un renglón hacia arriba y una columna a la izquierda, para localizar el punto correcto.
- ✓ La aparición de más de un grupo complicando la localización del punto de particionamiento. **[F]**

1.7.2.3.2.4. Corrección de la fragmentación vertical

Completitud: Esta garantizada por el algoritmo de particionamiento. Debido a que los atributos de la relación global se asignan a uno de los fragmentos. Para asegurar la completitud de la fragmentación vertical, el conjunto de atributos *A* sobre los cuáles se define una relación *R* radica en:

$$A = TA \cup TB$$

Reconstrucción: Esta encargada de reconstruir la relación global original, usando la operación de junta. Así que la relación *R,* con fragmentación vertical $F_R = \{R_1, R_2,..., R_r\}$ y llave *K*, esta dada por:

$$R = \rhd \lhd_K R_i \; \forall \; R_i \in F_R$$

Si R_i esta completo, la operación de junto reconstruirá correctamente *R*.

Fragmentos disjuntos. Existen dos casos:

- ✓ Los identificadores de tuplos no se considera que se traslapen, porque son mantenidos por el sistema.
- ✓ Las llaves duplicadas no se considera que se traslapen. **[F]**

1.7.2.3.3. Híbrida

Este tipo de fragmentación es usada para cumplir con los requerimientos de las aplicaciones de usuarios, cuando no son resueltas con la fragmentación horizontal y vertical. Esto genera un árbol de particionamiento estructurado (ver la Figura 1.7) donde se aplica la fragmentación horizontal y posteriormente la fragmentación vertical o viceversa. **[5]**

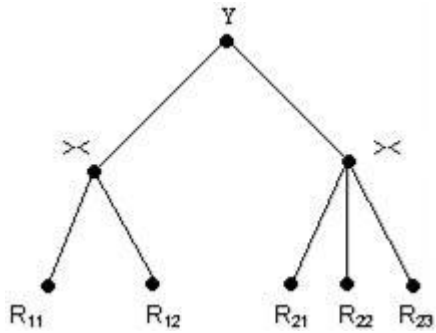


Figura 1.7: Fragmentación híbrida. **[6]**

1.7.3. Asignación

Para poder realizar la asignación es necesario contar con datos cualitativos sobre la base de datos, las aplicaciones que serán usadas, la red de comunicación, características de procesos y el límite de almacenamiento del lugar de la red. **[6]**

1.7.3.1. Asignación de fragmentos

Suponiendo que se cuenta con un conjunto de fragmentos *F = {F1, F2, …, Fn}* y un conjunto de nodos *S = {S1, S2, …, Sm}*. El problema de asignación establece la distribución óptima de *F* en *S*. Esta optimización se da mediante:

Coste mínimo: Es la identificación de los costos necesarios para tener una comunicación de datos, almacenamiento y procesamiento.

Rendimiento: Las estrategias de asignación son diseñadas para lograr un beneficio de rendimiento, en cuanto a tiempo de respuesta y productividad.

Con respecto a las estrategias de asignación de fragmentos se mencionan enseguida:

No soportar replicación: Cada fragmento es ubicado en un solo lugar.

Soporta replicación completa: Cada fragmento se localiza en un nodo.

Soporta replicación parcial: Cada fragmento reside en alguno de los sitios. **[7]**

Nota: Cuando se cuenta con muchas consultas de solo lectura, se recomienda la replicación.

1.7.3.2. Asignar fragmentos a esquemas locales

Sin replicación: Cada fragmento se localiza en un solo nodo, esto es bueno para actualizaciones y malo para consultas.

Con replicación total: Todos los fragmentos se encuentran en todos los nodos, siendo bueno para consultas, malo para actualizaciones.

Con replicación parcial: Algunos fragmentos tienen la posibilidad de estar en más de un nodo. **[6]**

1.7.4. Replicación

La replica de base de datos proporciona grandes beneficios, así como desventajas, esto dependerá de la distribución y tipo de replica que se use, para poder disponer de los datos, a continuación se hace mención de estas características:

Replica completa de la base de datos en cada nodo: Ayuda a mejorar el rendimiento local y global; así como la disponibilidad de los datos.

Replica parcial: Los fragmentos se localizan en diferentes lugares, esto genera que si se desea hacer una consulta a un nodo que no funciona, no se podrá acceder a la información. **[8]**

1.8. Consultas

En la actualidad el proceso de consultas, se le ha dado una gran importancia debido al surgimiento de muchos lenguajes no procedurales (esconder detalles de bajo nivel con respecto a la localización física de datos), usados para la realización de estas, debido a que estos lenguajes de aplicación ayudan al desarrollo de aplicaciones y la productividad del usuario. En otras palabras la respuesta a una consulta es realizada por el modulo del SGBD (procesador de consultas).

1.8.1. Problema de procesamiento de consultas

El problema principal para lograr los procesos de consulta radica en la construcción o selección de una estrategia adecuada para reducir el consumo de recursos.

En un argumento centralizado, las estrategias de ejecución de consultas y el álgebra relacional no es suficiente para expresar la ejecución de estrategias. Esta estrategia tiene que complementarse con operaciones para el intercambio de datos entre nodos distintos; así como elegir los nodos para procesarlos. **[F]**

1.8.2. Objetivos de la optimización de consultas

Este objetivo consiste en la conversión de una consulta en una especificación de alto nivel a una estrategia de ejecución eficaz expresada en un lenguaje de bajo nivel. Con la siguiente función se reduce el costo de la consulta:

Función de costo total = costo de I/O + costo de CPU + costo de comunicación

1.8.3. Complejidad de las operaciones del álgebra relacional

Esta complejidad afecta directamente el tiempo de ejecución y crean ciertos principios necesarios para el procesador de consultas. La forma más fácil de definir la complejidad es el uso de la cardinalidad de las relaciones independientemente de los detalles de implementación como fragmentación y estructuras de almacenamiento. **[F]**

1.8.4. Características de los procesadores de consultas

Estas características son variadas dependiendo del SGBD y la ubicación de las bases de datos:

Tipo de optimización: Es importante saber que tipo de optimización será utilizada para lograr que un proceso sea óptimo durante la ejecución, si se emplea la búsqueda exhaustiva (aplicada a consultas simples) ya que tiene una complejidad combinada con el número de relaciones de la consulta. Los algoritmos heurísticos consiguen solo aproximaciones a la transformación óptima.

***Tiempo de optimización*:** Para lograr la optimización de una consulta se puede aplicar la optimización estática y dinámica, por un lado la estática se puede hacer antes de la consulta, no a si con la dinámica, ya que esta es ejecutada durante la consulta y se debe de tener una estadística del tamaño de los resultados intermedios.

***Estadísticas**:* Facilitan la elección de las operaciones que serán ejecutadas primero.

Nodos de Decisión: Al escoger un nodo de decisión, se debe de hacer con mucho cuidado ya que ayudan a elegir la estrategia que será aplicada para ejecutar cierta consulta.

Topología de la Red: Puede favorecer o perjudicar la función objetivo a optimizar para elegir la estrategia de ejecución, por esta razón se recomienda implementar la red con mucho cuidado. **[F]**

1.8.5. Arquitectura del procesamiento de consultas

Esta se puede dividir en diferentes subproblemas correspondientes a diferentes niveles.

Ejemplo: Suponiendo que se tiene un procesador de consultas estático semicentralizado en donde no se tienen fragmentos replicados. En este ejemplo se hará uso de las capas de *descomposición de consultas*, *localización de datos*, *optimización global de consultas* y *optimización local de consultas* (ver Figura 1.8).

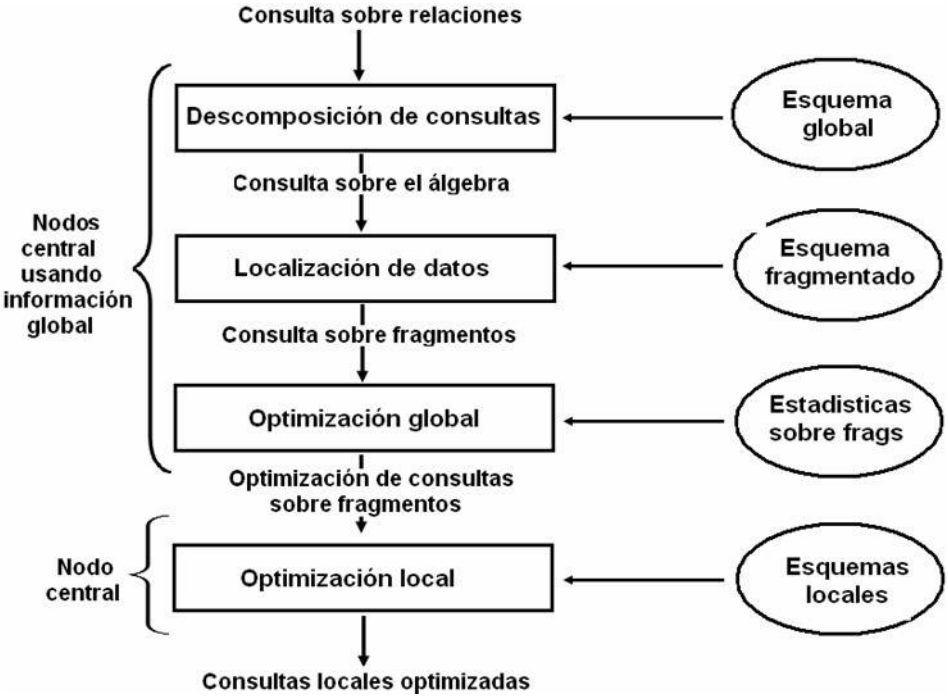


Figura 1.8: Arquitectura en capas del procesamiento de consultas. **[F]**

Descomposición de consultas: Para poder efectuar la descomposición de una consulta en el álgebra relacional que opera sobre relaciones globales, será necesario contar con lo siguiente:

- ✓ *Normalización:* Implica la manipulación de los cuantificadores de la consulta, los calificadores y la aplicación de la prioridad de los operadores lógicos.
- ✓ *Análisis:* Se encuentran y eliminan consultas semánticamente erróneas.
- ✓ *Simplificación:* Suprime predicados repetidos.
- ✓ *Reestructuración:* Se aplica para mejorar las aplicaciones, cuando existe una falla en la ejecución de una consulta algebraica.

Localización de datos: Permite que fragmentos participan en la consulta y convierte la consulta distribuida en una consulta sobre fragmentos.

Optimización global de consultas: El punto principal es encontrar una estrategia de ejecución para la consulta con el objeto de lograr la optimización.

Optimización local de consultas: Estas consultas se realizan en los nodos de consulta local. En otras palabras se efectúa en todos los nodos con fragmentos que participan en la consulta. **[F]**

1.8.6. Descomposición de consultas

Para lograr la descomposición de las consultas será necesario de la normalización, análisis, simplificación y reestructuración, las cuáles se explican más adelante.

Normalización: La normalización se encarga de convertir una consulta a una forma normalizada para lograr facilitar su procesamiento. Para lograr la normalización se hace uso del análisis léxico y sintáctico, los cuáles se encargan de que las expresiones que dan origen a la consulta sean correctas.

Para lograr la normalización existen dos formas nórmales:

1.- Forma normal conjuntiva (conjunto de disyunciones):

$$(p_{11} \vee p_{12} \vee \ldots \vee p_{1n}) \wedge (p_{21} \vee p_{22} \vee \ldots \vee p_{2n}) \wedge \ldots \wedge (p_{m1} \vee p_{m2} \vee \ldots \vee p_{mn})$$

2.- Forma normal disyuntiva (disyunción de conjunciones):

$$(p_{11} \wedge p_{12} \wedge \ldots \wedge p_{1n}) \vee (p_{21} \wedge p_{22} \wedge \ldots \wedge p_{2n}) \vee \ldots \vee (p_{m1} \wedge p_{m2} \wedge \ldots \wedge p_{mn})$$

Análisis: El análisis se usa en la eliminación de consultas normalizadas que no requieren mucho procesamiento. Factores para eliminar una consulta normalizada:

- ✓ Si los atributos no están definidos en el esquema global.
- ✓ Si las operaciones que se usan en los atributos no son los adecuados.
- ✓ Si los componentes no ayudan a generar el resultado.

El análisis representa una consulta con una gráfica de conectividad o de consulta (ver Figura 1.9), donde:

- ✓ Los nodos especifican la relación resultante: Cualquier otro nodo representa la relación operante.
- ✓ Un arco representa una junta (ver Figura 1.10): Si un arco tiene un nodo destino será una relación resultante representando una proyección.
- ✓ Un nodo no resultado puede ser etiquetado por un predicado de selección.

Ejemplo: Haciendo referencia a la Figura 1.3, "*Encontrar los nombres y responsabilidades de los programadores que han estado trabajando en el proyecto de CAD/CAM por más de tres años y el nombre de su administrador*" **[F]**

La consulta expresada en SQL es:

```
SELECT ENOMBRE, RESP
FROM E, G, J
WHERE E.ENO = G.ENO AND G.JNO = J.JNO AND JNOMBRE = "CAD/CAM"

AND DUR ≥ 36 AND TITLE = "Programador"
```

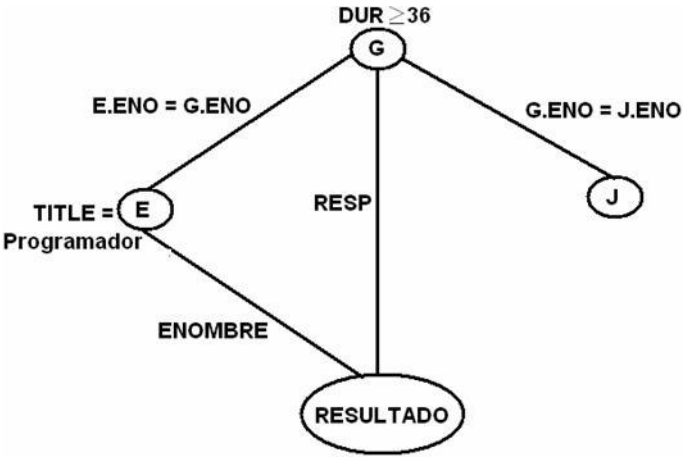


Figura 1.9: Gráfica de la consulta. **[F]**

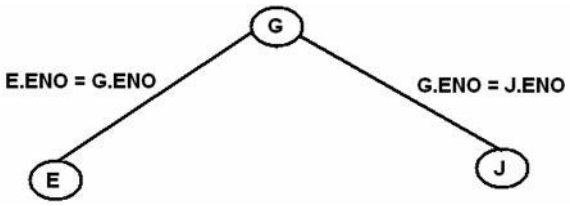


Figura 1.10: Gráfica de juntas. **[F]**

Eliminación de redundancia: Hace uso de reglas idempotencia para eliminar la redundancia de predicados en las consultas de forma normal conjuntiva:

- ✓ $p \wedge p \Leftrightarrow p$
- ✓ $p \vee p \Leftrightarrow p$
- ✓ $p \wedge true \Leftrightarrow p$
- ✓ $p \vee false \Leftrightarrow p$
- ✓ $p \wedge false \Leftrightarrow false$
- ✓ $p \vee true \Leftrightarrow true$
- ✓ $p \wedge \neg p \Leftrightarrow false$
- ✓ $p \vee \neg p \Leftrightarrow true$
- ✓ $p_1 \wedge (p_1 \vee p_2) \Leftrightarrow p_1$
- ✓ $p_1 \vee (p_1 \wedge p_2) \Leftrightarrow p_1$

Ejemplo de consulta SQL de la base de datos de la Figura 1.3.

```
SELECT TITULO
FROM E
WHERE (NOT (TITULO = "Programador"))
AND (TITULO = "Programador"
OR TITULO = "Ingeniero Eléctrico")
AND NOT (TITULO = "Ingeniero Eléctrico")
OR ENOMBRE = "J. Doe"
```

Lo anterior se reduce de la siguiente manera:

```
SELECT TITULO
FROM E
WHERE ENOMBRE = "J. Doe"
```

Reestructuración: En este punto se reescribe la consulta en el álgebra relacional mediante los siguientes pasos:

- ✓ Una transformación directa del cálculo relacional en el álgebra relacional.
- ✓ Una reestructuración de la consulta en el álgebra relacional para mejorar la eficiencia

Se crea una hoja diferente para cada variable de tuplo diferente, las hojas se encuentran en la cláusula FROM.

El nodo raíz se crea como operación de proyección involucrando a los atributos resultantes, esta se localizan en la cláusula SELECT.

La calificación WHERE es traducida en un orden adecuado de operaciones relacionales (select, join, unión) de hojas a raíz.

Ejemplo: Haciendo referencia a la Figura 1.3, "*Encontrar los nombres de empleados diferentes de "J. Doe" que trabajaron en el proyecto de CAD/CAM por uno o dos años*"

El mapeo del ejemplo, se logra de manera directa al árbol del álgebra relacional (ver Figura 1.11).

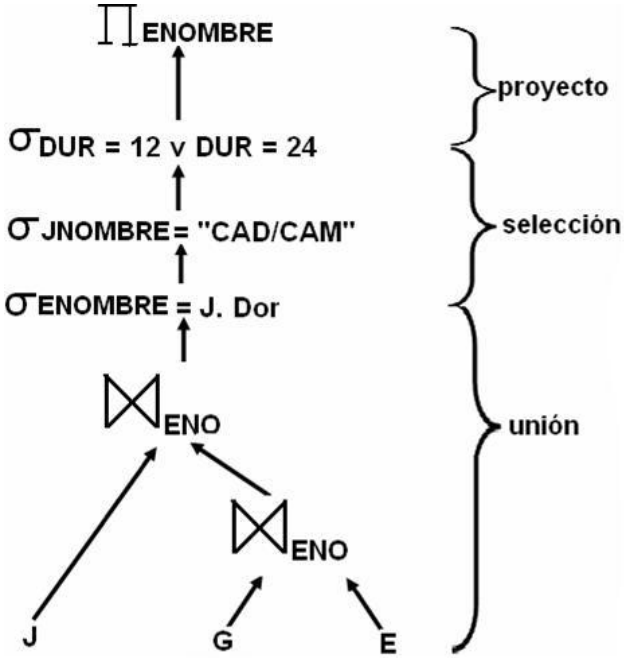


Figura 1.11: Árbol del álgebra relacional. **[F]**

La consulta SQL de este ejemplo se puede realiza de la siguiente manera:

```
SELECT ENAME
FROM J, E, G
WHERE E.ENO = G.ENO
AND G.JNO = J.JNO
AND ENAME ≠ "J. Doe"
AND JNAME = "CAD/CAM"
AND (DUR = 12 OR DUR = 24)
```

Si se aplican las siguientes reglas de transformación (reglas de equivalencia más usadas). Sea *R*, *S* y *T* son relaciones donde *R* se define sobre los atributos $A = \{A_1, A_2,..., A_n\}$ y *S* se define sobre los atributos $B = \{B_1, B_2,..., B_n\}$, quedando de la siguiente forma:

- ✓ Conmutatividad de operaciones binarias:

$R \times S = S \times R$
$R >< S = S >< R$
$R \cup S = S \cup R$

- ✓ Asociatividad de operaciones binarias:

$R \times (S \times T) \Leftrightarrow (R \times S) \times T$
$R >< (S >< T) \Leftrightarrow (R >< S) >< T$

- ✓ Idempotencia de operaciones unarias:

$\Pi_{A'} (\Pi_{A''} (R)) \Leftrightarrow \Pi_{A'} (R)$
$\sigma_{p1(A1)} (\sigma_{p2(A2)} (R)) \Leftrightarrow \sigma_{p1(A1) \wedge p2(A2)} (R)$

Donde $R[A]$ y $A' \subseteq A$, $A'' \subseteq A$ y $A' \subseteq A$.

- ✓ Conmutando selección con proyección:

$\Pi_{A1, ..., An} (\sigma_{p(Ap)} (R)) \Leftrightarrow \Pi_{A1, ..., An} (\sigma_{p(Ap)} (\Pi_{A1, ..., An, Ap} (R)))$

- ✓ Conmutando selección con operaciones binarias:

$\sigma_{p(A)} (R \times S) \Leftrightarrow (\sigma_{p(A)} (R)) \times S$
$\sigma_{p(Ai)} (R >< _{(Aj, Bk)} S) \Leftrightarrow (\sigma_{p(Ai)} R) >< _{(Aj, Bk)} S$
$\sigma_{p(Ai)} (R \cup T) \Leftrightarrow \sigma_{p(Ai)} (R) \cup \sigma_{p(Ai)} (T)$

Donde A_i pertenece a *R* y a *T*.

- ✓ Conmutando proyección con operaciones binarias:

$\Pi_C (R \times S) \Leftrightarrow \Pi_{A'} (R) \times \Pi_{B'} (S)$
$\Pi_C (R >< _{(Aj, Bk)} S) \Leftrightarrow \Pi_{A'} (R) >< _{(Aj, Bk)} \Pi_{B'} (S)$
$\Pi_C (R \cup S) \Leftrightarrow \Pi_C (R) \cup \Pi_C (T)$

Donde $R[A]$ y $S[B]$; $C = A' \cup B'$ donde $A' \subseteq A$ y $B' \subseteq B$. **[F]**

1.9. Transacciones

Lo fundamental en una transacción es la combinación de la consistencia y confiabilidad de la base de datos.

1.9.1. Definición de una transacción

"Recopilación de acciones que hacen transformaciones consistentes de los estados de un sistema preservando la consistencia del sistema". Lo que se busca al hacer uso de la transacción es que no existan errores en las acciones concurrentes y manejo de las fallas de una base de datos.

1.9.2. Condiciones de terminación de una transacción

Con respecto a la terminación de una transacción; es necesario decir, que una transacción siempre termina con o sin errores su proceso. Cuando la transacción termina satisfactoriamente está realiza un *commit*, en caso de que ocurra un error y no terminé la transacción se dice que *aborta* desechando las acciones que esté ejecutando, de esta manera la base de datos regresara a su estado inicial. **[5]**

1.9.3. Caracterización de transacciones

Esta opción se refiere a los componentes de los datos que leen una transacción, formado por el *conjunto de lectura* (RS) y el *conjunto de escritura* (WS). La unión de los conjuntos anteriores se les llama *conjunto base* de la transacción ($BS = RS \cup WS$). **[5]**

1.9.4. Formalización del concepto de transacción

Sea $O_{ij}(x)$ una operación O_j de la transacción T_i la cuál trabaja sobre alguna entidad x. $O_j \in$ {read, write} y O_j es una operación atómica; esto se ejecuta como una unidad indivisible. Se indica por $OS_i = \in_j O_{ij}$ al conjunto de todas las operaciones de la transacción T_i. También, se denota por N_i, la condición de finalización para T_i, donde, $N_i \in$ {abort, commit}.

La transacción T_i es un orden parcial, $T_i = \{ \Sigma_i, <_i \}$, donde: **[5]**

1. $_i = OS_i \cup \{N_i\}$
2. Para cualesquiera dos operaciones, O_{ij}, $O_{ik} \in OS_i$, si $O_{ij} = R(x)$ y $O_{ik} = W(x)$ para cualquier elemento de datos x, entonces, ó $O_{ij} <_i O_{ik}$ ó $O_{ik} <_i O_{ij}$
3. $O_{ij} \in OS_i$, $O_{ij} <_i N_i$

1.9.5. Propiedades de las transacciones

Atomicidad: Significa que todas las acciones de las transacciones se deben de realizar correctamente, ya que si no se realizan de manera correcta estas son eliminadas.

***Consistencia*:** Es cuando una transacción es correcta, sin problema o error alguno.

***Aislamiento*:** Indica que las transacciones que se encuentran en ejecución deben de esconder sus resultados ante otras transacciones concurrentes antes de su *commit*.

***Durabilidad*:** Confirma que los resultados obtenidos por una transacción al hacer *commit* no podrán ser eliminados de la base de datos. **[5]**

1.9.6. Tipos de transacciones

De acuerdo al área de aplicación: Las transacciones pueden ser locales y distribuidas.

Por su tiempo de duración: Se toma desde el inicio de una transacción asta la culminación de un *commit* o se *aborta*.

Por su estructura.

1.9.7. Estructura de transacciones

Transacciones planas: Secuencia de operaciones primitivas localizadas dentro de *begin* y *end*.

Transacciones anidadas: Las operaciones de una transacción también pueden ser transacciones con las mismas propiedades de la transacción padre. Esto da origen a un nivel más alto de concurrencia. A continuación se mencionan los diferentes tipos de modelos de las transacciones:

- ✓ Si existe una mezcla de acciones de lectura y escritura, serán llamadas acciones *generales*.
- ✓ Si un dato se limita a ser leído antes que sea escrito se tendrá una transacción *restringida*.
- ✓ Si las transacciones son restringidas a que todas las acciones de lectura se realicen antes de las acciones de escritura entonces se les conoce como de *dos.*
- ✓ Para las transacciones restringidas, está un modelo de *acción* donde se aplica aún más la restricción de cada par <read, write>, teniéndose que ejecutar de manera atómica. **[5]**

1.9.8. Aspectos relacionados al procesamiento de transacciones

Modelo de estructura de transacciones: Verifica si las transacciones son planas o pueden estar anidadas.

Consistencia de la base de datos interna: Los algoritmos de control de datos semántico tienen que satisfacer siempre las restricciones de integridad cuando una transacción va hacer un commit.

Protocolos de confiabilidad: Es necesario anexar medios de comunicación entre los diferentes nodos de una red para garantizar la atomicidad y durabilidad de las transacciones. Así también, se requieren protocolos para la recuperación local y para hacer commit globales.

Algoritmos de control de concurrencia: Se encargan de combinar la ejecución de transacciones concurrentes bajo el criterio de correctitud.

Protocolos de control de réplicas: Forma de garantizar la consistencia entre los datos replicados. **[5]**

1.9.9. Incorporación del manejador de transacciones a la arquitectura de un SGBD

El monitor de ejecución distribuida consiste de dos módulos:

El administrador de transacciones (TM): Se encarga de coordinar la ejecución de las operaciones que realiza una aplicación en la base de datos.

El despachador (SC): Su función es la implementación de un algoritmo para el control de concurrencia para lograr la sincronización de los accesos a la base de datos.

En la Figura 1.12, se representa la ejecución distribuida de transacciones, así como los protocolos de comunicación necesarios para la administración de transacciones distribuidas. **[F]**

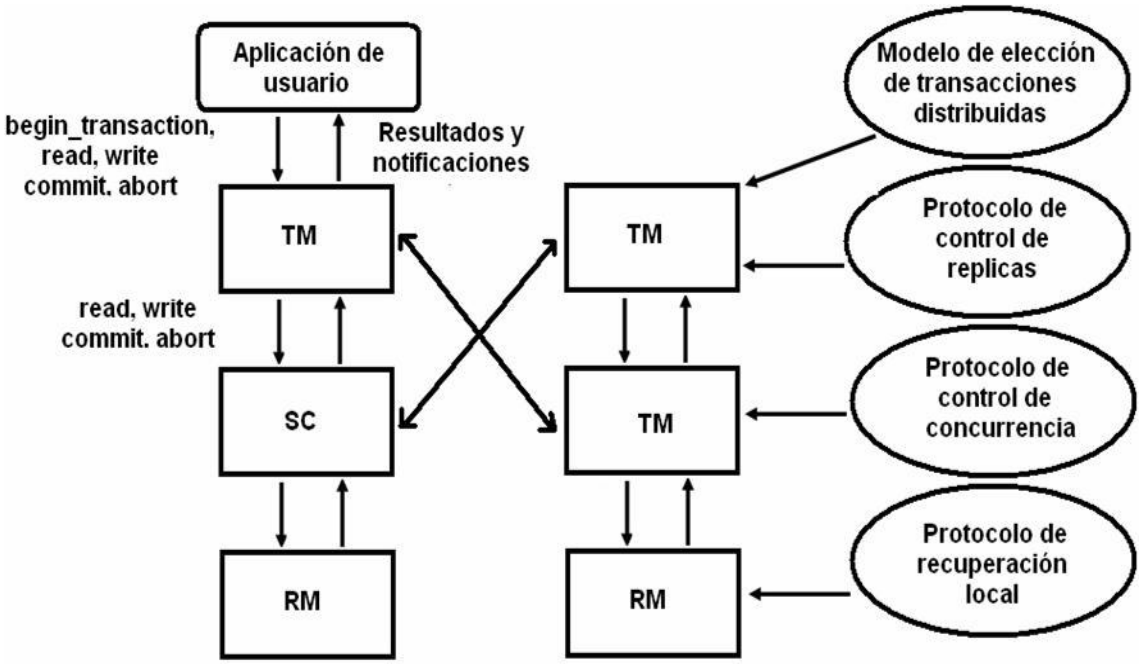


Figura 1.12: Arquitectura para la ejecución distribuida de transacciones. **[F]**

1.10. Concurrencia

Para lograr que la concurrencia sea eficiente debe asegurar la consistencia de los procesos y problemas de las transacciones. En las bases de datos distribuidas la concurrencia se ejecuta en varios nodos. Por esta razón los componentes de control de concurrencia tienen el objetivo de hacer un equilibrio entre el mantenimiento de la consistencia de la base de datos y el mantenimiento de un alto nivel de concurrencia. En caso de que no se logre un buen control de concurrencia, esto generara los problemas de perdida de actualizaciones y recuperación de información incorrecta o inconsistente. **[F]**

1.10.1. Teoría de la seriabilidad

Una historia o *calendarización*, especificada sobre un conjunto de transacciones $T = \{ T_1, T_2, ..., T_n \}$, que define un orden entrelazado de la ejecución de las operaciones de las transacciones. La historia puede ser definida como un orden parcial sobre T.

Ejemplo: Tomando las siguientes transacciones:

T_1: Read(x)	T_2: Write(x)	T_3: Read(x)
Write(x)	Write(y)	Read(y)
Commit	Read(z)	Read(z)
	Commit	Commit

La calendarización de las acciones de las transacciones pueden ser las que se muestra continuación:

$$H_1 = \{ W_2(x), R_1(x), R_3(x), W_1(x), C_1, W_2(y), R_3(y), R_2(z), C_2, R_3(z), C_3 \}$$

Cuando dos operaciones $O_{ij}(x)$ y $O_{kl}(x)$ (i y k no necesariamente distintos) se encuentran en conflicto por acceder a un mismo dato, surgen dos conflictos *read-write* o *write-read* y *write-writ*e, los problemas pueden estar en una o en distintas transacciones.

Calendarización completa: Define el orden de ejecución de todas las operaciones en su dominio. Formalmente la calendarización completa S_T^c definido sobre un conjunto de transacciones $T = \{ T_1, T_2, ..., T_n \}$ es un orden parcial $S_T^c = \{ \Sigma_T, <_T \}$ donde:

1.- El dominio de la calendarización es la unión de los dominios de las transacciones individuales.

$\Sigma_T = \bigcup_i \Sigma_i$, para todos los $i = 1, 2, ..., n$

2.- Definiendo la relación de ordenamiento como un superconjunto de la relación de ordenamiento de transacciones individuales.

$<_T \supseteq \bigcup_i <_i$, para todos los $i = 1, 2, ..., n$

3.- Cuando dos operaciones están en conflicto O_{ij} y $O_{kl} \in \Sigma_T$, ó $O_{ij} <_T O_{kl}$ ó $O_{kl} <_T O_{ij}$

Definición de un prefijo de orden parcial: Dado un orden parcial $P = \{ \Sigma , < \}$, $P' = \{ \Sigma ', <' \}$, es un *prefijo* de P si:

1.- Definiendo a P' como una restricción de P' en el dominio $\Sigma '$, donde las relaciones de ordenamiento en P se mantienen por P'
$\Sigma' \subset \Sigma_i$
$\forall\ e_i \in \Sigma '$, $e_1 <' e_2$, si y solamente si, $e_1 < e_2$, y

2.- Para cualquier elemento de $\Sigma '$, todos sus predecesores en Σ deben ser incluidos en $\Sigma '$
$\forall\ e_i \in \Sigma '$, si $\exists\ e_j \in \Sigma$ y $e_j < e_i$, entonces, $e_j \in \Sigma '$

Si dos *historias*, S_1 y S_2, definidas sobre el mismo conjunto de transacciones T, se dice que son *equivalentes* si para cada par de operaciones en conflicto O_{ij} y O_{kl} ($i \neq k$), cada vez que $O_{ij} <_1 O_{kl}$, entonces, $O_{ij} <_2 O_{kl}$. A esta relación se le conoce como *equivalencia de conflictos* puesto que define la equivalencia de dos *historia* en término del orden de ejecución relativo de las operaciones en conflicto en ellas.

Una calendarización S se dice que es *serializable*, si y solo si, es equivalente por conflictos a una *historia* serial.

Por último cabe mencionar que el objetivo principal del controlador de concurrencia, es construir una calendarización serializable para la ejecución de todas las transacciones. **[F]**

1.10.2. Seriabilidad en SGBD distribuidos

Para poder crear calendarizaciones serializables en una base de datos distribuida es necesario contar con *calendarización local* y la *calendarización global* de las transacciones que se ejecutan en el sistema.

Para que funcionen las calendarizaciones globales será necesario que se cumplan los siguientes puntos:

- ✓ Las *historias locales* debe ser serializable.
- ✓ Las dos operaciones en conflicto tienen que tener el mismo orden relativo a las *historias locales* donde se encuentran las operaciones. **[F]**

1.10.3. Taxonomía de los mecanismos de control de concurrencia

En este punto se habla acerca de la clasificación de los algoritmos de control de concurrencia, los cuáles son clasificados de la siguiente manera:

- ✓ Algoritmos basados en accesos mutuamente exclusivos a datos compartidos (candados).

- ✓ Algoritmos que ordenan la ejecución de las transacciones de acuerdo a un conjunto de reglas (protocolos).

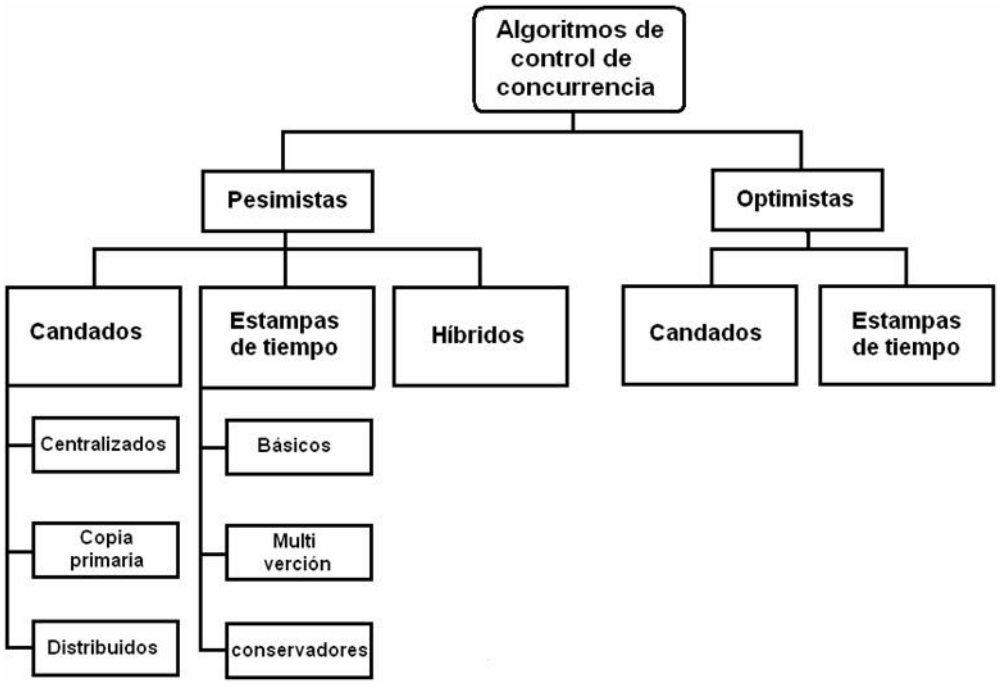


Figura 1.13: Clasificación de los algoritmos de control de concurrencia. **[F]**

Otro tipo de calificación de los algoritmos (ver Figura 1.13) es la siguiente:

- ✓ *Algoritmos pesimistas:* Sincronizan la ejecución concurrente de las transacciones en su etapa inicial de su ciclo de ejecución.
- ✓ *Algoritmos optimistas:* Detiene la sincronización de las transacciones hasta su finalización. **[F]**

1.10.4. Algoritmos basados en candados

También son llamados exclusivos, ya que sus candados son de lectura (rl) y escritura (wl), estas operaciones son incompatibles (ver Tabla 1.1). En estos algoritmos las transacciones indican sus intenciones solicitando candados al despachador.

	rl	wl
rl	si	no
wl	no	no

Tabla 1.1: Conflictos entre candados de lectura y escritura.

El despachador, es un administrador de candados (LM).

Candados de dos fases (2PL): En este candado una transacción le asigna un candado a un objeto antes de ser usado, obligando a la transacción solicitante a

esperar. Una vez que el candado se anula ya no se le podrá asignar otro candado (ver Figura 1.14).

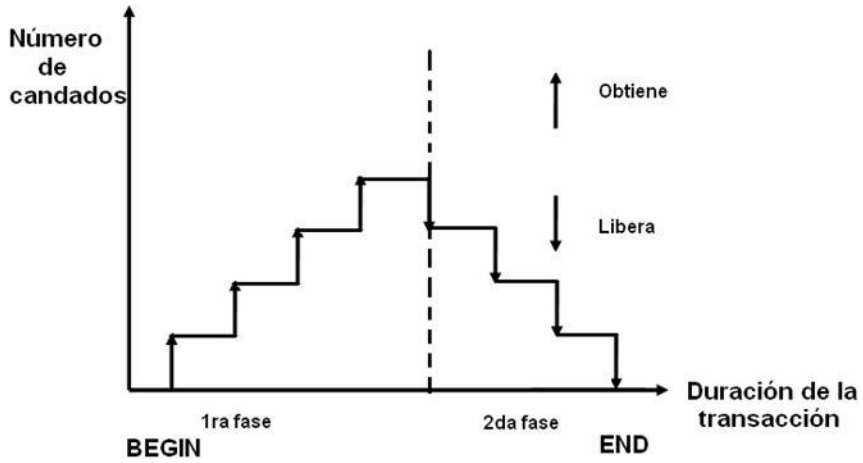


Figura 1.14: Uso de los candados de dos fases. **[F]**

En la 1a fase: Solicita los candados de los elementos que se usaran.
En la 2da fase: Libera los candados uno por uno.

La ventaja teóricamente del uso de los candados de dos fases, es por que las calendarizaciones creadas son centralizables.

Para evitar que las transacciones que acceden a un mismo dato *aborten* después de liberar un candado, los despachadores ponen en práctica los *candados estrictos de dos fases* (ver Figura 1.15), para liberar los candados cuando la transacción termina con un *commit* o *aborta*. **[F]**

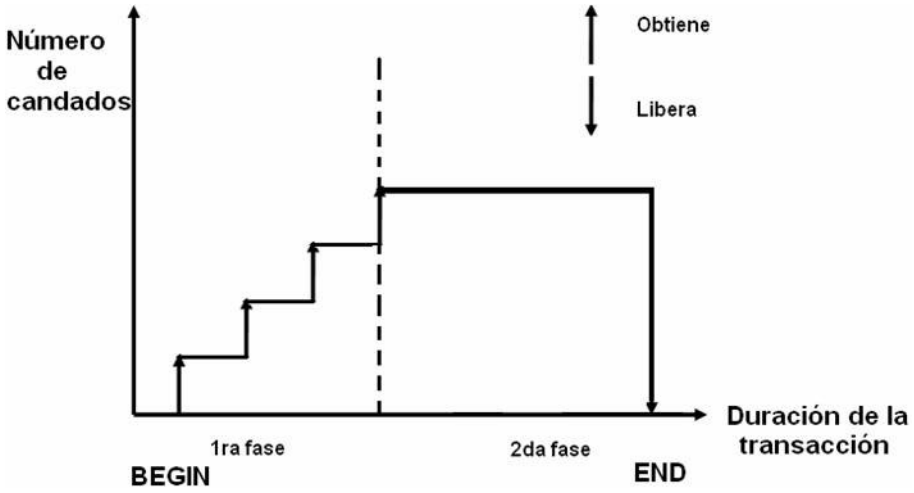


Figura 1.15: Comportamiento de los candados de dos fases estrictos. **[F]**

Candados de dos fases centralizados: En este candado el despachador central recibe todas las solicitudes de candados de sistema. En la Figura 1.16, la

comunicación se efectúa entre el administrador de transacciones del nodo donde se origina la transacción (*coordinador* TM), el administrador de candados en el nodo central y los procesadores de datos (DP) de todos los nodos participantes.

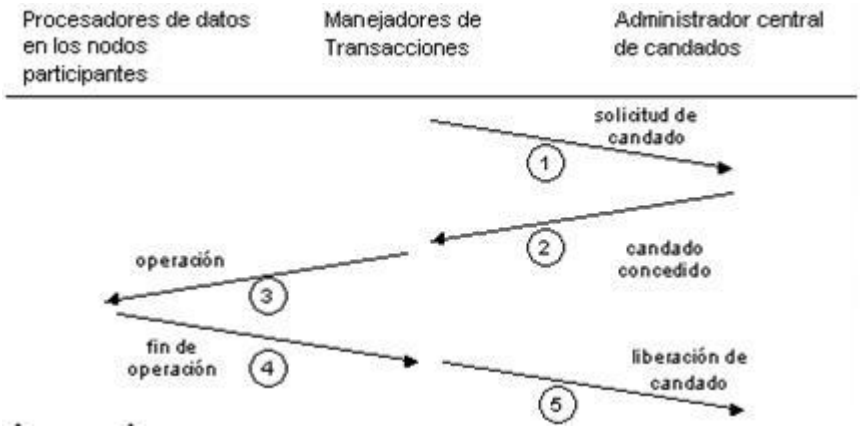


Figura 1.16: Comunicación en un administrador centralizado de candados de dos fases estrictos. **[F]**

Una de las desventajas de hace uso de este candado es el cuello de botella, lo que hace que disminuya el tiempo de respuesta de todo el sistema. **[F]**

Candados de dos fases distribuidos: Estos candados tienen un despachador en cada nodo, el cuál se encarga de las solicitudes de candados. Aquí las transacciones tienen la facilidad de leer todas las copias de replica de cualquier elemento para obtener un candado de lectura para cada copia. Esto hace que los mensajes de solicitud se manden a los administradores de los candados que actúan en el sistema. Las operaciones son pasadas por los administradores de candados a los procesadores de datos. Los procesadores de datos mandan un mensaje (fin de operación) al administrador de transacciones coordinador (ver Figura 1.17). **[F]**

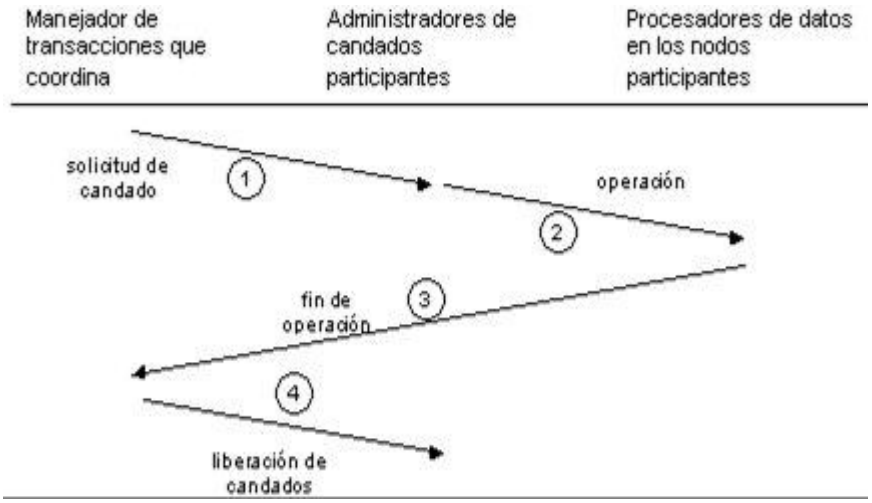


Figura 1.17: Comunicación en candados de dos fases distribuidos. **[F]**

1.10.5. Algoritmos basados en estampas de tiempo

Estos algoritmos se encargan de elegir un orden de serialización a priori y ejecutan las transacciones con respecto al orden seleccionado. El logro del ordenamiento de este algoritmo se logra, dando a cada transacción T_i una etapa de tiempo $ts(T_i)$ cuando inicia.

Características de una estampa de tiempo:

- ✓ Identifica la transacción de manera única.
- ✓ Son valores derivados de un dominio totalmente ordenado.
- ✓ Monoticidad: Son dos estampas de tiempo generadas por el mismo administrador de transacciones.

Una forma adecuada para asignar estampas de tiempo, es usando un controlador local, donde cada nodo le asigna al controlador su identificador, de esta manera la estampa de tiempo será un par de la forma siguiente:

<contador local, identificador de nodo>

Cuando el administrador de transacciones le asigna una estampa de tiempo a las operaciones pedidas por una transacción o a cada elemento de dato *x* se le asigna una *estampa de tiempo de escritura*, *wts*(*x*), y *de lectura*, *rts*(*x*); sus valores indican la estampa de tiempo más grande para cualquier lectura y escritura de *x*.

El ordenamiento de estampas de tiempo (TO) es generado por la regla que se describen a continuación:

> **Regla TO:** Dada dos operaciones en conflicto, O_{ij} y O_{kl}, perteneciendo a las transacciones T_i y T_k, respectivamente, O_{ij} es ejecutada antes de O_{kl}, si y solamente si, $ts(T_i) < ts(T_k)$. En este caso T_i se dice ser una transacción *más vieja* y T_k se dice ser una transacción *más joven*.

En caso de que se rechace una acción de operación, se dice que la transacción que la envió necesita reiniciarse para tener la estampa de tiempo más actual del dato e intentar de nuevo la operación sobre dicho dato.

Ordenamiento conservador por estampas de tiempo: Este ordenamiento retrasa la operación hasta que sea segura de que no será reiniciada. El retraso de la operación puede producir un interbloqueos (*deadlocks*).

Ordenamiento por estampas de tiempo múltiples: Con este ordenamiento se pueden crear copias múltiples únicas de cada dato, usando las siguientes estrategias:

> *Estrategia 1:* Una operación de lectura $R_i(x)$ se traduce a una operación de lectura de *x* de una sola versión encontrando la versión de *x*, (x_v), tal que, $ts(x_v)$ es la estampa de tiempo más grande que tiene un valor menor a $ts(T_i)$.

Estrategia 2: Una operación de escritura $W_i(x)$ se traduce en una sola versión, $W_i(x_w)$, y es aceptada si el despachador no ha procesado cualquier lectura $R_j(x_r)$, tal que, $ts(T_i) < ts(x_r) < ts(T_j)$. **[F]**

1.10.6. Control de concurrencia optimista

Este tipo de algoritmo retrasa la fase de validación antes de la fase de escritura, solo así se logra que la operación realizada por un despachador optimista no se retrase.

Las operaciones de lectura, cómputo y escrita de las transacciones se realizan sin actualizar la base de datos. Las transacciones primero realizan sus cambios en copias locales de los datos, donde son validadas (corroborará si las actualizaciones conservan la consistencia de la base de datos), si son correctas se harán cambios globales, en caso contrario las transacciones serán abortadas y tendrán que reiniciarse. Cabe mencionar que las operaciones de una transacción siguen la secuencia de las fases de: validación (V), lectura (R), cómputo (C) y escritura (W).

Las estampas de tiempo se asignan al inicio de la fase de validación de una transacción. Cada transacción T_i se subdivide en varias subtransacciones, las cuáles se puede ejecutar en nodos diferentes.

Para hacer una validación, se hacer uso de una de las reglas, que a continuación se mencionan:

Regla 1.- Si todas las transacciones T_k, tales que, $ts(T_k) < ts(T_{ij})$, han terminado su fase de escritura antes que T_{ij} ha iniciado su fase de lectura entonces la validación tiene éxito. En este caso la ejecución de las transacciones es completamente serial (ver Figura 1.18).

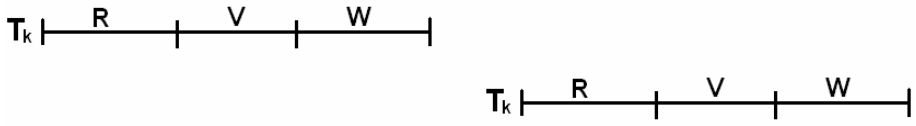


Figura 1.18: Primer regla de validación para control de concurrencia optimista. **[F]**

Regla 2.- Si existe alguna transacción T_k, tal que, $ts(T_k) < ts(T_{ij})$ la cuál completa su fase de escritura mientras T_{ij} está en su fase de lectura, entonces, la validación tiene éxito si $WS(T_k) \cap RS(T_{ij}) = \varnothing$. En este caso, las fases de lectura y escritura se traslapan (ver Figura 1.19), pero T_{ij} no lee datos escritos por T_k.

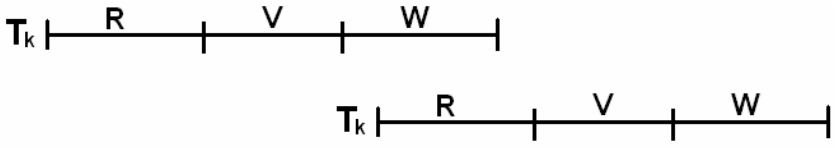


Figura 1.19: Segunda regla de validación para control de concurrencia optimista. **[F]**

Regla 3.- Si existe alguna transacción T_k, tal qué, $ts(T_k) < ts(T_{ij})$ y completa su fase de lectura antes que T_{ij} terminé su fase de lectura, entonces, la validación tiene éxito si $WS(T_k) \cap RS(T_{ij}) = \varnothing$ y $WS(T_k) \cap WS(T_{ij}) = \varnothing$. En este caso, las fases de lectura se traslapan, pero las transacciones no acceden a datos comunes (ver Figura 1.20). **[F]**

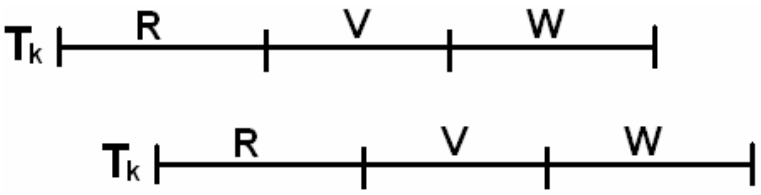


Figura 1.20: Tercer regla de validación para control de concurrencia optimista. **[F]**

Capítulo

2

Sistemas de bases de datos distribuidas

Los sistemas de bases de datos distribuidas están diseñados para gestionar grandes volúmenes de información. Generalmente, las bases de datos requieren gran cantidad de espacio de almacenamiento, por lo qué las bases de datos de las empresas o instituciones se miden en términos de Gigabytes o Terabytes de datos. Por esta razón en el presenté capítulo se hace mención de las principales características de los Sistemas de Bases de Datos Distribuidas (SBDD), Sistemas Gestores de Bases de Datos (SGBD) “en ingles Database Management System” y los Sistemas Distribuidos. Entre las características que se mencionarán están los conceptos, evolución, componentes, ventajas, desventajas, entre otras. Para finalizar con el capítulo se menciona un pequeño tema que habla en donde se pueden aplicar los sistemas de bases de datos distribuidos.

2.1. Sistemas de bases de datos distribuidas (SBDD)

Los sistemas de bases de datos distribuidas se crean para acumular grandes cantidades de información que se necesiten acceder desde distintos lugares. La gestión de los datos involucra la definición de la estructura para guardar la información como el abastecimiento de dispositivos para el manejo de la información. Los sistemas de bases de datos distribuidos deben aportar seguridad a la información almacenada, en caso de alguna falla en el sistema o intentos de accesos no autorizado al mismo; éste debe impedir posibles consecuencias negativas. **[G]**

2.1.1. Concepto de sistema de bases de datos distribuidas

Un sistema de bases de datos distribuidas, es un sistema el cuál esta formado por muchos sistemas de bases de datos unidos por un sistema de comunicación, para cuando el usuario quiera acceder desde cualquier lugar éste lo pueda hacer como si los datos estuvieran almacenados en el lugar desde donde quiera tener acceso.

También se puede decir; que un sistema de bases de datos distribuidas esta compuesta por varias bases de datos vinculadas lógicamente entre sí; éstas están asignadas en distintos locales unidas por una red. **[9]**

2.2. Especificaciones de los sistemas gestores de bases de datos (SGBD)

Los sistemas gestores de bases de datos son la parte importante del software de un sistema de base de datos. Por esta razón el sistema gestor de base de datos es software definido; con el propósito de servir como interfaz entre la base de datos y los usuarios que la deseen utilizar.

Por otra parte cabe mencionar que los términos de Sistema Gestor de Base de datos (SGBD) y DataBase Management System (DBMS, término en ingles) son equivalentemente e iguales. **[10]**

2.2.1. Evolución de los SGBD

A principios de los años 40´s, los sistemas de archivos creados con los lenguajes de programación no propietarios como Cobol y Fortran, permitían almacenar y acumular datos usando archivos planos con funciones básicas como lectura y escritura. En 1964, se crean los primeros sistemas gestores de base de datos (SGBD), con los cuales se intenta dar un desvío a los sistemas de archivos, los cuales se restringen solo a la estructuración del almacenamiento físico de los datos.

Con los DBMS o SGBD se crea el concepto de administración de los datos, por medio de acciones integradas que permitan verlos físicamente en un solo almacenamiento, pero lógicamente se manipulan a través de esquemas mezclados por estructuras en el que se establecen vínculos de integridad, métodos de acceso

y organización física referente a los datos, admitiendo obtener valores adicionados como: administración de usuarios, seguridad, entre otros.

En 1965 se crea el primer sistema gestor de base de datos comercial, Integrated Data Store (IDS), bajo el concepto de Modelo de Datos de Red (Bachgman), después de este sistema se crea el Information Management System (IMS), bajo el concepto de Modelo de Datos Jerárquico. Los primeros SGBD eran accedidos mediante lenguajes de programación como Cobol empleando interfaces de bajo nivel haciendo que la creación de aplicaciones y mantenimiento de los datos fueran controlables, pero a pesar de esto, aún eran complejos.

Conforme evolucionaron los SGBD, los lenguajes de programación también lo hacían. En el año de 1967 nace Simula, el primer lenguaje de programación orientado a objetos, el cuál fue planteado para la simulación de actividades. En Simula los procedimientos eran asociados a un tipo, para representar el comportamiento de una instancia, introduciendo la palabra *Clase.*

En la década de los 80´s nacen muchos SGBD Relacionales (SGBDR) o Relational Database Management System (RDBMS), por ejemplo Oracle, Informix y DB2. Esto conllevo a que nacieran más lenguajes de programación, como C++, Java y Eiffel. Los lenguajes de programación adoptaron y pulieron el concepto de Clase, tal desarrollo era independiente de los SGBD.

Cuando inician los años 80´s, nace la necesidad de que los SGBD o SMBD puedan manipular objetos y estructuras complejas. Por tal motivo nacen dos tendencias las cuáles son: Object Relational Database Management System (ORDBMS) los cuáles se proyectan como una extensión de los RDBMS y a finales de los 80´s se crean los Object Oriented Database Management System (OODBMS), los cuales estarían para almacenar y manipular las clases, los objetos y la asociación entre ellos.

En 1991 nace la Object Database Management Group (ODMG), el cuál se encarga de estandarizar los OODBMS´s, un año mas tarde, 1992 el comité ANSI X3H2 comienza a trabajar con SQL.3, dando como resultado los DBMS objeto relacional ORDBMS. **[11]**

2.2.2. Sistema gestor de base de datos (SGBD)

Un sistema gestor de base de datos se define como un ***"conjunto coordinado de programas, procedimientos, lenguajes, etc. que suministran a los distintos tipos de usuarios los medios necesarios para describir y manipular los datos almacenados en la base de datos, garantizando su seguridad***". **[A]**

El SGBD se dice que es el software que interactúa entre el usuario y la base de datos. Gracias a este software el usuario y las aplicaciones no acceden a los datos tal y como son físicamente, sino que se ven representadas lógicamente. Esto permite que exista independencia entre aplicaciones y la forma física de los datos,

permitiendo que un administrador de base de datos pueda cambiar la forma física de los datos, e incluso puede cambiar de SGBD, pero no la forma lógica conocida como "esquema de la base de datos, siendo la forma de representar los datos" ya que es la única parte que conocen los usuarios.

En pocas palabras se dice que el objetivo principal de los SGBD, es proporcionar un entorno amigable y eficiente para los usuarios que lo usan; así como recuperar y almacenar la información. **[G]**

La estructura básica de un sistema gestor de base de datos esta formada por el lenguaje de definición de datos (DDL) o Data Definition Language y el lenguaje de manipulación de datos (DML) o Data Manipulation Language. **[11]**

2.2.3. Componentes de los SGBD

Los componentes de los SGBD se encargan de la definición, actualización y recuperación de datos estructurados. Por tal razón se mencionan a continuación:

Datos: Son los que se encuentran almacenados en la base de datos o diferentes bases de datos. Por está razón la base de datos funciona como un acumulador de datos estando integrada y compartida:

Integrada: Es cuando la base de datos esta unida a diferentes archivos de datos independientes donde se excluye parcial o totalmente la redundancia entre archivos.

Compartida: Son las partes o archivos individuales de la base de datos que se comparten entre varios usuarios, teniendo acceso a ellos, dándoles un uso distinto. Estos permisos de comunicación a diferentes archivos se logran gracias a los programas de administración de la red.

Software: Entre el almacenamiento físico de datos y los usuarios coexiste el software que adopta el nombre de SGBD *(Software que sirve como interfaz entre la base de datos y el usuario)*. Este SGBD tiene como objetivo manipular las solicitudes formuladas por los usuarios y programas conforme a las funciones del SGBD, entre otros objetivos del SGBD esta, la ayuda hacia los usuarios contra detalles a nivel de hardware; es decir, cuando el usuario realiza una consulta a las base de datos, el SGBD va a la tabla de la misma para buscarla y después mostrarla.

Hardware: Se encuentra formado por discos duros, tambores, etc. de grandes capacidades para el almacenamiento de los datos; también se encuentran los dispositivos que permiten controlar y compartir los datos.

Usuarios: Los usuarios son una de las partes importantes de los componentes de los SGBD; ya que son los que interactuaran con los datos de la base o bases de datos, entre los cuales se mencionan tres tipos de usuario:

Usuario final: Este es el usuario que realiza consultas, mantenimiento, entre otras acciones a la base de datos a través de ciertos programas.
Programador de aplicaciones: Usuario que se encarga del desarrollo de los sistemas necesarios, para que los usuarios finales se puedan comunicar con la base de datos.

Administrador de la base de datos: Usuario encargado de mantener en forma óptima y eficiente la base de datos, controlando las instalaciones, procedimientos procesos, etc.; entre sus funciones, se encarga de hacer auditorias, crear accesos y crear usuarios; así como la seguridad de la base de datos. **[12]**

2.2.4. Objetivos de los SGBD

Cuando se habla de SGBD surge la gran pregunta "¿cuáles son los objetivos principales de los SGBD que deben de cumplir para que tengan una eficiencia y eficacia razonable?; por tal motivo se plasman los siguientes:

Abstracción de la información: Los usuarios de los SGBD ahorran a los usuarios finales detalles acerca del almacenamiento físico de los datos. Da lo mismo sí una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario; así, se definen varios niveles de abstracción.

Independencia de datos: Consiste en poder modificar el esquema (físico o lógico) de la base de datos sin tener que efectuar cambios en las aplicaciones que dependan de ella.

Redundancia: Es cuando se logra evitar información repetida en la base de datos. Para que no exista redundancia se debe de realizar un buen diseño de la base de datos; aunque en ocasiones no se puede eliminar por completo la redundancia, pero sí se puede reducir.

Consistencia: Se habla de consistencia cuando la información repetida existente se actualizará de forma simultánea.

Seguridad: Se da seguridad a la información almacenada en cierta base de datos que tenga mucho valor para los propietarios de la misma. Por tal motivo los SGBD les corresponde garantizar que esta información este segura ante los usuarios que quieran poseer, destruir o manipular esta información, o simplemente ante el descuido de un usuario autorizado.

Respaldo y recuperación: Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia: El control de la concurrencia existe cuando varios usuarios acceden a los datos almacenados en la base de datos, para recuperar o

consultar información. Por tal motivo los SGBD controla los accesos que ocurran simultáneamente evitando inconsistencia en la base de datos.

Integridad: Es cuando se toman las medidas necesarias para garantizar la validez de los datos almacenados en la base de datos. En pocas palabras se dice que se protegen los datos ante fallos de hardware, datos introducidos por usuarios distraídos o cualquier otro problema que dañe la información.

Tiempo de respuesta: Surge cuando el SGBD minimiza el tiempo en dar una respuesta acerca de la información consultada y en almacenar los cambios que se efectúan en la base de datos.

2.2.5. Ventajas de los sistemas gestores de bases de datos distribuidas

Los SGBDD poseen distintas ventajas, como las siguientes:

- ✓ Los datos se almacenan en lugares cercanos, provocando un acceso rápido.
- ✓ El procesamiento de datos se acelera, ya que participan muchos nodos en una carga de trabajo.
- ✓ Se facilita agregar nodos rápidamente.
- ✓ La comunicación entre nodos se mejora.
- ✓ Los costos de operación se reducen.
- ✓ Son amigables con el usuario.
- ✓ La falla en un nodo es difícil que afecte al sistema.
- ✓ Existe una autonomía e independencia entre los nodos.

Estos motivos conllevan a que compañías y negocios adopten bases de datos distribuidas, ya que incluye motivos organizacionales y económicos, para tener una comunicación segura y flexible con las diferentes bases de datos. **[9]**

2.2.6. Inconvenientes de los SGBD

Cuando se tiene un SGBD que ofrecen grandes ventajas para el manejo de información desde diferentes lugares o distancias, también tiene más inconvenientes para que se lleguen a implementar en las diferentes organizaciones; las cuáles se mencionan a continuación:

Complejidad: Los SGBD están formados por varios programas complejos con grandes funciones. Es necesario entender estas funciones para aprovecharlas.

Tamaño: Los SGBD son programas complicados y extensos los cuáles necesitan grandes cantidades de espacio en disco duro y memoria RAM para poder procesar los datos de manera eficiente.

Costo de la conversión: Este tipo de costo contiene el costo de enseñar a la plantilla a utilizar estos sistemas y el costo del personal especializado para hacer la

conversión y poner en marcha el sistema. Este costo es una de las razones por las cuáles no todas las empresas adquieren un sistema de base de datos.

Costo económico: Los costos de un SGBD, varían dependiendo del entorno y la funcionalidad.

Prestaciones: Las prestaciones es uno de los inconvenientes mas grandes de los SGBD ante los sistemas de ficheros, estos están escritos en una aplicación especifica por tal razón son muy buenos; sin embargo, los SGBD son mas generales y útiles en muchas aplicaciones, lo que puede ocasionar lentitud en alguna de ellas.

Vulnerable a fallos: Son ocasionados por tener los datos centralizados en el SGBD. **[13]**

2.2.7. Funciones de los SGBD

Los SGBD puede tener varias funciones, pero solo existen tres funciones fundamentales, las cuáles se explican a continuación:

Función de Definición de datos: En esta función se habla del Lenguaje de Definición de Datos o Data Definition Language (DDL), el cuál permite al diseñador de la base de datos definir los componentes de datos que la forman; así como su estructura y la relación que existe entre ellos, las reglas de integridad semántica, etc., tanto las características de tipo físico y las vistas lógicas que verán los usuarios.

El DDL del SGBD tiene que proveer los medios para definir las estructuras, acorde con el modelo de arquitectura de tres niveles, habrá un lenguaje de definición de la estructura lógica global, otro para la definición de la estructura interna, y un tercero para la definición de las estructuras externas; las cuáles se mencionan en seguida:

Estructura interna: indica el tipo de espacio (volúmenes, cilindros y pistas) apartado para la base de datos, la longitud de los campos, su modo de representación (binario, decimal, alfanumérico, punto flotante, etc.). También se pueden definir caminos de acceso, como punteros, índices, entre otros.

Estructura externa: Se encarga de dar los componentes para la definición de los objetos (entidades, tablas, etc.); así como su identificación, atributos y las interrelaciones que existan entre ellos, autorizaciones de accesos, restricciones de integridad, entre otros.

Estructura lógica global: Tiene encomendadas las descripciones de las estructuras lógicas de los usuarios. **[A]**

Función de manipulación de datos: Aquí se menciona el Lenguaje de Manipulación de Datos o Data Manipulation Language (DML), encargado de las solicitudes realizadas por los usuarios como inserción, borrado, modificado de los

datos en la base de datos. También se puede mencionar que es el encargado de facilitar a los usuarios el acceso y manipulación de los datos. **[12]**

Inserción: Cuando se requieran o se tengan nuevos elementos; será necesario dar de alta a los nuevos usuarios en la base de datos.

Borrado: Esta opción es aplicada cuando se tienen elementos registrados en la base de datos que se desean dar de baja.

Modificado: Existe cuando los datos almacenados en las base de datos han sufrido cambios.

En la figura 2.1, se representa la realización de solicitudes que hace el usuario con la base de datos.

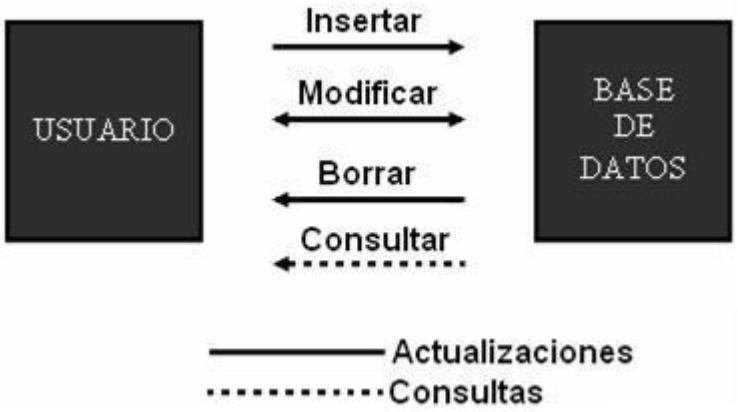


Figura 2.1: Interacción usuario/base de datos. **[A]**

Función de control: Esta funciona junto con las interfaces necesarias que permiten comunicar al usuario con la base de datos, proporcionando un conjunto de procedimientos para el administrador de la base de datos. **[A]**

Entre otras funciones de los SGBD también se pueden mencionar las siguientes:

- ✓ Manejar los datos de acuerdo a las peticiones de los usuarios.
- ✓ Registrar el uso de las bases de datos.
- ✓ Interacción con el manejador de archivos; a través, de las sentencias DML, de esta forma el SGBD se encarga del verdadero almacenamiento de los datos.
- ✓ Respaldo y recuperación; se logra mediante elementos que permitan la recuperación rápida y fácil de los datos, en cuanto surja un fallo en el sistema de base de datos.
- ✓ Control de concurrencia: Es el control de acceso de usuarios para no afectar la inconsistencia de los datos.
- ✓ Seguridad e integridad: Está en contar con mecanismos que permitan el control de la consistencia de los datos evitando que sean afectados por cambios no autorizados o previstos. **[10]**

2.2.8. Servicios que ofrecen los SGBD

El creador del modelo relacional "*E. F. Codd*", ha determinado los siguientes servicios que debe ofrecer todo SGBD:

1.- El SGBD tiene como función principal facilitar a los usuarios la capacidad de almacenamiento, acceso y actualización de datos. También se encarga de ocultar la estructura física de los datos, mostrando al usuario una vista amigable.

2.- Un SGBD da un catálogo en el que se acumulan las descripciones de los datos donde los usuarios puedan acceder. El catálogo es también llamado diccionario de datos el cuál contiene información que describe los datos de la base de datos, como:

- ✓ Nombre, tipo y tamaño de los datos.
- ✓ Nombre de las relaciones entre los datos.
- ✓ Restricciones de integridad sobre los datos.
- ✓ Nombre de los usuarios autorizados a acceder a la base de datos.
- ✓ Esquemas externos, conceptuales e internos, y correspondencia entre los esquemas.

Cuando se habla de diccionario de datos, se puede mencionar algunas ventajas que proporcionan; por ejemplo:

- ✓ La información sobre los datos se puede almacenar de un modo centralizado. Esto facilita el control sobre los datos.
- ✓ El significado de los datos se definen, lo que ayudará a los usuarios a entender el propósito de los mismos.
- ✓ La comunicación se simplifica ya que se almacena el significado exacto. El diccionario de datos también puede identificar al usuario o usuarios que poseen los datos o que los acceden.
- ✓ Se identifican fácilmente las redundancias y las inconsistencias.
- ✓ Tiene la historial de los cambios realizados sobre la base de datos.
- ✓ El impacto que produzca un cambio se puede determinar antes de que sea implementado.
- ✓ Se respeta la seguridad.
- ✓ Garantiza la integridad.
- ✓ Puede facilitar información para auditorias.

3.- El SGBD provee un componente que garantiza que las actualizaciones que se hagan a una transacción se puedan realizar o no.

4.- Facilita un mecanismo que asegura que la base de datos se actualice correctamente cuando varios usuarios la están actualizando al mismo tiempo.

5.- Un SGBD debe proporcionar los medios necesarios para garantizar que tanto los datos de la base de datos, como los cambios que se realizan sobre estos datos,

sigan ciertas reglas. La integridad de la base de datos requiere la validez y consistencia de los datos almacenados.

6.- Aporta un componente que garantiza que sólo los usuarios autorizados pueden acceder a los datos; tal seguridad debe ser contra accesos no autorizados ni accidentales.

7.- Los SGBD se tienen que poder integrar con software de comunicación. Para qué los usuarios puedan acceder a la base de datos desde distintos lugares.

8.- Proporciona mecanismos para poder recuperar la base de datos en caso de que ocurra un fallo, llevándolas a un estado consistente. **[14]**

9.- El SGBD da herramientas que permitan administrar la base de datos de modo efectivo. Algunas herramientas trabajan a nivel externo; por lo que habrán sido producidas por el administrador de la base de datos. Las herramientas que trabajan a nivel interno deben ser proporcionadas por el distribuidor del SGBD, como:

- ✓ Herramientas para importar y exportar datos.
- ✓ Herramientas para monitorear el uso y el funcionamiento de la base de datos.
- ✓ Herramientas para reorganización de índices.

2.2.9. Tipos de SGBD

La estructura y arquitectura de los sistemas de bases de datos; están influidas por los sistemas de telecomunicaciones que toleran la instalación de los SGBD, lo que permite que existan distintos tipos de SGBD.

2.2.9.1. SGBD centralizados

Estos sistemas son los que se ejecutan en un solo sistema de computadora; sin interactuar con otro u otros sistemas, teniendo como rango desde los sistemas de bases de datos monousuarios que corren en computadoras personales hasta los sistemas de bases de datos que se ejecutan en sistemas de alto rendimiento.

Básicamente los sistemas monousuarios no proporcionan todas las facilidades que los sistemas multiusuarios, especialmente este tipo de sistemas no tienen control de concurrencia ni sistemas de recuperación, en caso de que los tengan estos serán inestables.

Regularmente cuando se ejecutan los sistemas monousuarios, se usan computadoras de propósito general, teniendo una arquitectura de uno a dos procesadores que comparten la memoria principal, por esta razón se dice que los sistemas de base de datos se ejecutan mas rápido ya que cada procesador ejecuta una consulta por separado, esto ocasiona que se paralicen las tares de consultas. **[9]**

2.2.9.2. SGBD cliente-servidor

Con la evolución de las computadoras personales (PC) y las redes de área local (LAN), se han ido desplazando hacia el lado del cliente la funcionalidad de la parte visible al usuario de la base de datos (interfaces de formularios, gestión de informes) de tal forma que los sistemas servidores proporcionen la parte de accesos a las estructuras de datos, evaluación y procesamiento de consultas, control de concurrencia y recuperación. Estos sistemas de servidores se dividen en dos tipos los cuáles se mencionan a continuación:

Servidores transaccionales: Encargado de agrupar la lógica del negocio en un servicio aparte, provee una interfaz con la cuál el usuario podrá realizar sus peticiones.

Servidores de datos: Servidores encargados de enviar datos a bajo nivel y descansan en la capacidad de procesamiento de datos en la base de datos de las máquinas clientes.

En la construcción de motores de base de datos cliente-servidor existe la arquitectura motores multiprocesos y motores multihilos. **[9]**

2.2.9.2.1. Motores de base de datos multiprocesos

En esta arquitectura, cada vez que un usuario se conecta a la base de datos, está inicia una instancia de la aplicación de la base de datos, para poder coordinar varios usuarios que acceden a los mismos datos. La comunicación que existe entre aplicaciones se hacen mediante un sistema propietario de comunicaciones ínterproceso (IPC). Un ejemplo de este motor de base de datos multiprocesos es Oracle Server, el cuál carga 16 tipos de ejecutables diferentes que realizan distintas tareas.

En el ambiente de multitarea el sistema operativo divide el tiempo de procesamiento entre múltiples aplicaciones concediéndoles una fracción de tiempo de CPU a cada una. De esta manera siempre habrá una tarea ejecutándose a la vez; pero el resultado es que múltiples aplicaciones aparentan estar ejecutándose a la vez en un CPU. Seria más grande la ventaja cuando el sistema operativo cuente con más de un CPU. **[9]**

2.2.9.2.2. Motores de base de datos multihilos

Los motores de base de datos multihilos tratan el problema de varios accesos simultáneamente, hechas por los usuarios de manera distinta que los motores de base de datos multiprocesos, pero usando principios similares. En lugar de dejar que el sistema operativo comparta los recursos de procesamiento, el motor de base de datos toma la responsabilidad, esto da origen a una mejor portabilidad del sistema. Un ejemplo de este tipo de motor, es Microsoft Sql Server.

Algunas de las ventajas importantes que tiene el motor de base de datos multihilos ante el motor de base de datos multiprocesos, es que éste maneja una cantidad menor de memoria RAM por conexión; otra de las ventajas es que no es necesario un mecanismo de comunicación ínterproceso. De esta forma la base de datos usa el elemento finito (hilo) de trabajo, para las operaciones (instrucciones de usuarios, bloqueos de datos, E/S de disco, administración del caché, etc.) en lugar de usar aplicaciones especializadas para cada tarea.

Desventajas de los motores multihilos:

Escalabilidad: Depende de la capacidad que tengan los diferentes motores de bases de datos multihilos de deshacer una operación para que varias tareas puedan ejecutar esta operación.

Portabilidad: Este problema esta relacionado con el Multiprocesamiento Simétrico (SMP), dado que existen distintos fabricantes que le dan soporte de diferente forma, los motores de bases de datos han buscado la manera de implementar técnicas neutrales para que tengan un buen funcionamiento en la implementación física; lo que ocasiona sobrecarga en el motor y una limitación en la habilidad de escalar a un gran número de procesadores. **[9]**

2.2.9.3. SGBD paralelos

Estos sistemas de base de datos están formados por diversos procesadores y discos interconectados mediante una red de alta velocidad. Para medir el rendimiento de estos sistemas de bases de datos existen dos medidas, las cuales se mencionan a continuación:

Productividad (throughput): Número de tareas que se completan en un tiempo determinado.

Tiempo de respuesta (response time): Tiempo que se necesita para completar una tarea a partir del momento en que se inicia.

Un sistema que procesa muchas transacciones pequeñas puede mejorar su productividad efectuando numerosas transacciones en paralelo y un sistema que procesa transacciones más grandes logra optimizar su productividad y sus tiempos de respuesta ejecutando en paralelo sus subtareas de cada transacción.

Las ventajas en este tipo de SGBD se da en:

- ✓ Velocidad: Tiempo mínimo en ejecutarse determinada tarea.
- ✓ Ampliabilidad: Capacidad de procesar tareas mas grandes en el mismo tiempo. **[9]**

2.2.9.4. SGBD distribuidos

En los SGBD distribuido, la base de datos se almacena en varias computadoras permitiendo comunicarse por diferentes medios de comunicación como: redes de alta velocidad y líneas telefónicas. Estos sistemas no comparten memoria RAM ni discos de almacenamiento y sus tamaños varían tanto como sus funciones pueden incluir desde computadoras personales hasta grandes sistemas.

Las computadoras que forman parte del sistema distribuido se les denomina nodos o locales, los cuáles se encuentran en diferentes lugares geográficos y se administran de forma separada, estos tienen una conexión lenta. En estos sistemas se dan dos tipos de transacciones:

Transacción local: Solo permite acceder a los datos que se almacenen en el nodo en la cuál se realiza la transacción.

Transacción global: Permite acceder a los datos que se encuentren en un nodo distinto en donde se realiza la transacción; también pueden acceder a datos de distintos nodos.

Este tipo de sistemas se les conoce por:

- ✓ Los distintos emplazamientos están informados de los demás.
- ✓ Aunque algunas relaciones pueden estar almacenadas, sólo en algunos emplazamientos, éstos comparten un esquema global común.
- ✓ Cada emplazamiento proporciona un entorno para la ejecución de transacciones tanto locales como globales. **[9]**

2.2.9.5. Clasificación de SGBD

Conforme se fueron desarrollando los SGBD, surgieron distintas distribuciones de estos, por esta causa se clasificaron de dos formas:

SGBD libres: Aquellos que al ser usados no se paga alguna cantidad monetaria.

- ✓ PostgreSQL (http://www.postgresql.org Postgresql) Licencia GPL
- ✓ MySQL Licencia Dual, depende el uso.

SGBD comerciales: Para poder usar estos, se tienen que pagar una cierta cantidad monetaria a su o sus creadores:

- ✓ dBase
- ✓ Fox Pro
- ✓ Informix
- ✓ MAGIC
- ✓ Microsoft Access
- ✓ Microsoft SQL Server
- ✓ MySQL
- ✓ Oracle

- ✓ Paradox
- ✓ PervasiveSQL. **[15]**

2.3. Sistemas distribuidos

El desarrollo global requiere de sistemas y aplicaciones informáticas, prestaciones que van más allá de lo alcanzable por los ordenadores aislados, no importando lo potente que sean. Este gran motivo conlleva a que los sistemas distribuidos se conviertan en un modelo general de avances tecnológicos, desde las más extendidas arquitecturas cliente-servidor e incluso los sistemas en cluster que brindan la imagen de una máquina; así los sistemas distribuidos tienen muchas mas ventajas que los sistemas centralizados respecto a su rendimiento, fiabilidad, escalabilidad y efectividad de costo. **[16]**

2.3.1. Concepto de sistema distribuido

Se dice que es un conjunto de recursos informáticos de propósito general; tanto físico como lógico, que se le pueden asignar tareas concretas. Los recursos se encuentran repartidos físicamente, y enlazados por una red de comunicaciones. Existe un sistema operativo con grandes capacidades que une el control de los dispositivos. Esta distribución es transparente facilitando que los servicios puedan ser solicitados especificando su nombre y no su localización. **[17]**

También se dice que un sistema distribuido es cualquier sistema multiprocesador en el que cada procesador posee una memoria local. La forma de comunicación entre procesadores es por medio de envió de mensajes mediante una red de comunicación. Proporcionando un interfaz de alto nivel.

Los sistemas distribuidos se basan en ideas de transparencia, eficiencia, flexibilidad, escalabilidad y fiabilidad. Pero estos conceptos pueden ser contrarios, a menos que se logren solucionar de forma aceptable:

Transparencia: Se refiere a que todo el sistema funcione de forma similar en todos los nodos unidos por la red, no importando la localización de los usuarios. Por otra parte el sistema operativo esta encargado de crear los dispositivos que no dejen ver la naturaleza distribuida del sistema; así como dejar trabajar a los usuarios, como sí se tratara de un solo equipo. Igualmente el sistema operativo se encarga de llevar el control de las copias, actualizaciones, modificaciones y la unificación de recursos, y el control de la concurrencia

Cuando el sistema tenga varios procesadores, este debe tener mayor rendimiento, pero el sistema operativo le corresponde controlar a los usuarios y que los programadores distingan el núcleo del sistema distribuido como un solo procesador. También el sistema operativo debe de controlar el paralelismo que distribuye las tareas entre los diferentes procesadores, resultando más eficientes y flexibles.

Eficiencia: Se obtiene con la ejecución de los programas en diferentes procesadores, distribuyendo las tares en procesadores que estén libres, esto hará que los procesos se agilicen.

Flexibilidad: La flexibilidad surge cuando un sistema distribuido se diseña para aceptar cambios y actualizaciones que mejoren el sistema. **[18]**

2.3.2. Características de los sistemas distribuidos

- ✓ Todo componente de cómputo tiene su propia memoria y sistema operativo.
- ✓ Control de recursos locales y remotos.
- ✓ Sistemas abiertos a cambio y crecimiento.
- ✓ Plataforma no standard (Unix, NT, Intel, RISC, entre otros).
- ✓ Medios de comunicación (redes, protocolos, dispositivos, etc.).
- ✓ Capacidad de procesamiento en paralelo.
- ✓ Dispersión y parcialidad.

2.3.3. Factores que han afectado el desarrollo de los sistemas distribuidos

- ✓ Avances tecnológicos.
- ✓ Nuevos requerimientos.
- ✓ Globalización.
- ✓ Aspectos externos como políticos y económicos.
- ✓ Integración.

2.3.4. Ventajas

- ✓ Procesadores más eficaces y a menos costos.
- ✓ Desarrollo de estaciones con más capacidades.
- ✓ Las estaciones satisfacen las necesidades de los usuarios.
- ✓ Uso de nuevas interfaces.
- ✓ Avances en la tecnología de comunicaciones.
- ✓ Disponibilidad de comunicación.
- ✓ Desarrollo de nuevas técnicas.
- ✓ Compartición de recursos.
- ✓ Dispositivos de hardware.
- ✓ Programas o software.
- ✓ Eficiencia y flexibilidad.
- ✓ Respuesta rápida.
- ✓ Ejecución concurrente de procesos en varias computadoras.
- ✓ Técnicas de procesamiento distribuido.
- ✓ Disponibilidad y confiabilidad.
- ✓ Sistema poco propenso a fallas.
- ✓ Introducción rápida de nuevos recursos.
- ✓ Los recursos actuales no los afectan. **[H]**

2.3.5. Desventajas

- ✓ Requerimientos de mayores controles de procesamiento.
- ✓ Velocidad de propagación de información muy lenta a veces.
- ✓ Servicios de replicación de datos y servicios con posibilidades de fallas.
- ✓ Mayores controles de acceso y proceso.
- ✓ Administración compleja.
- ✓ Costos elevados.
- ✓ Diseños complejos. **[H]**

2.4. Aplicaciones de SBDD

La distribución de las bases de datos son aplicables a las organizaciones; por las necesidades del crecimiento global, de esta manera las organizaciones tienen un crecimiento de departamentos. Con los sistemas distribuidos se podrán poner los datos, para que los usuarios puedan acceder y tener un control sobre estos.

Los lugares donde regularmente se encuentra funcionando las bases de datos distribuidas, son:

- ✓ Cualquier organización que tiene una estructura descentralizada.
- ✓ Organismos gubernamentales y de servicio público.
- ✓ La industria de la manufactura.
- ✓ Líneas de transportación aérea (aeropuertos).
- ✓ Cadenas hoteleras.
- ✓ Servicios bancarios y financieros.
- ✓ Escuelas.
- ✓ En el ejército. **[9]**

Capítulo

El software libre y la distribución Debian

Los sistemas GNU/Linux han llegado a un grado de madurez importante, lo que ha hecho que se usen en varios ambientes de trabajo, ya sea desde el escritorio de una PC personal, hasta el servidor de una gran empresa. La distribución que se selecciona para este documento es Debian Woody 3.0, debido a que esta no pertenece a ninguna empresa y esta formada por las aportaciones de los voluntarios distribuidos por todo el mundo. Esto quiere decir; que Debian es elegido por ser 100% libre.

Los objetivos principales de este capítulo serán, dar una pequeña redacción de cómo surgió el proyecto GNU; así como las licencias de software y documentación libre compatible e incompatible con la GNU GPL (Licencia Publica General). Entre otras cosas, se hará mención de las características principales de la distribución Debian y la instalación del sistema operativo Debian GNU/Linux.

3.1. Software libre

El proyecto GNU “ñu” (ver Figura 3.1), nace en 1983 como una forma de devolver el espíritu cooperativo que predominaba en la comunidad computacional en días pasados, de esta forma el cooperativismo tenia como objetivo eliminar las barreras que los dueños del software privado habían implantado.

Figura 3.1: Logotipo de software libre (GNU) “ñu”. **[19]**

En el año de 1984, Richard Stallman *(nació el 16 de marzo de 1953 en Manhattan, Nueva York)*, empezó a trabajar con el proyecto GNU, un año después de incorporarse a este proyecto funda la Free Software Foundation (FSF). Por esta razón Richard Stallman implanta la definición *free software* y el concepto “copyleft”, el cuál introdujo para que los usuarios tuvieran la libertad de desarrollar software y evitar la apropiación del mismo.

Con el “copyleft”, surgen licencias para este tipo de software libre, para que sea publicada su libre redistribución, pero conservando los derechos de autor. El término *software libre* se usa para referirse al software distribuido bajo ciertas licencias que no garantizan todas las libertades del software. Las leyes de *propiedad intelectual* por lo general reservan los derechos de modificación, duplicación y redistribución para el propietario del “copyright”.

Por otra parte, las personas usan los términos “libre” *(Software libre)* y ”gratis” (Software gratis), para no ocasionar confusión con el término ingles “free”. De esta forma se dice que estos términos se usan solo en el *movimiento del software libré*, además existen personas que defienden el término open source software (software de código abierto), existiendo diferencias entre estos dos términos, por un lado “free software”, se refiere a los aspectos éticos y filosóficos de la libertad, el “open source” se refiere a los aspectos técnicos. **[19]**

“El titular de los derechos de autor (copyright) de un software bajo licencia copyleft puede realizar una versión modificada bajo su copyright original, y venderla bajo cualquier licencia que desee; además de distribuir la versión original como software libre.”

3.1.1. Definición de software libre

El software libre es un tipo particular de software, el cuál el grupo GNU lo de fine como:

``Software Libre" se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades que tienen los usuarios ante este software:

- ✓ *Libertad 0: La libertad de usar el programa, con cualquier propósito.*
- ✓ *Libertad 1: La libertad de estudiar cómo funciona el programa, y adaptarlo a las necesidades. El acceso al código fuente es una condición previa para esto.*
- ✓ *Libertad 2: La libertad de distribuir copias.*
- ✓ *Libertad 3: La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.* **[20]**

A continuación se mencionan otros conceptos de software libre:

Software libre: El software libre es aquél que se puede utilizar y distribuir libremente, mediante el pago de tasas gratuitas; cuyo código fuente es accesible, en caso contrario, está hecho con ciertas especificaciones públicas y su desarrollo esta construido por la comunidad de usuarios o cuenta con las aportaciones de los mismos.

Software libre: Software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. Este tipo de software por lo general se encuentra disponible en Internet ó a precios de la distribución, no necesariamente debe de ser de esta forma ya que puede ser vendido comercialmente, como Software libre. **[21]**

Software libre: Es aquel software cuyo secreto de fabricación (código fuente), es conocido y publicado libremente por el usuario. El autor del software confiere a cualquiera el derecho de usar su software; así como de modificarla y adaptarla a sus necesidades definidas. El derecho de libertad en ocasiones se da sin restricciones, pero otras veces con la condición de que las mejoras que tenga sean distribuidas con las mismas condiciones que tenia originalmente, para que siga disponible para todo el mundo (licencias tipo copyleft, como la GPL de la Free Software Foundation). **[22]**

Por otra parte, se puede mencionar que las personas u organizaciones tienen la libertad de usarlo en cualquier tipo de sistema computacional; así como para cualquier trabajo, sin tener alguna obligación de comunicarlo al desarrollador o alguna institución. **[20]**

A continuación se hace mención de las ventajas que tiene el software libre ante el software propietario:

Libertad para modificar el software: Según las necesidades de los usuarios u organizaciones, por otro lado el software propietario prohíbe las modificaciones del mismo.

Seguridad: Se da por la publicación del código fuente, permitiendo que el código pueda ser analizado por cualquier persona, para detectar fallos en el mismo. El software propietario no incluye el código fuente y no se puede lograr el análisis de este.

Confiabilidad: Se origina gracias a la disponibilidad del código fuente ante los usuarios, esto permite que el código se vaya mejorando. Sin en cambio el software propietario es desarrollado y probado por unas pocas personas que pueden pasar por alto errores, y cuando se encuentran estos tardan en solucionarse.

Portabilidad: Gracias a que el código esta disponible es más sencillo adaptar los programas, para el funcionamiento en varias arquitecturas y el software propietario solo se puede usar en arquitecturas para las que se diseñaron.

Precio: El código al no tener restricciones en la distribución origina que el costo sea barato y en ocasiones no tiene algún costo. Por otro lado en el software propietario se tienen que hacer pagos por licencias de cada copia de este.

En la actualidad existe software disponible con licencias de software libre. Por tal motivo los observadores mencionan que este es el movimiento del software libre. Ya que la gran mayoría de este tipo de software tiene:

- ✓ El kernel de los sistemas operativos GNU/Linux y BSD.
- ✓ Los copiladores GCC.
- ✓ El depurador GDB.
- ✓ Las bibliotecas de C.
- ✓ Los servidores Sendmail y el servidor web Apache.
- ✓ Sistemas de bases de datos relacionales (MySQL y PosgreSQL).
- ✓ Los lenguajes de programación (Perl, Python, Tcl y PHP).
- ✓ El sistema X Windows.
- ✓ Los entornos de escritorio (GNOME y KDE).
- ✓ La suite de ofimática OpenOffice.
- ✓ El navegador Mozilla.
- ✓ El servidor de ficheros Samba.
- ✓ El editor grafico GIMP. **[23]**

3.1.2. Licencias

Palabra referente a un contrato mediante el cuál el autor o autores de cierto software dan permisos restringidos de sus derechos de propiedad.

3.1.2.1. Licencias de software libre compatibles con GNU GPL

Licencia Pública General de GNU o GNU GPL: Licencia de software libre de tipo copyleft. Recomendada para la mayoría del software.

Licencia Pública General Reducida de GNU o GNU LGPL: Licencia de software libre, pero no tiene un copyleft fuerte, ya que permite que el software se enlace con módulos no libres. Recomendada para circunstancias especiales.

Licencia de Guile: Radica en la GNU GPL y una declaración que permite trabajar con software propietario. Esta no tiene un copyleft estricto.

Licencias simples, permisibles sin copyleft y compatibles con la GNU GPL:

- ✓ Licencia X11.
- ✓ Licencia Expat o MIT.
- ✓ Licencia de Copyright ML Estándar de New Jersey.
- ✓ Licencia General de Cryptix, similar a la licencia X11.
- ✓ Licencia BSD modificada.
- ✓ Licencia de ZLib
- ✓ La licencia de OpenLDAP, versión 2.7
- ✓ La Licencia Pública de Zope versión 2.0

Licencias de software libre compatibles con GNU GPL:

- ✓ El aviso y la licencia de software del W3C.
- ✓ La licencia de la base de datos de Berkeley.
- ✓ La Licencia de Python, versión 1.6a2 y anteriores.
- ✓ La Licencia de Python, versiones 2.0.1, 2.1.1 y posteriores.
- ✓ La Licencia Artística con aclaraciones.
- ✓ La Licencia Artística 2.0: Licencia que ya no esta en uso, considerada en Perl 6.
- ✓ La licencia eCos versión 2.0: Es GPL y permite unir software que no esta en GPL.
- ✓ La licencia del Forum Eiffel, versión 2: Las versiones anteriores de la licencia Eiffel no son compatibles con la GNU GPL.
- ✓ La licencia de `Vim`, versión 6.1 o posterior: Licencia no totalmente copyleft. **[24]**

3.1.2.2. Licencias de software libre incompatibles con la GNU GPL

Licencia Pública General de Affero: Licencia de tipo copyleft.

Licencia Pública Arphic: Licencia de tipo copyleft de uso para fuentes tipo gráficas.

Licencia BSD original: Licencia de tipo simple y permisible sin copyleft con el defecto de la cláusula publicitaria de BSD que ocasiona problemas prácticos.

Licencia Libre Académica (AFL), versión 1.1: Licencia sin copyleft. Incompatible por tener demandas de patentes.

Licencia de Apache, versión 1.0: Licencia simple y permisible sin copyleft, por tener problemas prácticos.

Licencia de OpenSSL: Licencia fusionada por la licencia de OpenSSL y la de SSLeay. Esta combinación da como resultado una licencia libre de tipo copyleft, pero el problema de incompatibilidad surge por la cláusula de publicidad de BSD.

Licencia de Software Abierto, versión 1.0.

Licencia Pública de Zope versión 1: Licencia simple y permisible sin copyleft con problemas prácticos.

Licencia de `xinetd`: Licencia de tipo copyleft, pero incompatible por poner restricciones en la distribución de las versiones modificadas.

Licencia de Python 1.6b1 y versiones posteriores hasta la 2.0, más la 2.1: Licencias incompatible por estar sujeta a leyes del estado de Virginia, en los EE.UU.

Licencia Pública Común versión 1.0.

Licencia de Código Fuente Abierto de Jabber, versión 1.0: Licencia que permite redistribuir el código con requisitos de las licencias Jabber y la GPL no pertenece a estas.

Licencia Pública Q (QPL), versión 1.0: Licencia sin copyleft, también posee inconvenientes prácticos y las fuentes modificadas se reparten en forma de parches.

Licencia FreeType: Su incompatibilidad de esta licencia se da por no tener copyleft y por razones técnicas.

Licencia de PHP, versión 3.0: Licencia sin copyleft.

Licencia Zend, versión 2.0: Licencia sin copyleft y con problemas prácticos.

Licencia de Plan 9 (junio de 2003).

Licencia de Fuente Pública de Apple (APSL), versión 2.

Licencia de Phorum, versión 2.0

Licencias que requieren de requisitos que no tiene la GPL:

- ✓ Licencia Pública de IBM, versión 1.0.
- ✓ Licencia de Apache, versión 2.0.
- ✓ Licencia de Apache, versión 1.1: Licencia permisible sin copyleft.
- ✓ Licencia Pública del Proyecto LaTeX (LPPL)

Licencias sin un copyleft fuente o con ciertas restricciones:

- ✓ Licencia Pública de Mozilla (MPL).
- ✓ Licencia de Código Fuente Abierto de Netizen (NOSL), versión 1.0.
- ✓ Licencia Pública de Interbase (IPL), versión 1.0.

- ✓ Licencia Pública de Sun.
- ✓ Licencia de Código Fuente Abierto de Nokia.
- ✓ Licencia Pública de Netscape (NPL)
- ✓ Licencia para el Código Fuente según los Estándares Industriales de Sun, versión 1.0. **[24]**

3.1.2.3. Licencias de documentación libre

Licencia de documentación libre de GNU: Licencia para documentos con copyleft y se pretende que se aplique a todos los manuales de GNU, también se puede aplicar en libros y diccionarios.

Licencia de documentación de FreeBSD: Licencia simple y permisible sin copyleft compatible con la GNU FDL.

Licencia de documentación común de Apple, versión 1.0: Licencia incompatible con la GNU FDL porque la sección (2c) dice: "no añada otros términos o condiciones a los de esta licencia", y la GNU FDL tiene términos adicionales no considerados en esta licencia.

Licencia de Publicación Abierta, versión 1.0: Licencia simple con copyleft que se puede usar como licencia de documentación libre, siempre y cuando el autor de los derechos no ejerza alguna opción de licencia. Si se solicita cualquiera de las opciones, la licencia deja de ser libre. Si es usada esta licencia sin alguna de las opciones para hacer un manual libre, alguien podrá cambiar los conceptos o palabras que estén en el manual, resultando el manual no libre. **[24]**

3.1.3. Historia de Debian

La organización Debian fue fundada en agosto de 1993 por *Ian Murdock*, en ese tiempo era estudiante de la Universidad de Purdue (Indiana, EEUU), cuando Ian Murdock funda Debian lo hace con respecto a las necesidades que tuvieran los usuarios del sistema conformando las características del software distribuido, evitando el beneficio comercial.

En enero de 1994 este proyecto contaba con grande grupo de personas interesadas, al mismo tiempo se publicaba un manifiesto fundacional con la declaración de intención (Debian Linux Manifestó). El manifiesto pronosticaba que la distribución adquiriría características del sistema GNU/Linux.

La idea de Debian de fundar un sistema operativo libre con plataforma Unix, que no necesita elementos que tengan propietario, tiene orígenes con el proyecto GNU de la Free Software Foundation (FSF). Esto por la adopción del concepto de “free software” (libertad de copia, modificación y redistribución). La Free Software Foundation dio apoyo financiero a Ian Murdock para la distribución de este sistema durante su consolidación (periodo de 1994 a 1995). Debian usa el kernel Linux, por

estas circunstancias los desarrolladores de Debian y la FSF trabajan para que en el futuro Debian pueda usar otros núcleos.

En 1996 Ian Murdock es sustituido del proyecto por Bruce Perens, quien en los dos años siguientes formo parte fundamental para establecer el Contrato Social de Debian, así mismo para que Debian tuviera influencia ante el público como sistema serio y fiable. También dirigió la iniciativa para fundar *Software in the Public Interest (SPI)*, organización fundada para dar validación a Debian, sin lucro y para dirigir las donaciones al proyecto. Otras de las circunstancias por las que fueron creadas las SPI, son para apoyar proyectos creadores y distribuidores de software y hardware libre.

Con el paso del tiempo Bruce Perens y Eric S. Raymond fundan la Open Source Initiative (OSI) basada en las Debian Guidelines para crear su concepto de open source y en 1999 Bruce Perens deja OSI para reclamar el nombre “free software”, para el movimiento de software libre.

Sucesores del proyecto Debian GNU/Linux (ver Tabla 3.1):

FECHA	NOMBRE
Agosto de 1993	Ian Murdock funda Debian.
En 1996	Bruce Pernees.
Enero 1998	Ian Jackson.
Enero de 1999 a marzo del 2001	Wichert Akkerman.
Abril del 2002	Hacker estadounidense Bdale Garbee.

Tabla 3.1: Sucesores del proyecto Debian GNU/Linux.

Hasta 1996 solo aparecen las versiones de Debian posteriores a la versión 1.0 ya que en este lapso de tiempo se ocupo para la consolidación del proyecto a nivel público, legal, organizacional y técnico, con el objetivo de tener un crecimiento en desarrolladores y numero de paquetes. Después de esta consolidación aparecen las versiones con los nombres de personajes de la película de Toy Store (ver Tabla 3.2), ideado por Bruce Perens ya que trabajaba en Pixar. **[25]**

De acuerdo a los analistas, Debian a acumulado una gran cantidad de usuarios y desarrolladores para continuar su desarrollo; es así que lo avalan diez distribuciones comerciales basadas en Debian. Sin embargo ya que Debian esta formado por usuarios, estos son los que deben de resolver sus problemas y estrategias, por esta razón nadie puede garantizar nada en Debian ni en el software libre.

Conforme se desarrolla Debian se va mejorando el proceso de instalación. Sin la intención de simplificar este proceso al usuario, no tomando todas las decisiones en su lugar, que algunas distribuciones llegan a impedir la instalación de hardware que no detecte. Debian en la actualidad no detecta todo el hardware, manteniendo un interfaz de instalación, con menús en modo texto que realizan preguntas, esto en ocasiones confunde al usuario, permitiendo una instalación incorrecta de los

componentes, resultando que la instalación tenga fama de ser compleja, por lo general esta fama es provocada por usuarios que intentan hacer una instalación apresurada. **[26]**

ESTADO	FECHA	VERSIÓN	NOMBRE	NOTA	DESARROLLO	PKGS	ARQUITECTURAS
?	08-12/1993	0.1-0.9	?	[1]	1	?	i386
?	01/1994	0.91	?	[2]	~12	?	i386
?	03/1995	0.93R5	?	[3]	~30	?	i386
?	11/1995	0.93R6	?	[4]	60	250	i386
Estable	06/1996	1.1	Buzz	[5]	?	474	i386
Estable	12/1996	1.2	Rex		120	848	i386
Estable	07/1997	1.3	Bo	[6]	200	974	i386
Estable	07/1998	2.0	Hamm	[7]	400	1500+	i386/m68k
Estable	03/1999	2.1	Slink	[8]	~400	2250	i386/m68k/ alpha/sparc
Estable	08/2000	2.2	Potato	[9]	450	3900+	i386/m68k/ alpha/sparc/ ppc/arm
Estable	?	3.0	Woody	[10]	~500		i386/m68k/ alpha/sparc/ ppc/arm
Inestable	?	?	Sid	[11]	~500	6433	i386/m68k/ alpha/sparc/ ppc/arm/ mips/ia64/ s390/sh/hppa/ hurd

Tabla 3.2: Versiones de Debian GNU/Linux.

En conclusión, Debian no solo es una gran distribución GNU/Linux estable, también tiene gran preferencia por los usuarios. El proyecto Debian no vende CD`s con su software y hardware, por que este puede ser distribuido por cualquier persona que lo tenga. **[27]**

3.1.4. Debian

Debian es una organización compuesta de voluntarios, para el desarrollo de software libre, con los ideales de la Free Software Foundation. En 1993 Ian Murdock rehizo una invitación a los desarrolladores de software a contribuir para hacer una distribución completa usando el núcleo Linux influenciado por la filosofía de la GNU (ver Figura 3.2). Los desarrolladores de Debian se han implicado en muchas labores como administración del servidor Web y FTP, diseño gráfico, análisis legal de licencias de software, escribir documentación y mantener paquetes de software.

Figura 3.2: Logotipo de la distribución Debian. **[28]**

El proyecto Debian ha publicado documentos que contienen sus valores, sirviendo como guía de lo que significa ser Desarrollador Debian: **[28]**

Contrato Social de Debian: Es la aceptación de responsabilidad de Debian con la comunidad del Software Libre. Cualquier que acepte el Contrato Social puede convertirse en un desarrollador. Cualquier desarrollador puede anexar nuevo software en Debian, siempre y cuando éste cumpla los principios de software libre y estándares de calidad. **[29]**

El documento Directrices de Software Libre de Debian (DFSG): Informe claro y conciso de los criterios de Debian sobre el software libre. La DFSG es de gran influencia en el movimiento del Software Libre, y proporciona las bases de la Definición de Open Source. **[30]**

La Debian Policy: Explica los estándares de calidad del Proyecto Debian. **[31]**

Proyectos de la comunidad Linux, en los cuales están involucrados los desarrolladores de Debian:

El Linux Standard Base (LSB): Proyecto que intenta estandarizar el sistema básico de GNU/Linux, facilitando a desarrolladores desconocidos de software y hardware, desarrollar con facilidad programas y controladores de dispositivos para Linux y no para la distribución de GNU/Linux. **[32]**

El Filesystem Hierarchy Standard (FHS): Esfuerzo que quiere estandarizar la distribución del sistema de ficheros de Linux. El FHS permitiendo a desarrolladores de software a unir esfuerzos para diseñar programas, sin importarles cómo se instalen en las diferentes distribuciones de GNU/Linux. **[33]**

Debian Jr: Proyecto interno que asegura que Debian tiene algo que ofrecer a usuarios jóvenes. **[34]**

3.1.5. Linux

En la década de los 90´s el estudiante Linus Torvals de 23 años, de la Universidad de Helsinki en Finlandia, inicio a crear como pasatiempo el proyecto basado en Minix de Andrew Tenembaum, en una computadora con un procesador Intel 80386, un sistema operativo tipo UNIX, que diera mayores ventajas que Minix. Así Linus comenzó a desarrollar su proyecto basado en lenguaje ensamblador, después de esto añade código hecho en el lenguaje C, acelerando el desarrollo de lo que seria Linux, de esta forma Linus tomo en serio el desarrollo de un Minix mejorado. La primer versión 0.01 fue creada sin tener controladores para disquete, por tal razón ni siquiera la dio a conocer, mas adelante le anexo un sistema de archivos y controladores para disquete con errores pero funcional, siendo a si en octubre de 1991 anuncia la primer versión oficial de Linux 0.02, capaz de ejecutar el SHELL bash y el compilador gcc de GNU. **[35]**

Cuando Linus tuvo una versión estable comenzó a distribuirla bajo la licencia GPL y solicito ayuda para probarlo y mejorarlo con el mensaje. El mensaje apareció en uno de los foros de discusión de *usenet comp.os.minix*, con el cuál Linus Torvalds animaba a los desarrolladores a participar en el proyecto Linux

Do you pine for the nice days of Minix 1.1, when men were men and wrote their own device drivers? Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on your OS? No more all-nighters to get a nifty program working? Then this post might be just for you.

¿Echa de menos aquellos maravillosos días de Minix 1.1, cuando los hombres eran hombres y escribían sus propios controladores de dispositivos? ¿No tiene ningún proyecto interesante y se muere de ganas por hincarle el diente a un sistema operativo que pueda modificar para ajústalos a sus propias necesidades? ¿Encuentra frustrante que todo funciona en su sistema operativo? ¿Ya no para noche en vela intentando configurar un ridículo programa? Entonces, este mensaje puede ser justamente para usted.

A partir de ahí, el sistema de Linus empezó a crecer. **[I]**

Linux tiene como logotipo o mascota oficial un pequeño pingüino sonriendo llamado "Tux" (ver Figura 3.3), derivado de **T**orvalds **U**ni**x**, creado en 1996 por Larry Ewing. Este logotipo se puede modificar como se requiera con las características que Larry Ewing define **[36].**

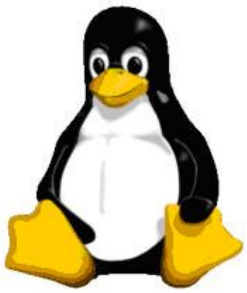


Figura 3.3: Logotipo de Linux "Tux". **[36]**

Como resultado de esto, surge lo que es la distribución Debian GNU/Linux:

Linux →Núcleo del sistema operativo.
GNU → Proyecto orientado a crear y desarrollar software libre.
Debian → Empresa dedicada a desarrollo de software libre como Debian GNU/Linux.

3.1.6. GNU/Linux

En 1991 el estudiante de ciencias de la computación Linus Torvalds da a conocer la primer versión del núcleo Linux el cuál sustituiría a Minix, en el grupo noticiero

Usenet comp.os.minix. Cuatro años después, el programador Richard M. Stallman se propone crear un sistema operativo funcional totalmente libre usando el núcleo Linux. De esta manera surgía el proyecto GNU, "GNU is not Unix" haciendo referencia al sistema operativo al que pretendía emular.

El proyecto GNU esta orientado a crear herramientas para el Software libre de tipo Unix como Linux. Estas herramientas ayudan a los usuarios a realizar las operaciones como: copiar, eliminar ficheros y escribir, compilar programas en diferentes formatos de documentos. Tomando en cuenta que el componente mas importante de un sistema operativo es el núcleo.

El sistema operativo GNU/Linux esta formado por el núcleo Linux y los demás componentes o herramientas son GNU, por esta razón ya que el núcleo Linux no es un sistema funcional, se optó por usar el término "GNU/Linux" (ver Figura 3.4) aunque la gran mayoría de las personas llaman de manera informal a este sistema operativo "Linux". **[37]**

Las características principales de un sistema GNU/Linux son:

- ✓ *Multiplataforma:* Funcionalidad en PC, Mac y en otras máquinas que no son PC.
- ✓ *Multitarea:* Capacidad de realizar varios procesos simultáneamente.
- ✓ *Multiusuario:* Barios usuarios pueden acceder simultáneamente a la misma máquina. **[22]**

Figura 3.4: Logotipo GNU/Linux. **[37]**

3.1.7. Debian GNU/Linux

Debian GNU/Linux es una distribución formada por la unión de la filosofía y metodología de Debian; así como las herramientas GNU, el núcleo Linux, y otro software libre. Debian GNU/Linux esta compuesto por paquetes, los cuales contienen ejecutables, scripts, documentación, información de configuración y tiene un administrador que tiene como función principal tener el paquete actualizado, así como seguir informes de errores y comunicarlos con los autores principales del software.

El cuidado de Debian a detalle da originen a una distribución de alta calidad, estable y escalable. Dando origen a una instalación fácil de configurar para servir a varios perfiles, desde cortafuegos reducidos a lo mínimo, estaciones de trabajo científicas, asta servidores de red de alta escala.

El factor que diferencia a la distribución Debian GNU/Linux de otras distribuciones GNU/Linux, es su sistema de gestión de paquetes. Ya que sus herramientas dan control al administrador sobre los paquetes instalados en su sistema como: capacidad de instalar un paquete o actualizar todo el sistema operativo, otra de las opciones sobre los paquetes es que se pueden proteger para no actualizarlos. También el sistema de gestión de paquetes avisa qué software ha compilado y que dependencias cumple.

En cuanto a protección del sistema contra virus Debian GNU/Linux comprueba que los paquetes que se instalen sean de orígenes de sus desarrolladores. Los empaquetadores de Debian GNU/Linux, ponen mucho cuidado en configurarlos de forma segura. En caso de detectar un problema de seguridad con los paquetes entregados, los parches se encuentran disponibles para resolver el problema. Con el sistema de actualización se pueden descargar modificaciones de seguridad automáticamente por Internet.

Uno de los métodos más recomendables para obtener soporte sobre el sistema Debian GNU/Linux, es comunicarse con los desarrolladores de Debian por medio de las distribuciones mantenidas por el proyecto Debian. **[38]**

3.1.8. Como obtener Debian GNU/Linux

Debian GNU/Linux se puede obtener de la mejor manera o como se le haga mas fácil al usuario interesado en tenerlo en su computadora, ya que en la actualidad existen cuatro diferentes formas de cómo se puede adquirir, las cuales se muestran a continuación:

Compra de CD´s a alguno de los vendedores que ofrecen Debian GNU/Linux: Cuando los usuarios son nuevos en instalar este sistema se recomienda que se adquieran los CD`s. La gran mayoría de los vendedores ofrecen Debian GNU/Linux a bajo costo más gastos de envió. Entre las ventajas de esta opción para obtener el sistema, son las siguientes:

- ✓ La instalación es más directa.
- ✓ Se logra instalar Debian GNU/Linux en la computadora, sin tener que estar conectado a Internet.
- ✓ Se puede instalar Debian GNU/Linux en diferentes computadoras.
- ✓ Se pueden usar los CD`s para recuperación del sistema.

Descargar archivos necesarios desde Internet para iniciar la instalación "creación de CD`s": Si el usuario elije esta opción solo tendrá las siguientes ventajas:

- ✓ Puede crear sus propios CD`s.
- ✓ Aprovecha su conexión a Internet.
- ✓ No tiene que esperar, ni pagar por gastos por el envió.

Descargar el sistema de instalación por Internet y adquirir el resto en el transcurso de la instalación: Cuando se instala Debian GNU/Linux por esta opción, se instalaran solo archivos de tipo *.deb* que el usuario quiera, ya que los archivos por lo general están actualizados.

Comprar una computadora con Debian GNU/Linux preinstalado: Al adquirir Debian GNU/Linux en una computadora, no se tiene que instalar ya que la instalación viene configurada con el hardware y en ocasiones el fabricante proporciona soporte técnico. **[39]**

3.1.9. Sobre Copyrights

En muchas ocasiones las personas dicen que el software de código abierto no tiene copyrights ya que en el software comercial o propietario el copyrights dice que solo se puede instalar una copia en una computadora.

En el software libre también existe el copyrights, este es empleado para garantizar su libertad, por esta razón se cambia el nombre de "copyrights" a "copyleft" teniendo el derecho de copia. En otras palabras se dice que el copyrights del software libre es la libertad de copia.

El significado de copyleft, expresa que cualquiera que distribuya el software con o sin modificaciones, no podrá quitarle la libertad de ser copiado, redistribuido o cambiado. Copyleft garantiza que el usuario mantenga su libertad de hacer lo que quiera con el software libre. **[40]**

3.1.10. Las distribuciones GNU/Linux

Cuando se habla de distribuciones Linux es lógico saber que se habla de software libre, debido a que la palabra "gratis" aparece frecuentemente, esto no quiere decir que el lector no debe de dejar de tener en cuenta que no haya implicaciones comerciales o monetarias el mundo de Linux. Si existen, tienen una filosofía diferente, aunque buscan el beneficio propio. Aquí se debe de tener en cuenta que las empresazas de este sector cobran cantidades monetarias pequeñas por ser propietaria del sistema operativo, por empaquetarlo, por facilitar la instalación y añadir aplicaciones que pueden ser útiles, como:

- ✓ Copiladores C/C++.
- ✓ Sutes ofimáticas.
- ✓ Juegos.
- ✓ Acceso a Internet.
- ✓ Entornos gráficos "KDE y Gnome"
- ✓ Herramientas para administrar redes.
- ✓ Servidor Web.
- ✓ Servidor de correos.
- ✓ Servidor FTP.
- ✓ Etc……. **[J]**

También es importante mencionar que existen muchas distribuciones que vale la pena mencionar, entre las cueles se encuentran las siguientes:

- ✓ MCC Interim Linux.
- ✓ Tamu Linux
- ✓ Debian GNU/Linux,
- ✓ El CD-ROM Ydddrasil Plug-And-Play y Linux bible.
- ✓ El CD-ROM Trans-Ameritech Linux Plus BSD.
- ✓ El CD-ROM The Linux Quarterly.
- ✓ Caldera (El fabricante utiliza la distribución Red Hat).
- ✓ Red Hat.
- ✓ Gentoo.
- ✓ Knoppix
- ✓ GNU/LinEx.
- ✓ Mandrake Linux.
- ✓ MkLinux.
- ✓ Slackware.
- ✓ SuSE.
- ✓ Yellow Dog Linux.**[K]**

3.2. Hardware soportado

Actualmente Debian GNU/Linux no necesita software adicional, solo el hardware que requiere el núcleo Linux y las herramientas GNU. En otras palabras se dice que en cualquier arquitectura y plataforma que funcione el kernel Linux, libc, gcc, etc. y adaptaciones que existan para Debian GNU/Linux, este podrá ser instalado sin ningún problema de Hardware.

De todas formas existen limitaciones respecto al hardware soportado por los disquetes o CD`s de instalación. Algunas plataformas soportadas por el núcleo Linux, en ocasiones no son directamente soportadas por los disquetes o CD`s de instalación. En caso de que ocurriera esto puede crear un disco de rescate personalizado u obtener información sobre instalaciones de red.

3.2.1. Múltiples procesadores

Debian GNU/Linux es soportado por el multiprocesamiento simétrico (SMP), el cual esta incluido en una imagen precompilada de su núcleo. Dependiendo de su medio de instalación, este núcleo con soporte de SMP podrá ser instalado o no de la misma forma que el núcleo. Esto no debería obstaculizar la instalación, ya que el núcleo estándar (sin SMP), debería arrancar en sistemas SMP. En estas circunstancias, el núcleo usa el primer procesador.

Al usar múltiples procesadores el paquete del núcleo que soporta SMP debe de estar instalado, en caso de que no lo este, se debe de elegir un núcleo correcto. Además se puede construir núcleos personalizados con soporte SMP. En la versión del

núcleo 2.6.11, la forma de habilitar SMP es elegir *symmetric multi-processing* en la sección *General* de la configuración del núcleo. **[41]**

3.2.2. Arquitecturas soportadas

Actualmente se puede instalar Debian GNU/Linux en once tipos diferentes de arquitecturas, así como en las variaciones que existen entre ellas (ver Tablas 3.3).

Arquitectura	**Designación de Debian**	**Subarquitectura**	**Variación de arquitectura**
Basada en Intel x86	I386		vanilla idepci compact bf2.4 (experimental)
Motorola 680x0	m68k	Atari Amiga 68k Macintosh	Atari amiga mac
		VME	bvme6000 mvme147 mvme16x
DEC Alpha	alpha		generic jensen nautilus
Sun SPARC	sparc		sparc32 sparc64
ARM y StrongARM	arm		riscpc netwinder shark lart
IBM/Motorola PowerPC	powerpc	PReP PowerMac CHRP APUS	chrp powermac, new-powermac prep apus
HP PA-RISC	hppa	PA-RISC 1.1 PA-RISC 2.0	32 64
Basado en Intel ia64	ia64		
MIPS (big endian)	mips	SGI Indy/I2	r4k-ip22
		Broadcom BCM91250A (SWARM)	sb1-swarm-bn
MIPS (little endian)	mipsel	DEC Decstation	r4k-kn04 r3k-kn02
		Broadcom BCM91250A (SWARM)	sb1-swarm-bn
IBM S/390	s390		vmrdr tape

Tabla 3.3: Arquitecturas soportadas por Debian GNU/Linux.

3.2.3. CPU, motherboards (bus de E/S) y soporte de video

CPU: Debian GNU/Linux soporta los procesadores que estén basados en x86, incluyendo los procesadores AMD y VIA (antes Cyrix). De acuerdo a los

procesadores que van surgiendo con el paso del tiempo también son soportados, como Athlon XP e Intel Pentium 4 Xeo. Debian GNU/Linux no puede ser instalado en procesadores 286 y anteriores.

Motherboards (bus de E/S): En Motherboards se aborda acerca de los bus del sistema ya que estos son la parte importante que comunica la CPU con los demás periféricos. Los bus que tienen la compatibilidad con Debian GNU/Linux, son los siguientes:

- ✓ ISA.
- ✓ EISA.
- ✓ PCI.
- ✓ Arquitectura Microchannel (MCA, usado en IBM's PS/2).
- ✓ VESA Local Bus (VLB, también nombrado como bus VL)

Tarjetas de video: Debia GNU/Linux es capaz de trabajar con casi todo tipo de tarjeta de video, ya que usa una interfaz compatible con VGA para la terminal de consola. Por esta razón las tarjetas modernas que sean compatibles con VGA podrán soportar este sistema. Pero existen algunos modelos de tarjetas viejas que tienen compatibilidad con VGA, sin embargo no funcionan.

La ayuda para interfaces graficas de Debian están establecidos por el soporte encontrado en el sistema X11 de XFree86. Los puertos de video AGP actuales son modificaciones de PCI y las tarjetas de video AGP funcionan bajo XFree86. **[41]**

3.3. Requerimientos de sistema

En esta sección se especifica la memoria RAM, espacio en disco duro necesario para tener una instalación satisfactoria y un excelente funcionamiento del sistema operativo Debian GNU/Linux.

SISTEMA	ESPACIO EN DISCO	MEMORIA RAM
Sistema basado en consola.	250MB	32MB
Sistema con ventanas, algunos programas y bibliotecas de desarrollo.	400MB	32MB
Sistema más o menos completo.	800MB	32MB
Sistema Debian GNU/Linux completo.	2GB	32MB

Tabla 3.4: Requisitos para instalar Debian GNU/Linux.

Cuando se instala Debian GNU/Linux, se debe tener en cuenta que se puede instalar de distintas maneras, esto con la finalidad de que se use de la mejor manera, de acuerdo a las características que tenga el tipo de instalación (ver tabla 3.4). Sin dejar de mencionar que todos los discos duros son compatibles con Debian GNU/Linux. **[42]**

3.4. Recomendaciones antes de la instalación

Antes de instalar Debian GNU/Linux, se hacen las siguientes recomendaciones, para que no surjan problemas o en caso de que los haya, puedan resolverse de manera correcta y adecuada.

3.4.1. Respaldo de archivos

En caso de que sea la primera vez que se instala el sistema operativo Debian GNU/Linux, será necesario realizar particiones en disco duro dejando espacio para hacer dicha instalación.

En ocasiones se elijen programas para hacer espacio en disco duro, pero se tiene que tener en cuesta que una elección por parate del usuario puede ocasionar un error y borrar la información contenida en dicho disco. Por eso se recomienda que antes de hacer una instalación de cualquier sistema operativo, se realice una copia de la información que se tenga guardada en el disco duro, donde se realizara la instalación de dicho sistema, en este caso se instalara Debian GNU/Linux. **[43]**

3.4.2. Leer la documentación de instalación

Una de las recomendaciones mas importantes antes de iniciar la instalación de Debian GNU/Linux, es la lectura de la documentación de dicho sistema operativo para no llegar a tener dificultades o errores en el proceso de instalación, configuración y administración del sistema, otra de las recomendaciones que se hacen, es la investigación de información adicional de los componentes o requisitos necesarios para tener éxito con este proceso, uno de los documentos que mas se recomienda, es la lectura del manual o guía de instalación de Debian GNU/Linux, que se puede localizar en la dirección de Internet o página principal de Debian. **[43]**

3.5. Métodos de instalación

Al elegir el método de instalación, se puede observar que se tienen cuatro diferentes opciones para poder tener una instalación de Debian GNU/Linux en la computadora que se requiera, como se exponen a continuación:

CD-ROM/DVD ROM: Por medio de este método se puede realizar una instalación sin tener que recurrir a un disquete de arranque ya que se reconocen las unidades de CD-ROM´s IDE/ATAPI y SCSI. Los CD y DVD se pueden conseguir con cualquier distribuidor de Debian GNU/Linux, ofreciendo soporte para ciertas arquitecturas.
Disco duro: Método de instalación que se puede aplicar a diferentes arquitecturas, siempre y cuando exista oto sistema operativo instalado en el disco duro donde se desea instalar Debian GNU/Linux.

Red (Imagen NFS, FTP, HTTP): Opción que ofrece la instalación de Debian GNU/Linux desde una red de área local, montando imágenes NFS de los ficheros locales, para instalar el núcleo del sistema, posteriormente se podrán complementar

la instalación por medio de la configuración de una conexión de red mediante FTP o HTTP.

Sistema Unix o GNU: Este método puede ser útil para usuarios que desean instalar Debian GNU/Linux y que tengan hardware no soportado o computadoras que no dejen de dar servicios. Ofrece instalar Debian GNU/Linux desde un sistema de tipo Unix (Red Hat, Mandrake y SuSE), sin tener que usar el debian-installer.

En caso que se desee instalar Debian GNU/Linux, con otro sistema operativo en la misma computadora será necesario un sistema de arranque dual (partición de disco duro y configuración del gestor de arranque), solo así se podrán instalar ambos sistemas operativos, iniciando desde sus particiones correspondientes, pero no a la misma vez. **[43]**

3.6. Configuración de BIOS

Para iniciar con la instalación primero se tiene que contar con el juego de CD-ROM´s, después de haberlos conseguido se iniciara con la configuración del BIOS de la computadora donde se instalara Debian GNU/Linux, seleccionando la unidad de CD-ROM, para que pueda iniciar con la lectura de dichos CD-ROM´s como se menciona a continuación:

- ✓ AMI BIOS: Tecla **Supr** durante el POST (auto-verificación al encendido).
- ✓ Award BIOS: Teclas **Ctrl**-**Alt**-**Esc** o **Supr** durante el POST.
- ✓ DTK BIOS: Tecla **Esc** durante el POST.
- ✓ IBM PS/2 BIOS: **Ctrl**-**Alt**-**Insert** después **Ctrl**-**Alt**-**Supr.**
- ✓ Phoenix BIOS: **Ctrl**-**Alt**-**Esc** o **Ctrl**-**Alt**-**S** o **F1.**

Debido a la variedad de distintos BIOS es imposible dar instrucciones precisas acerca de como modificar las opciones necesarias, pero en la mayoría de las ocasiones se tiene que realizar lo siguiente:

- ✓ Desactivar la protección antivirus del BIOS.
- ✓ Desactivar sistema operativo Plug and Play.
- ✓ Seleccionar el CD-ROM (generalmente unidad D: ó E:) como primer dispositivo de arranque. **[44]**

3.7 Pasos para instalar Debian GNU/Linux desde CD-ROM

Como ya se mencionaron las características o requisitos principales para tener una instalación satisfactoria de Debian GNU/Linux, a continuación se enumeran los pasos necesarios para realizar dicha instalación del sistema operativo, cabe mencionar que durante el proceso de instalación solo se hará uso del teclado, ya que no se puede usar el Mouse para elegir las diferentes opciones.

Paso 1: Se inserta el CD-ROM número uno de Debian GNU/Linux, en la unidad lectora de CD-ROM, para que se ejecute la instalación, donde aparecerá una pantalla de bienvenida (ver Figura 3.5).

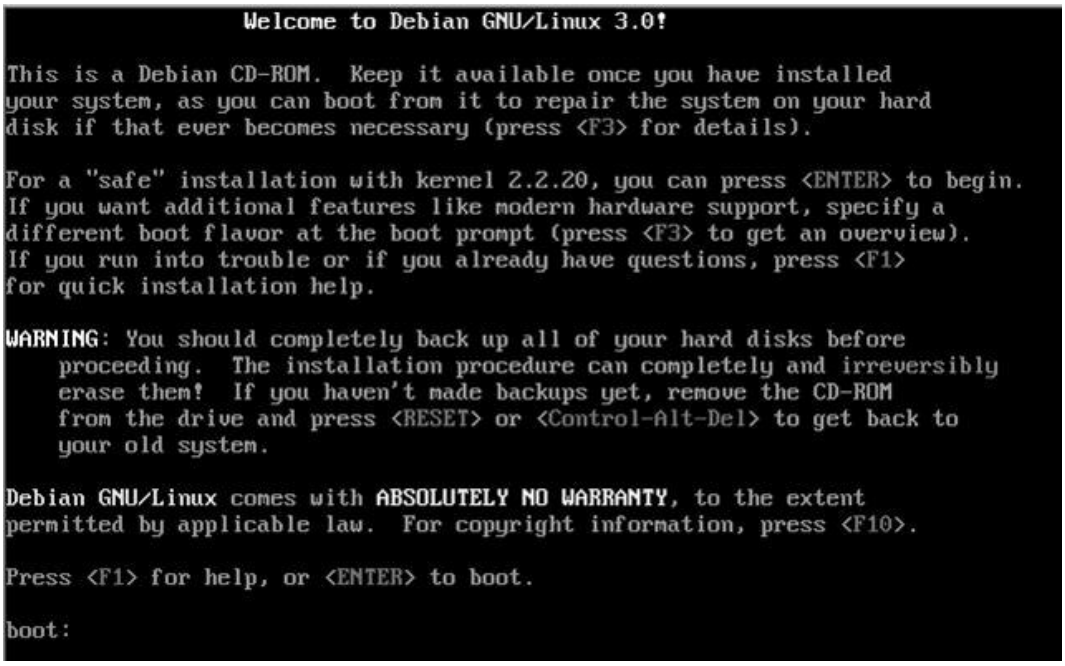


Figura 3.5: Pantalla de bienvenida a Debian GNU/Linux. **[45]**

Paso 2: Ventana de Selección de lenguaje:

es-Elija esta opción y pulse enter para continuar en español.

Paso 3: Ventana que muestra las notas de la versión (ver Figura 3.6), seleccionar *<continuar>*.

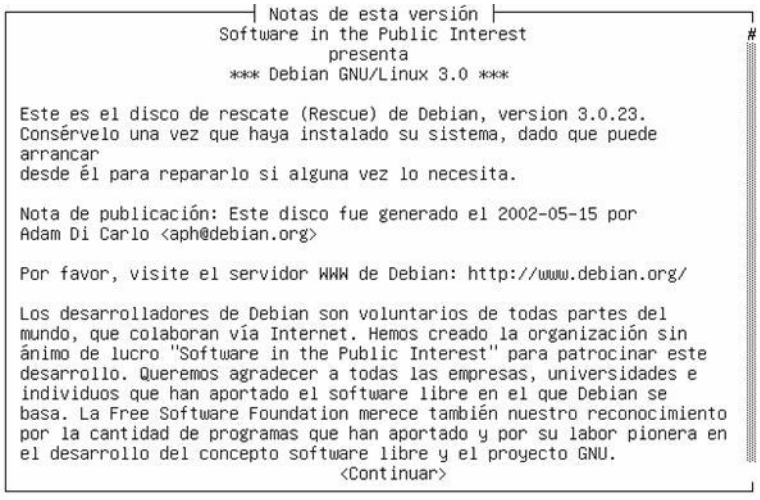


Figura 3.6: Ventana que muestra la nota de la versión de Debian GNU/Linux. **[45]**

Paso 4: Ventana principal del menú de instalación de Debian GNU/Linux (ver Figura 3.7), se teclea enter para continuar con la configuración del teclado.

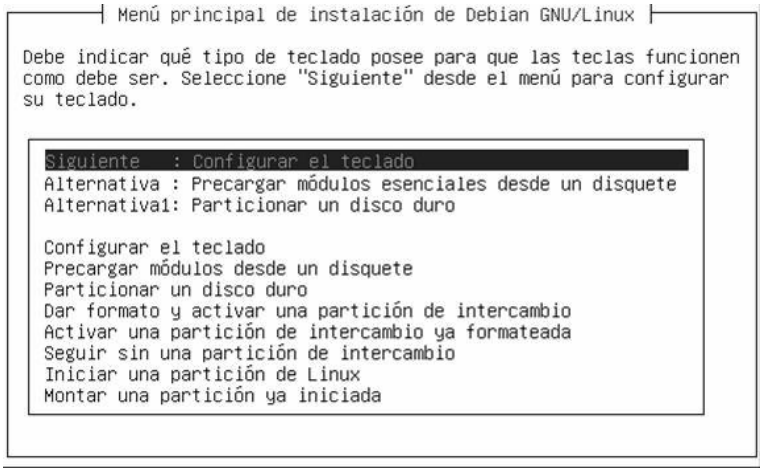


Figura 3.7: Menú principal de instalación de Debian GNU/Linux. **[45]**

Opciones del menú de instalación de Debian GNU/Linux:

- ✓ Configurar el teclado.
- ✓ Precargar módulos desde un disquete.
- ✓ Particionar un disco duro.
- ✓ Dar formato y activar una partición de intercambio.
- ✓ Activar una partición de intercambio ya formateada.
- ✓ Seguir sin una partición de intercambio.
- ✓ Iniciar una partición de Linux.
- ✓ Montar una partición ya iniciada.
- ✓ Desmontar una partición.
- ✓ Instalar el núcleo y los módulos.
- ✓ Configurar los controladores de dispositivos.
- ✓ Configurar el soporte para PCMCIA.
- ✓ Instalar módulos externos.
- ✓ Configurar el nombre del sistema.
- ✓ Configurar la red.
- ✓ Instalar el sistema de base.
- ✓ Editar parámetros de arranque del núcleo.
- ✓ Hacer el sistema arrancable.
- ✓ Crear un disquete de arranque.
- ✓ Reiniciar el sistema.
- ✓ Ver la tabla de particiones.
- ✓ Ejecutar un intérprete de órdenes.

Paso 5: Ventana de selección de teclado. Se selecciona la siguiente opción (enter para continuar):

qwerty/es : Español

Paso 6: En este paso se mostrara nuevamente el menú principal de Debian GNU/Linux, con la opción *Particionar un disco duro* (enter para continuar).

Paso 7: Ventana para seleccionar la unidad de disco duro (ver Figura 3.8), en esta elección se usara la utilidad *cfdisk* (solo permite crear y borrar particiones), opción donde se debe tener mucho cuidado para no llegar a dañar o borrar la información contenida en el disco duro que se va a particionar (enter para continuar).

Figura 3.8: Selección del disco duro a particionar. **[45]**

Paso 8: Después de haber seleccionado el disco duro, se mostrara una ventana con las *Limitaciones de LILO*. Solo bastara con seleccionar la opción *<continuar>* (enter), para ir al siguiente paso.

Paso 9: Ventana con las notas sobre el espacio adicional para la bitácora de ReiserFS, se opta por la opción *<continuar>*.

Paso 10: Pantalla principal de cfdisk (ver Figura3.9). Se procede a seleccionar (con las flechas arriba, abajo, izquierda y derecha) el espacio disponible en el disco duro y se elije la opción *New,* para crear una partición nueva (enter).

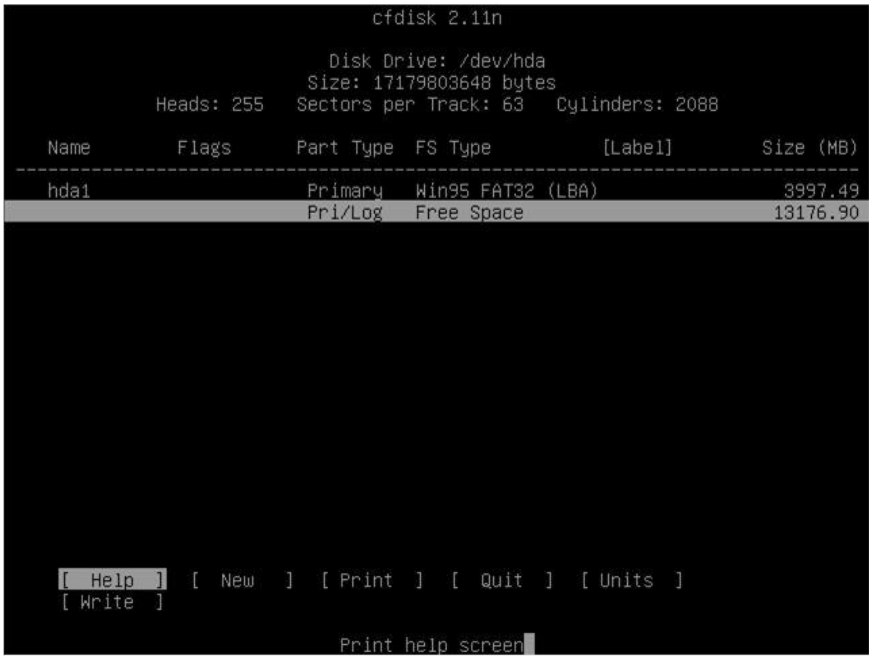


Figura 3.9: Pantalla principal de cfdisk. **[45]**

Paso 11: Se selecciona como se quiere especificar la partición (primary o logical) y el tamaño con la que se desea crear (ver Figura 3.10).

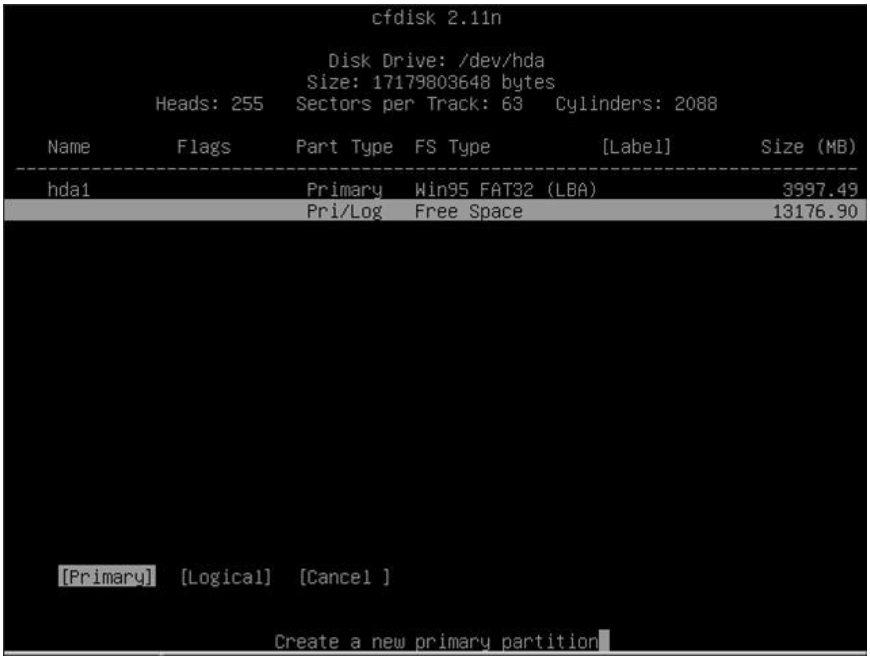


Figura 3.10: Creación de una partición. **[45]**

Paso 12: Para finalizar con este proceso se selecciona la opción *Type* (ver Figura 3.11), para mostrar los diferentes tipos de formatos que se le pueden a signar a las particiones creadas.

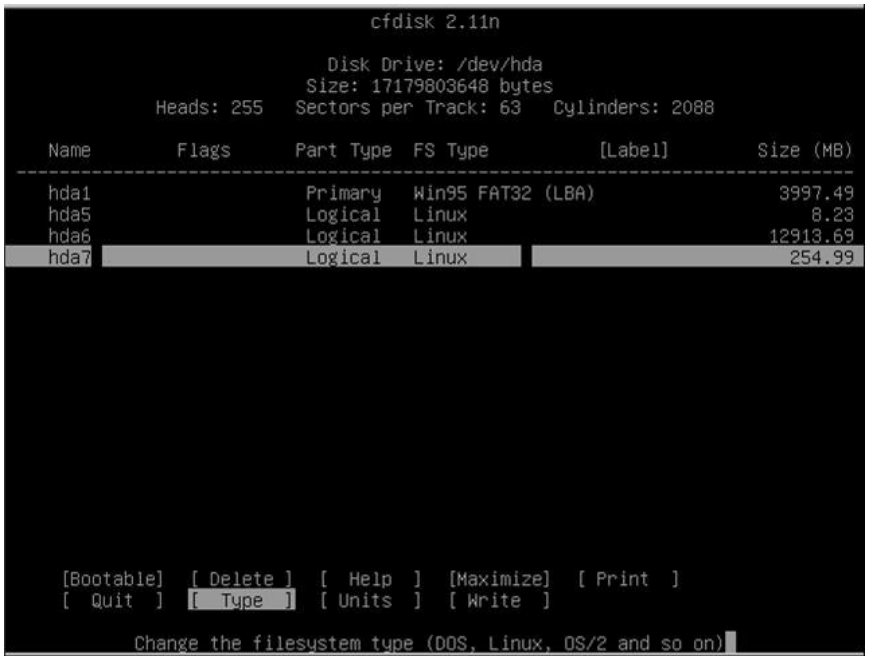


Figura 3.11: Selección de la opción Type. **[45]**

Paso 13: Después de seleccionar *Type* se mostrará una nueva pantalla (ver Figura 3.12), donde se escogerá la opción *82 Linux Swap,* para formatear la partición creada.

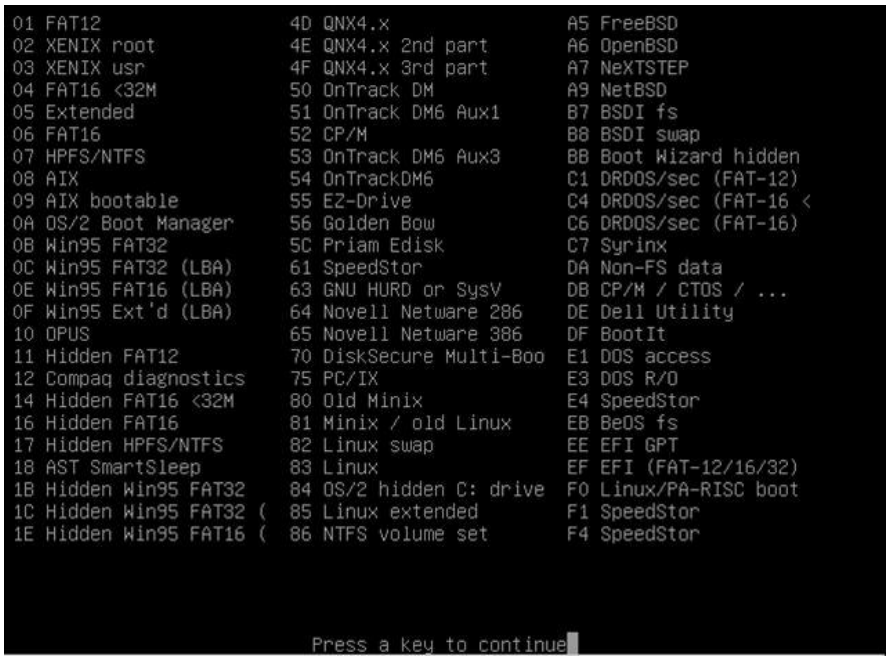


Figura 3.12: Tipos de particiones. **[45]**

Una vez seleccionada la opción *Swap*, se selecciona *Write* para escribir los cambios en la partición creada (se mostrará un mensaje, solo se teclea *yes* y enter para continuar). Cuando se terminan de crear las particiones para Debian GNU/Linux, se seleccionara la opción *Quit*, para salir de la pantalla *cfdisk.*

Paso 14: De nuevo se mostrará el menú principal de Debian GNU/Linux, con la opción *Dar formato y activar una partición de intercambio* (enter para continuar).

Paso 15: En este paso se mostrará una ventana para seleccionar una partición de intercambio (se selecciona la siguiente opción), donde # es el numero de partición:

/dev/hda#: Linux swap

o

/dev/hda#: Linux native

Paso 16: Ventana para buscar bloques defectuosos, aquí se opta por la opción *<No>* (ver Figura 3.13).

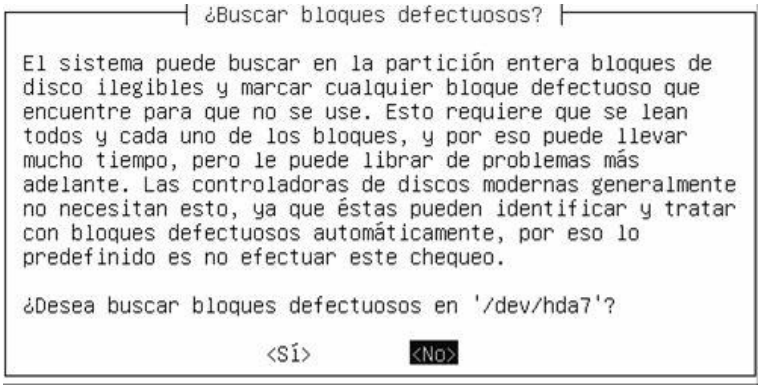


Figura 3.13: Ventana para buscar bloques defectuosos. **[45]**

Paso 17: Después del paso anterior, se mostrará la ventana que preguntara si se *¿Esta seguro?* de no buscar bloques defectuosos, escoger las opción *<Si>* (ver Figura 3.14).

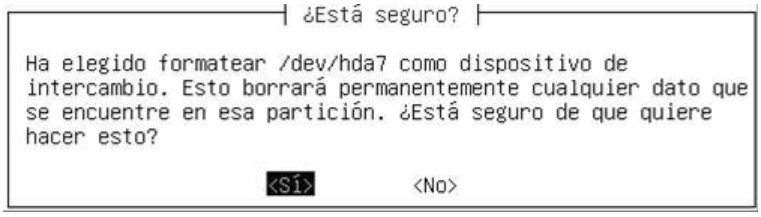


Figura 3.14: Confirmar o no la búsqueda de bloques defectuosos. **[45]**

Paso 18: A continuación se mostrara el menú principal de Debian GNU/Linux, con la opción *Iniciar una partición de Linux* (enter para continuar).

Paso 19: Ventana para *Seleccionar el tipo de sistema de ficheros,* elegir la opción siguiente:

Ext2: Sistema de ficheros tradicional de GNU/Linux

Se teclea enter para que se muestre los pasos 16 y 17.

Paso 20: Ventana, la cual preguntara si se quiere *¿Mostrar como sistema de fichero raíz?*, seleccionar la opción *<Si>* (ver Figura 3.15).

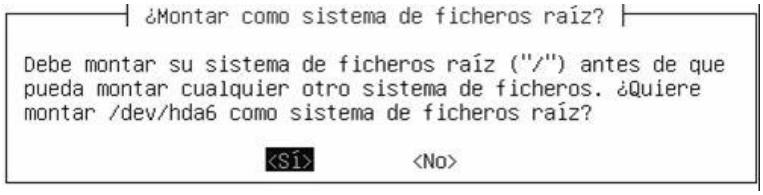


Figura 3.15: Mostrar el sistema como fichero raíz. **[45]**

Paso 21: Menú principal de Debian GNU/Linux, con la opción *Instalar el núcleo y los módulos* (enter para continuar).

Paso 22: Ventana con el mensaje de que, *Se encontró un CD-ROM de Debian* (ver Figura 3.16), se elige la opción *<Si>*.

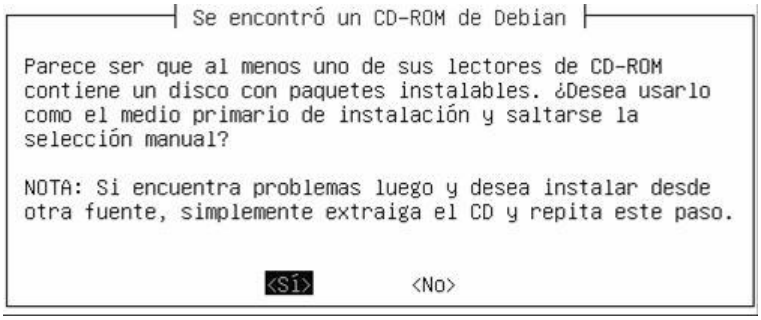


Figura 3.16: Mensaje de que se encontró un CD-ROM con paquetes instalables. **[45]**

En seguida se mostrará la ventana que contiene el medio de instalación. Se elige la opción *cdrom* (ver Figura 3.17), enter para continuar:

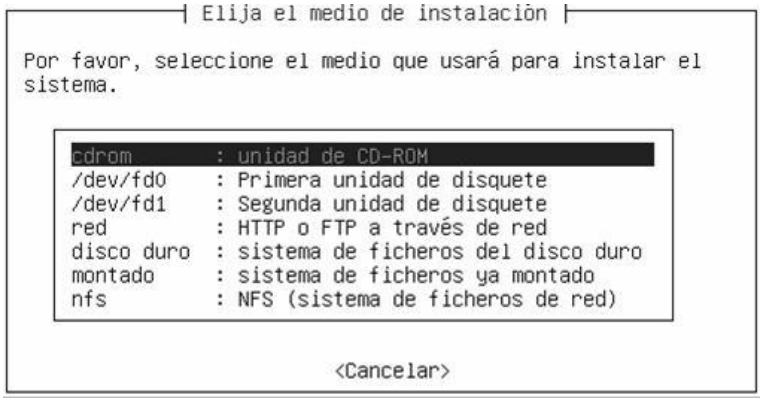


Figura 3.17: Selección del medio para instalar el sistema. **[45]**

Paso 23: Menú principal de Debian GNU/Linux, con la opción *Configurar los controladores de dispositivos* (enter para continuar).

Paso 24: Ventana que muestra los comentarios de los controladores ya cargados, se selecciona la opción *<Siguiente>* (enter para continuar)

Paso 25: Ventana para seleccionar las distintas categorías, se selecciona *<Ok>* (ver Figura 3.18).

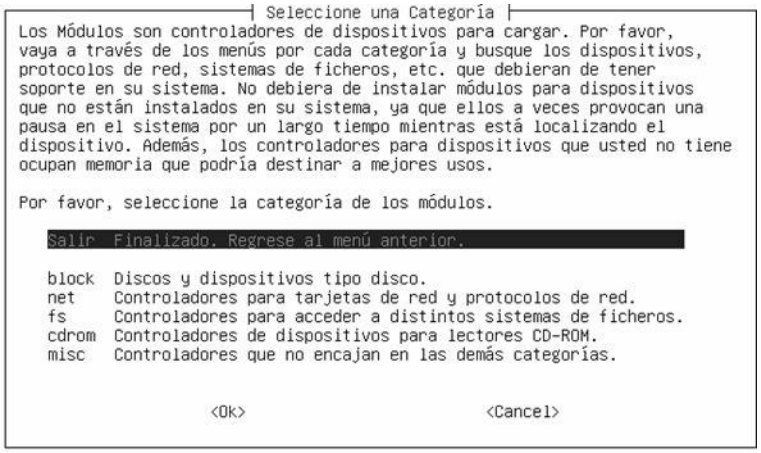


Figura 3.18: Seleccionando una categoría. **[45]**

Paso 26: Menú principal de Debian GNU/Linux, con la opción *Configurar la red* (enter para continuar). Esta opción aparece solo si existe algún tipo de hardware para la red "tarjeta de red", si no se detecta nada se pasa al paso 28.

Paso 27: Ventana la cual pide que sea escrito el nombre del host o computadora (ver Figura 3.19). Teclear el nombre y aceptar.

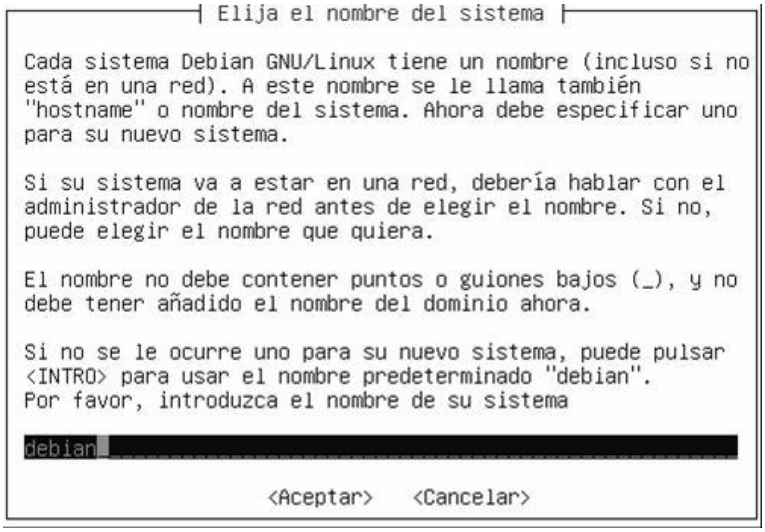


Figura 3.19: Tecleando el nombre del host. **[45]**

Paso 28: Ventana para continuar con la configuración automática de la red. Esta ventana tiene dos opciones, si se elije la opción *<Si>,* se necesitara un servidor DHCP o BOOTP, para que se pueda configurar automáticamente. En caso de que no se cuente con dichos servidores se elije la opción *<No>* para configurar la red manualmente. Selección de la opción *<No>* (enter para continuar).

Después de seleccionar la opción mencionada se mostraran las siguientes ventanas, donde se podrá escribir la dirección y nombre correcto. **[45]**

Escoja la dirección IP.
192.168.1.1
Escoja la mascara de red.
255.255.255.0
¿Cuál es la dirección IP de su pasarela?
192.168.1.2
Escoja el nombre de dominio.
Bienvenido
Escoja la dirección del servidor DNS.
192.168.1.2

Paso 29: Ventana con el menú principal de Debian GNU/Linux, donde muestra la opción, *Configurar el nombre del sistema* (enter para continuar).

Paso 30: Aparecerá una ventana para elegir el nombre del sistema o hostname, donde se tecleara el nombre y se selecciona *<aceptar>*.

Paso 31: Se muestra el menú principal de Debian GNU/Linux, con la opción *Instalar el sistema base* (enter para continuar).

Paso 32: Nuevamente se muestra el menú principal de Debian GNU/Linux, con la opción *Hacer el sistema arrancable* (enter para continuar).

Paso 33: Ventana que muestra donde se debe de instalar el sistema de arranque LILO (ver Figura 3.20) enter para continuar.

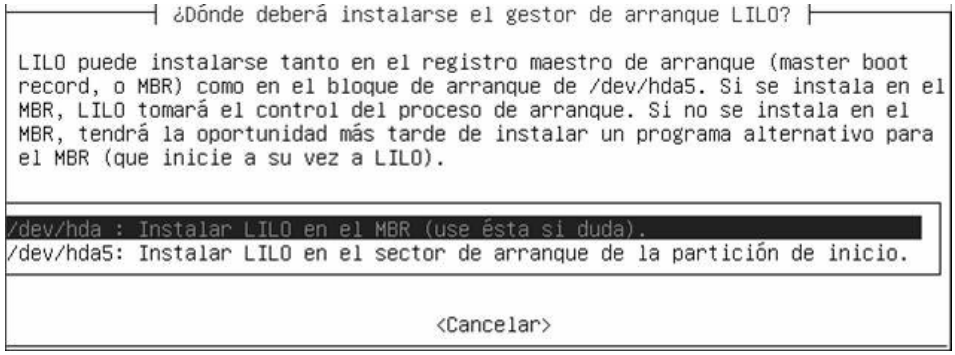


Figura 3.20: Selección de la partición donde se desea instalar LILO. **[46]**

Paso 34: Ventana que muestra otras particiones que pueden arrancar. Se escoge la opción:

Incluir: Poner todo en el menú.

Paso 35: Ventana para asegurar LILO. Se selecciona la opción *<Continuar>* (enter para seguir adelante).

Paso 36: Menú principal de Debian GNU/Linux, con la opción *Crear un disquete de arranque* (enter para continuar). Opción que se puede saltar, en dado caso que se

quiera crear el disco solo bastará con insertar el disquete en la computadora para iniciar el proceso.

Paso 37: Menú principal de Debian GNU/Linux, con la opción *Reiniciara el sistema* (enter para continuar).

Paso 38: Ventana con las opción de confirmar el reinicio del sistema, se selecciona la opción *<Continuar>*.

Paso 39: Después de reiniciar el sistema se mostrará el menú de LILO (ver Figura 3.21), donde se podrá escoger el sistema operativo con el que se desee iniciar (si se cuenta con Debian GNU/Linux y otro sistema operativo Windows). Cabe mencionar que hasta estos momentos solo se ha instalado un sistema mínimo o básico de Debian GNU/Linux.

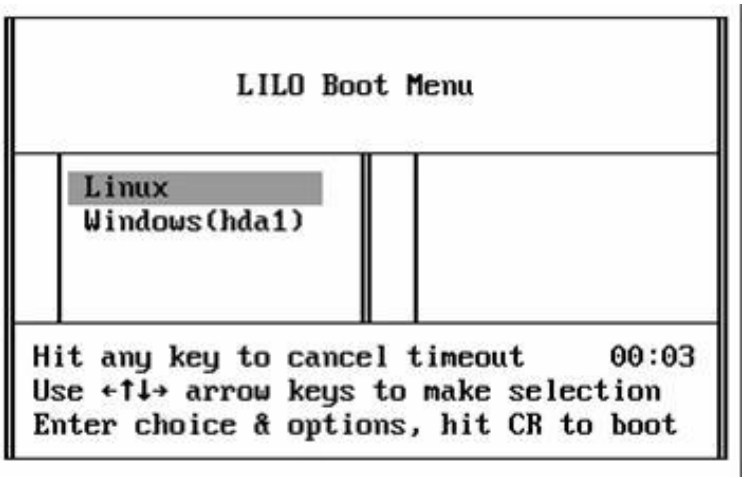


Figura 3.21: Menú de LILO. **[46]**

Paso 40: Continuando con la instalación de Debian GNU/Linux, se mostrara la ventana de configuración del sistema Debian, aquí se elige la opción *<Ok>* (ver Figura 3.22).

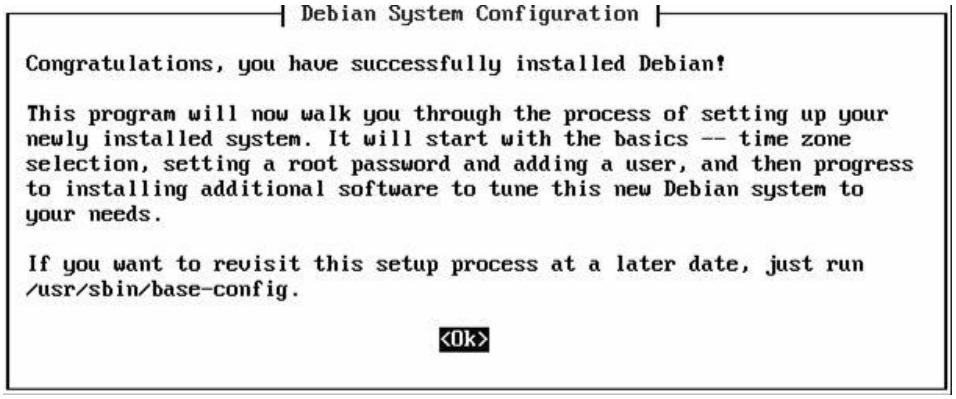


Figura 3.22: Ventana de configuración del sistema Debian. **[46]**

Paso 41: Ventana para configurar la zona horaria del sistema. Seleccionar la opción *<Yes>* (enter para continuar).

Paso 42: Ventana para realizar la configuración de la zona horaria. Elegir la opción *América* posteriormente se selecciona *<Ok>*.

Paso 43: Ventana para la configuración de la zona horaria. Escoger la opción *Mexico_City,* después se selecciona *<Ok>*.

Paso 44: Ventana *Password setup,* permite aceptar la opción de anexar la contraseña a Shadow (guarda las contraseñas cifrándolas con el *algoritmo MD5*, para que nadie pueda acceder a ellas). Se selecciona la opción *<Yes>.*

Paso 45: Nuevamente se muestra la ventana *Password setup,* solo que esta ventana pide que se introduzca la contraseña de *root o administrador* (ver Figura 3.23). Se teclea la contraseña y se elige la opción *<OK>*.

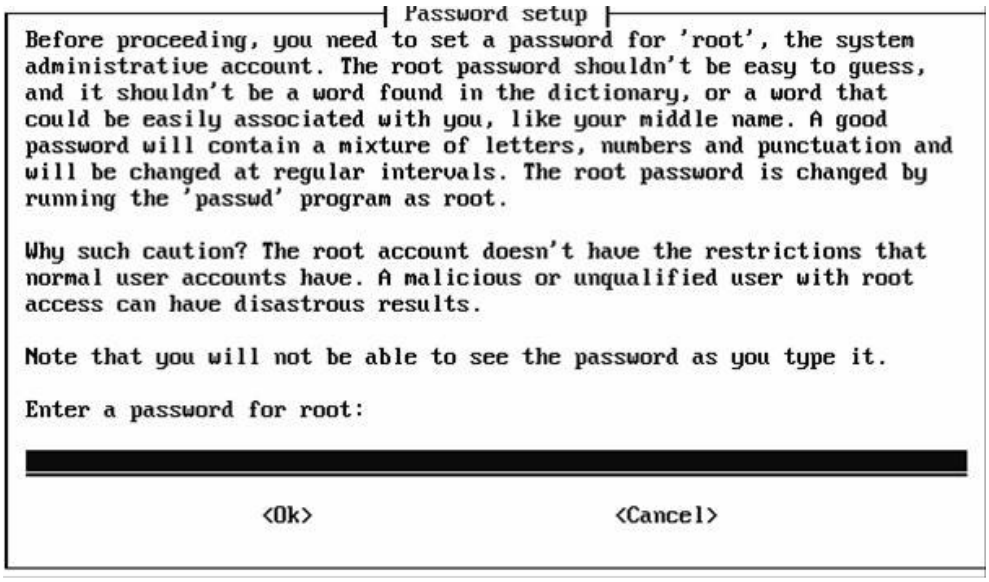


Figura 3.23: Tecleando la contraseña de administrador. **[46]**

Paso 46: La siguiente ventana de *Password setup,* es para verificar la contraseña del administrador. Teclear la contraseña y seleccionar la opción *<Ok>*.

Paso 47: Ventana *Password setup* (ver Figura 3.24), para crear un usuario normal. Se selecciona la opción *<Yes>*.

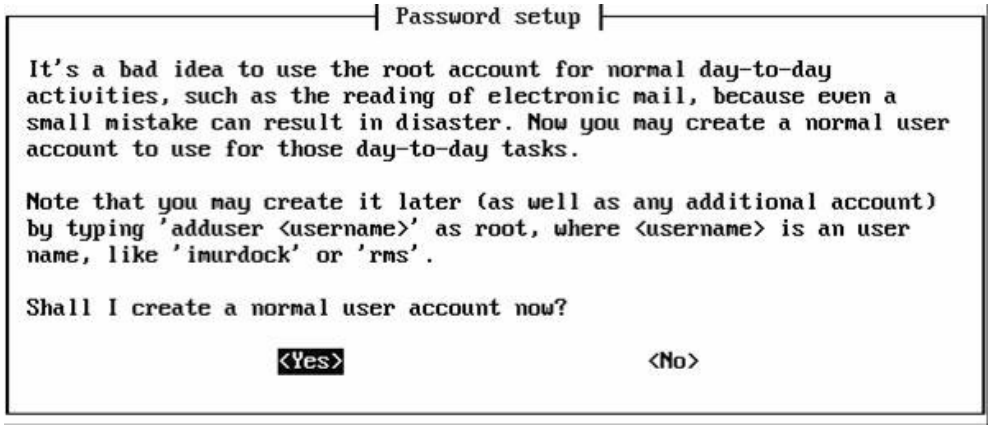


Figura 3.24: Creación de un usuario nuevo. **[46]**

Paso 48: Ventana *Password setup,* que solicita el nombre del usuario nuevo. Se escribe el nombre del usuario y se selecciona la opción *<Ok>* (enter para continuar). Por ultimo el sistema pedirá que se introduzca la contraseña para el usuario creado (opcional).

Paso 49: Nuevamente aparece la ventana de configuración del sistema Debian, aquí el sistema pedirá si se quieren eliminar los paquetes de PCMCIA, esto es opcional, ya que no pasa nada si se dejan o se eliminan.

Paso 50: Ventana para iniciar con la configuración de Apt (ver Figura 3.25), en esta ventana se puede elegir la opción *<Yes>*, si esta conectada la computadora a Internet para descargar paquetes vía FTP. En este caso se seleccionara la opción *<No>*, dado que la computadora no esta conectada a Internet.

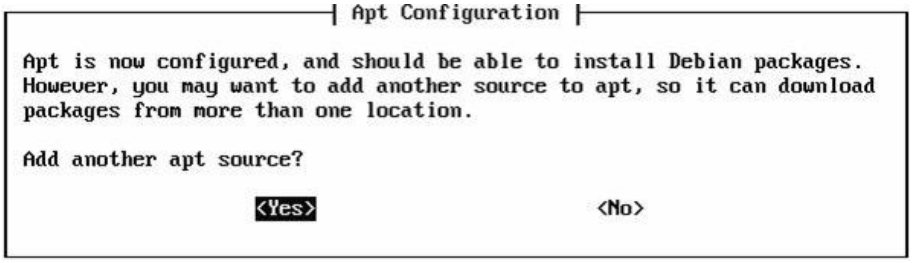


Figura 3.25: Configurando APT. **[46]**

Después de esta ventana, aparecerán otras dos ventanas de *Apt Configuration*, la primera preguntara si se quiere agregar otro *source (código)*, se selecciona la opción *<No>*. La segunda ventana pedirá si se quiere tener la actualización de http://security.debian.org/ se opta por la opción *<No>*.

Paso 51: Ventana de configuración del sistema Debian (ver Figura 3.26), esta ventana brinda la opción para ejecutar la ventana *Tasksel* (permite escoger los programas que se quieren instalar desde una lista).

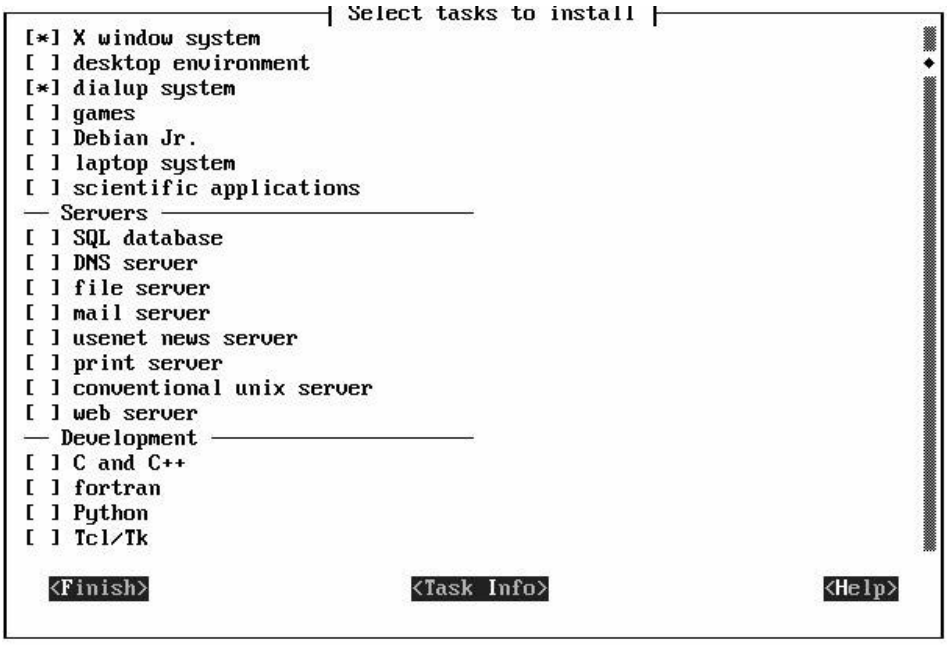


Figura 3.26: Ventana Tasksel. **[46]**

Paso 52: Ventana de configuración del sistema Debian, para quitar los paquetes seleccionados. Se selecciona la opción <*No*>.

Paso 53: Este paso es importante ya que aparecerán dos pantallas las cuales muestran una lista de los paquetes que se instalaran y los que se desinstalaran.

Paso 54: Después de ver los paquetes que se instalaran, se mostrara una pantalla de color negro con el mensaje siguiente:

Do you want to continue? [Y/n]

Se teclea la letra *Y,* después se pulsa enter para continuar. En ocasiones aparece el siguiente mensaje en la misma pantalla:

Media change: Please insert the disc labeled 'Debian GNU/Linux 3.0' r1_wody_official i386 binary-1 (20021218) in the drive' /cdrom/' and presse enter.

Se presiona enter para seguir con la ejecución del proceso.

Paso 55: Ventana para realizar la configuración local del idioma local para el sistema. Teclee la opción <yes>. Se mostraran dos ventanas mas para la configuración local, en la primera se selecciona la opción "es_ES ISO-8859-1" y se elije <*Ok*>. En la segunda se opta por la opción "es_ES" y se selecciona <*OK*> (enter para continuar). **[46]**

Paso 56: Ventana para poder configurar la X Window (ver Figura 3.27), para usar Linux en modo grafico. Se selecciona la opción "*vesa*" (enter para continuar).

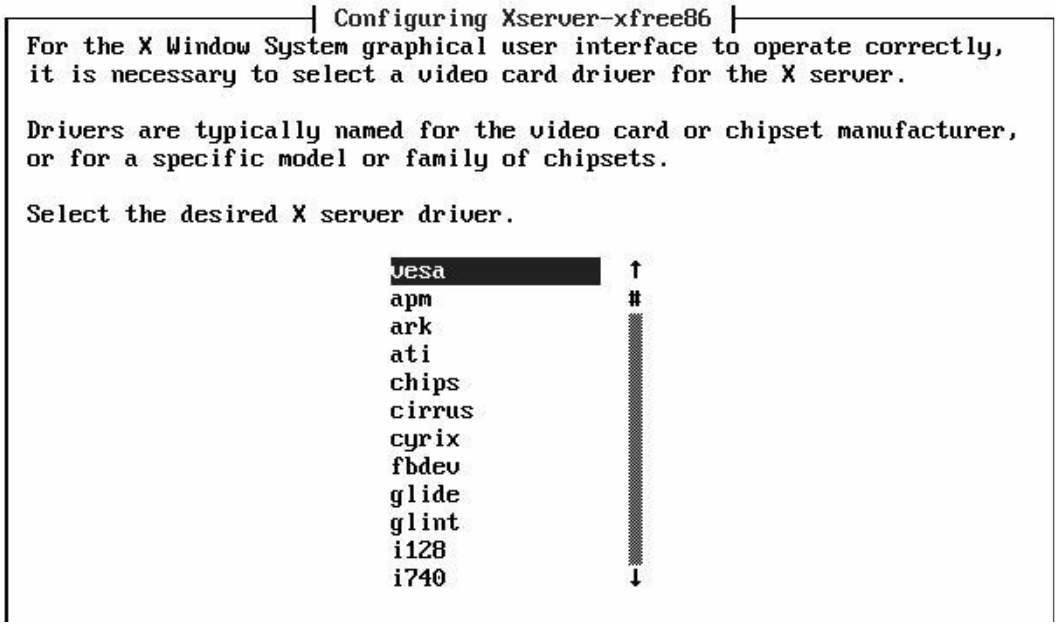


Figura 3.27: Configurando X Window. **[47]**

Paso 57: Ventana que muestra la posibilidad de que X Window funcione directamente (ver Figura 3.28), para esto se selecciona <*Yes*>.

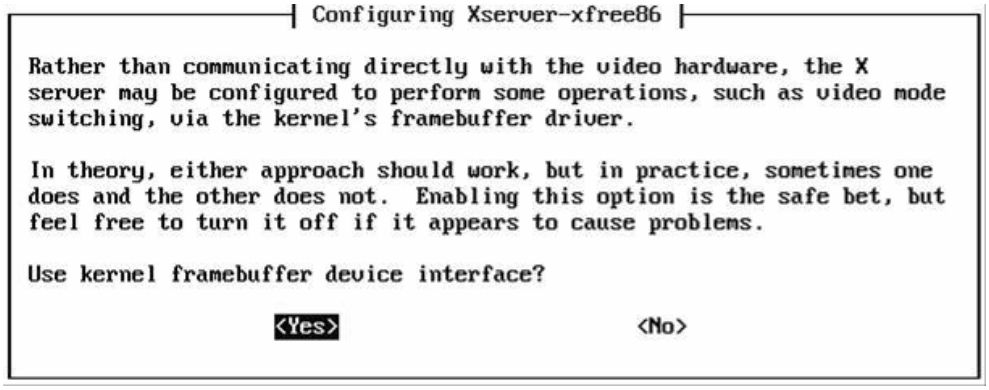


Figura 3.28: Selección de X Window, para que funcione directamente. **[47]**

Paso 58: En esta ventana se elige el teclado correcto, se deben de tener presentes las características de este para elegir el teclado correcto (ver Figura 3.29). Se selecciona "xfree86", <*Ok*>.

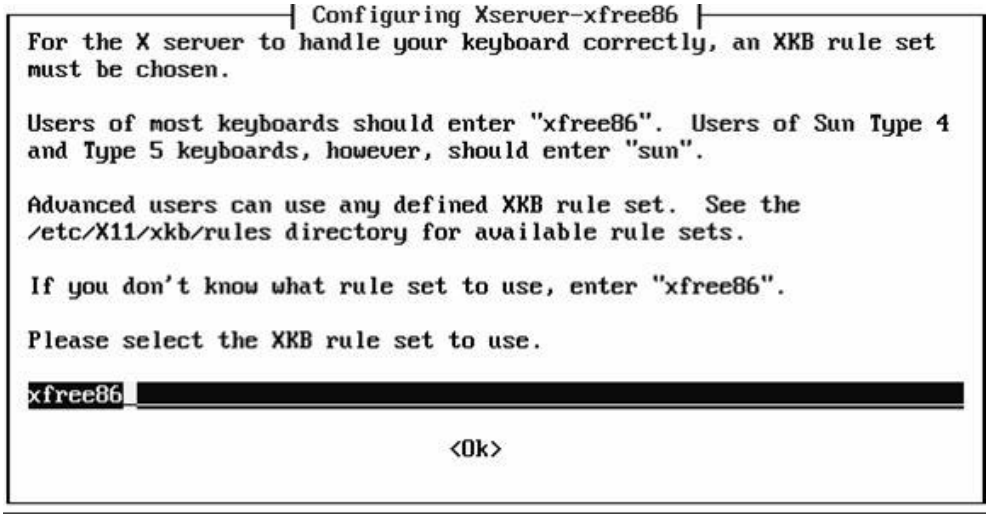


Figura 3.29: Seleccionando el teclado correcto. **[47]**

Paso 59: Ventana para confirma la disposición del teclado. Se selecciona *<Ok>* (enter para continuar).

Paso 60: Ventana que pedirá que se escriba el número de teclas que contiene el teclado (ver Figura 3.30). Se elige *<Ok>*.

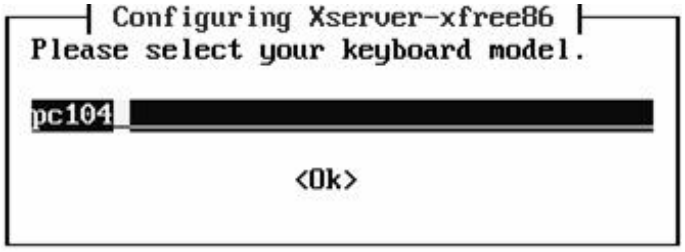


Figura 3.30: Escribiendo el número de teclas del teclado. **[47]**

Paso 61: Ventana, la cual requerirá que se escriba el código del país del teclado. Escribir la opción "*es*" y se selecciona *<Ok>*. Posteriormente se mostrara dos ventanas más, donde se seleccionara *<Ok>*.

Paso 62: Ventana para iniciar con la configuración del ratón (Mouse). En primera instancia se pedirá que se elija el tipo de conexión del ratón "*/dev/psaux*", *<Ok>* (enter para continuar).

Paso 63: Ventana donde se escogerá el tipo de ratón "*PS/2*" y se selecciona *<Ok>* (enter para continuar).

Paso 64: Ventana para iniciar con la configuración del monitor, esta ventana preguntará si se tiene un monitor LCD, si se tiene se selecciona la opción *<Yes>.* En este caso se selecciona la opción *<No>* (ver Figura 3.31).

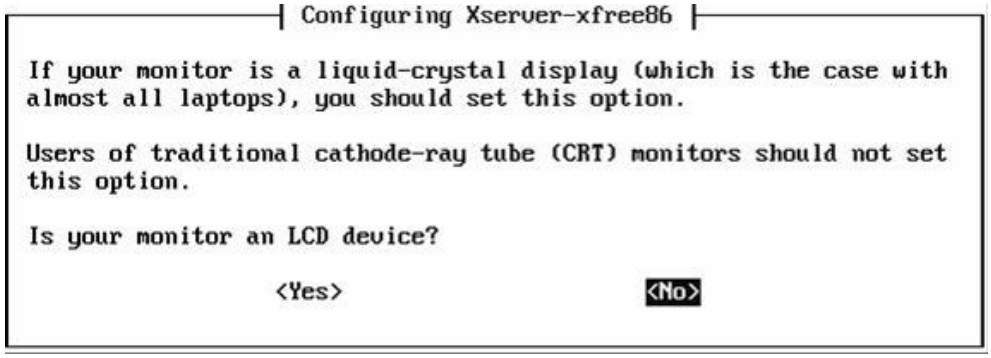


Figura 3.31: Inicio de la configuración del monitor. **[47]**

Paso 65: Continuando con esta configuración, se mostrará una nueva ventana con las siguientes opciones:

- ✓ Advanced: Se debe de conocer la medida de pantalla, resolución máxima y las frecuencias de refresco máximas del monitor.
- ✓ Médium: Se debe de conocer la medida y la resolución del monitor (ejemplo: 1024x768 a 70Hz).
- ✓ Simple: Sólo se debe indicar la medida del monitor.

En esta ventana se seleccionara "*Advanced*", enter para continuar.

Paso 66: Por último se mostrara una ventana (ver Figura 3.32), donde el programa da una felicitación por haber instalado correctamente Debian GNU/Linux.

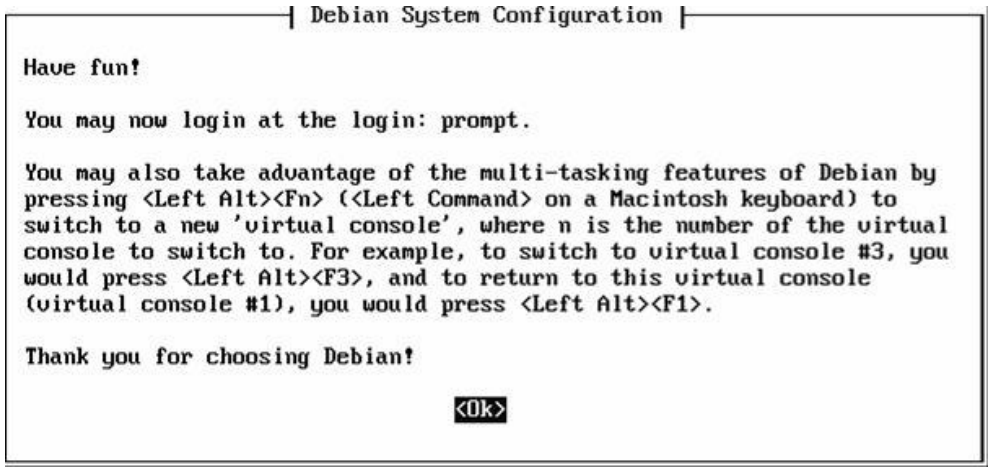


Figura 3.32: Ventana de felicitación por haber elegido Debian GNU/Linux. **[47]**

Después de dar *<Ok>* en esta ventana, se mostrara la pantalla para iniciar el "Login" he introducir la contraseña (password) para comenzar a utilizar Debian GNU/Linux (ver Figura 3.33). **[47]**

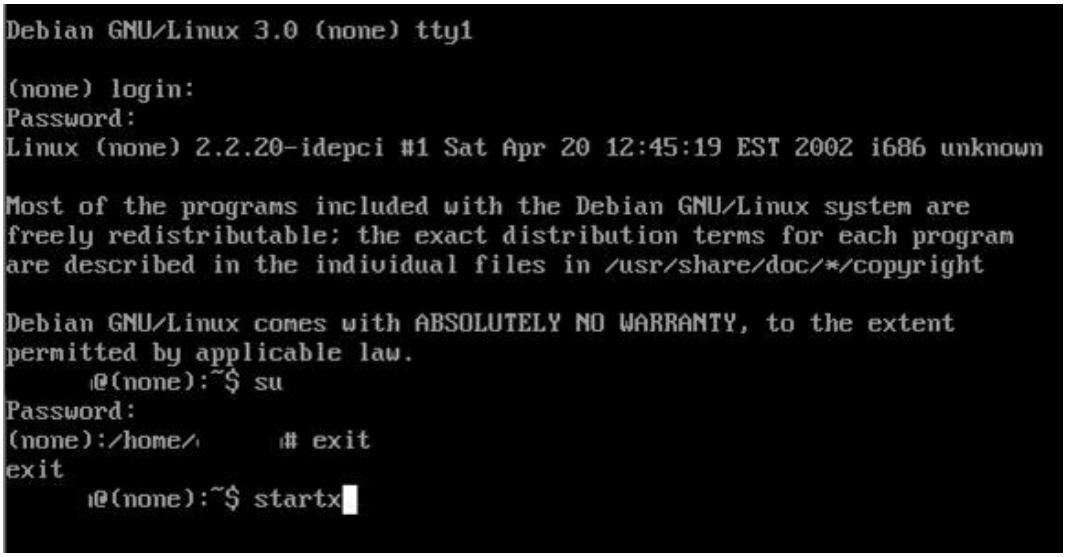


Figura 3.33: Iniciando el uso de Debian GNU/Linux. **[47]**

Capítulo

Servidor HTTP Apache

Apache en la actualidad es uno de los servidores más populares y de altas prestaciones de HTTP (*HyperText Transfer Protocol*). Apache tiene un diseño modular y soporta extensiones dinámicas de módulos durante su ejecución. Es muy fácil de configurable en muchos servidores, debido a que cuenta con módulos disponibles para lograr un soporte de varios mecanismos de autentificación, control de acceso, proxy, servidores virtuales, etc. Con módulos incluidos en Debian es posible instalar varios tipos de lenguajes de programación como PHP3, Perl, Java Servlets, etc.

En este capítulo se redacta el surgimiento de Apache y los principales pasos para poder adquirirlo o descargarlo desde su página principal de Internet, también se hace mención de las directivas del servidor Apache que deben ser configuradas por el usuario, estas directivas son básicas para tener un funcionamiento básico y adecuado del servidor.

4.1. Apache

Antes de que surgiera el nombre del servidor Apache, en febrero de 1995, existía un servidor Web desarrollado por NCSA (National Center for Supercomputing Applications en la Universidad de Illinois), este servidor era de dominio publico, pero en 1994 Rob McCool, su principal desarrollador había dejado este proyecto, desde entonces se dejo a este servidor en manos de los responsables del sitio Web, aportando avances para dicho servidor. Más tarde se creo un correo electrónico para tener una herramienta donde se pudieran aportar avances para dicho servidor, en forma de parches, después de esto los desarrolladores Brian Behlendorf y Cliff Skolnick, colocaron un servidor de información compartida en California, para poder contribuir con las mejora del servidor. Pasando esto surge el *Grupo Apache* formado por los ocho principales programadores de la Web, usando el servidor NCSA 1.3 como base de trabajo, agregando las mejoras al servidor y evitando errores publicados.

En abril de 1995 se publica la primera versión oficial de Apache 0.6.2. Surgiendo dos grupos diferentes de desarrolladores los cuales uno de ellos siguió con el desarrollo de apache 0.6.2, produciendo la versión 0.7, el otro grupo opto por reescribir todo el código de apache, formando una arquitectura modular. En julio de 1995 ambos grupos unen lo desarrollado para publicar Apache 0.8. Fue hasta el primero de diciembre de 1995 cuando aparece la versión de Apache 1.0, la cual incluía la documentación y mejoras en formas de módulos incrustables, mas tarde Apache era un servidor superior al servidor NCSA. Así en el año de 1999, los miembros del *Grupo Apache* fundan al *Apache Softwar*. **[48]**

Apache Software: Fundación, que provee soporte legal y financiero al desarrollo del servidor Apache y los proyectos laterales que han surgido de éste.

4.2. Servidor Apache

Apache es un servidor HTTP para diferentes sistemas operativos como; Unix, GNU/Linux, Windows, etc. Siendo así un servidor Web gratuito, potente, fácil de manejar y de configurar, se puede decir que este tipo de software es una de las mejores creaciones del software libre. Ya que Apache permite el hospedaje de páginas Web en la computadora donde se tenga instalado este servidor. **[49]**

4.3. Algunas características de Apache

- ✓ Funciona sobre muchas plataformas (muchos sabores de Unix, Linux, Vms, Win32,OS2)
- ✓ Modulos cargados dinámicamente.
- ✓ CGI, Perl (Formularios, diccionarios en línea, etc).
- ✓ Php + Bases de datos
- ✓ SSL: transacciones seguras.
- ✓ Soporte para host virtuales.
- ✓ Alto desempeño. **[50]**

4.4. Obteniendo Apache

La adquisición de Apache es muy sencilla ya que se encuentra en la dirección de Internet http://httpd.apache.org/, para esto se mencionan una serie de pasos para obtenerlo desde Internet.

Paso 1: Solo basta con teclear la dirección de la página principal de Apache. Donde se podrá seleccionar, dando un click en la liga *Download* (ver Figura 4.1). **[51]**

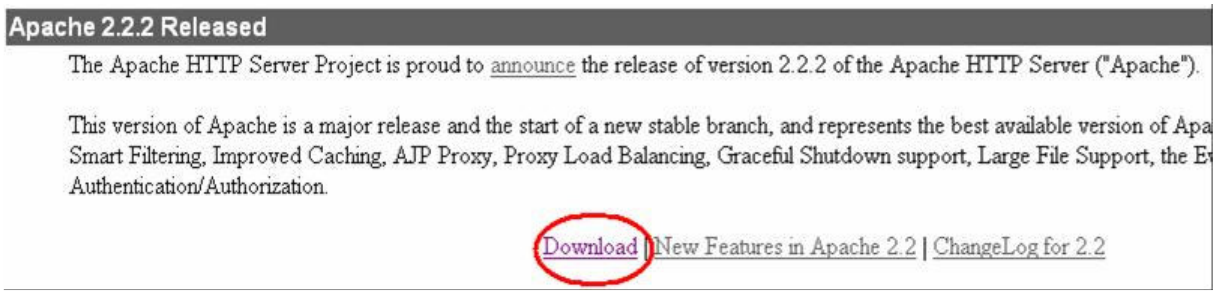


Figura 4.1: Obteniendo Apache. **[51]**

Paso 2: La liga anterior abrirá una nueva ventana, donde se mostraran los archivos httpd-2.2.2.tar.gz y PGP (ver Figura 4.2), para poder ser descargado (dar click). **[52]**

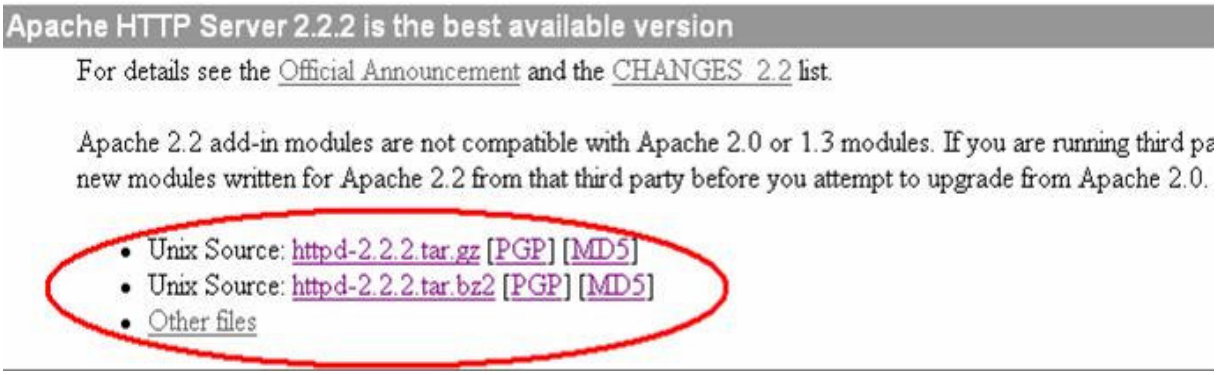


Figura 4.2: Descarga del archivos Apache. **[52]**

4.5. Instalación

El primer pasó para poder realizar una buena instalación del servidor Apache, se debe revisar que los archivos descargados estén completos y sin modificación alguna, comprobando el archivo *.tgz* con su firma *PGP*, después de haber hecho esto se continuara con los siguientes pasos:

httpd-2.2.2.tar.gz
httpd-2.2.2.tar

Paso 1: Descomprimir el archivo para crear un directorio donde se localizaran las fuentes del servidor, para esto solo basta con ejecutar los comandos siguientes:

```
gunzip httpd-version.tar.gz
tar xvf httpd-version.tar
```

Paso 2: Creación de enlace al directorio que se creo, después de descomprimir:

```
ln -s /usr/local/httpd-version httpd
```

Paso 3: Se accede al directorio *httpd,* con:

```
cd httpd
```

Paso 4: En este paso se ejecutara el script configure, para realizar la configuración de Apache, de la manera siguiente:

```
./configure --prefix=PREFIX
        o
./configure --prefix=/usr/local/httpd
```

Paso 5: En caso de que se desee compilar el archivo antes de instalarlo se ejecuta el siguiente comando:

```
make
```

Paso 6: Para poder instalar apache solo será necesario de especificarlo de la siguiente manera:

```
make install
```

Si la instalación ha sido correcta se verá el siguiente mensaje:

```
You now have successfully built and installed the Apache 2.0 HTTP Server. To
verify that Apache actually works correctly you now should first check the
(initially created or preserved)configuration files

    /usr/local/httpd/conf/httpd.conf

And then you should be able to immediately fire up Apache the first time by
running:

    /usr/local/httpd/bin/apachectl start

Thanks for using Apache.                    The Apache Group
                                             http://www.apache.org/
```

Antes de arrancar el servidor, se pueden modificar algunas directivas del fichero de configuración httpd.conf, como se muestra a continuación:

```
ServerAdmin webmaster@midominio.es
ServerName www.midominio.es
Port 80
```

Paso 7: En este paso se especifica la manera, de como se puede personalizar el servidor Apache, editando los ficheros de configuración localizados en PREFIX/conf/, como se muestra a continuación:

```
vi PREFIX/conf/httpd.conf
```

Paso 8: Paso para poder comprobar que la configuración se ha realizado de manera correcta:

```
/usr/local/httpd/bin/apachectl configtest
Sintaxis OK
```

Nota: Cabe mencionar que en algunos libros el directorio *httpd* puede aparecer con el nombre de *apache*. **[L]**

4.5.1. Comprobación de la instalación

Para poder realizara la comprobación de que el servidor Apache funciona, se tiene que iniciar el servidor, usando el comando:

```
prefix/bin/apachectl start
```

Después de haber iniciado el servidor Apache, se podrá acceder al documento que se haya especificado por defecto en *DocumentRoot,* usando la dirección:

```
http://localhost/.
```

Para poder detener el servidor solo se teclea el comando:

```
prefix/bin/apachectl stop
```

Con esto se podrá verificar si el servidor Apache funciona correctamente. **[53]**

4.5.2. Configuración de Apache

Cabe mencionar que al instalar el servidor Apache, no se realizan las configuraciones adecuadas, por esta razón se mencionaran en los siguientes tres

temas las directivas y ficheros mas importantes que deben ser configurados.

4.5.2.1. Directivas principales

ServerAdmin: Directiva encargada de especificar la dirección de correo electrónico del administrador de dicho servidor web. Esta directiva permite al usuario enviar un recado al administrador, cuando ocurren mensajes de error al acceder al servidor. **[54]**

Ejemplo:

```
ServerAdmin yo@mi.casa
```

ServerName: Esta directiva esta encargada de define el nombre y el puerto TCP que el servidor utilizara para identificarse. **[55]**

Ejemplo:

```
ServerName nombre o dirección:puerto
ServerName www.uoc.edu:80
ServerName 192.168.1.1:80
```

DocumentRoot: Directorio raíz donde se encuentran los documentos de páginas Web. Localizado en el directorio *htdocs*.

Ejemplo:

```
DocumentRoot "/var/www/html"
```

DirectoryIndex: Directiva encargada de declarar un archivo *.html* cuando no se especifica una dirección URL, el servidor por defecto declara index.html. **[54]**

Ejemplo:

```
DirectoryIndex index.html
www.mex.edu
www.mex.edu/index.html
```

AccessFileName: Especifica el nombre de fichero de configuración, por defecto se define .htaccess.

Ejemplo:

```
AccessFileName nombre_del_archivo [nombre_del_archivo] ...
AccessFileName .htaccess
```

ErrorDocument: Directiva encargada de establecer la configuración del servidor en caso de que ocurra algún error. Se puede configurar el servidor para hacer alguna de las siguientes cosas: **[56]**

- ✓ Mostrar un mensaje de error estándar.
- ✓ Devolver un mensaje de error personalizado.
- ✓ Redirigir una petición a una ruta-URL local.
- ✓ Redireccionar la petición a una URL externa.

Ejemplo:

```
ErrorDocument error mensaje_de_error
ErrorDocument 404 /noencont.html.
ErrorDocument 403 "Lo sentimos no podemos permitirle el acceso a esta página hoy"
```

Alias y AliasMatch: Permiten definir el acceso a directorios que se encuentran fuera de DocumentRoot. **[55]**

UserDir: Facilita la habilitación de peticiones a páginas de usuarios. **[54]**

Ejemplo:

```
UserDir public_html
```

4.5.2.2. Directivas globales

ServerRoot: La presente directiva se encarga de especificar el directorio en el que ha sido instalado el servidor web. Este directorio tendrá los subdirectorio conf/ y logs.

Ejemplo:

```
ServerRoot directorio_o_dirección
ServerRoot /usr/local/apache
ServerRoot /home/httpd
```

KeepAlive: Directiva encargada de facilitar la posibilidad de que se establezcan sesiones HTTP de larga duración, permitiendo enviar varias peticiones sobre la misma conexión TCP. **[56]**

Ejemplo:

```
KeepAlive On|Off
KeepAlive On
```

Listen: Esta directiva conecta el servidor Web a una dirección IP y número de puerto. Por defecto se conectara por medio del puerto 80 de TCP.

Ejemplo:

```
Listen dirección_IP:Puerto
Listen 172.20.30.40:80
Listen 172.20.30.40:8080
```

LoadModule: Esta directiva permite instalar los módulos adicionales en el servidor Web, siempre y cuando este instalado *mod_so*. **[54]**

Ejemplo:

```
LoadModule userdir_module modules/mod_userdir.so
```

4.5.2.3. Directivas para la configuración de ficheros

<Directory>: Comprende un conjunto de directivas que se aplican solamente al directorio del sistema de ficheros especificado y a sus subdirectorios.

Ejemplo:

```
<Directory directorio_o_dirección> ... </Directory>
<Directory /usr/local/httpd/htdocs>
```

<DirectoryMatch>: Incluye las directivas que se aplican a los directorios y subdirectorios del sistema de ficheros que equivalen a una expresión regular.

Ejemplo:

```
<DirectoryMatch regex> ... </DirectoryMatch>
<DirectoryMatch "^/www/.(.+)?[0-9]{3}">
```

<Files>: Contiene directivas que limitan el ámbito de aplicación de las directivas que se emplean en los ficheros cuyos nombres coincidan con los especificados.

Ejemplo:

```
<Files nombre_del_archivo> ... </Files>
<Files ~ "\ .(gif|jpe?g|png)$">
```

<FilesMatch>: Encargada de limitar el rango de las directivas incluidas según el nombre de los ficheros, también acepta expresiones regulares.

Ejemplo:

```
<FilesMatch regex> ... </FilesMatch>
<FilesMatch "\ . (gif|jpe?g|png)$">
```

<Location>: Proporciona un control de acceso de los ficheros por medio de la URL.

Ejemplo:

```
<Location URL-path|URL> ... </Location>
<Location ~ "/(extra|special)/data">
```

<LocationMatch>: Aplica las directivas URLs que tengan similitud con expresiones regulares que se especifiquen.

Ejemplo:

```
<LocationMatch regex> ... </LocationMatch>
<LocationMatch "/(extra|special)/data">
```

<VirtualHost>: Contiene las directivas que se aplicarán a las peticiones que vayan dirigidas a cierto *host, como;* nombre de servidor, dirección IP o puerto TCP. **[56]**

Ejemplo:

```
<VirtualHost addr[:port] [addr[:port]] ...> ... </VirtualHost>
<VirtualHost 10.1.2.3>
ServerAdmin webmaster@host.foo.com
DocumentRoot /www/docs/host.foo.com
ServerName host.foo.com
ErrorLog logs/host.foo.com-error_log
TransferLog logs/host.foo.com-access_log
</VirtualHost>
```

<Proxy>: Esta directiva sólo se aplica a los parámetros que hagan petición al *Proxy* que coincidan con la especificación de URL, para poder manipular esta directiva se tiene que tener instalado *mod proxy*.

Ejemplo:

```
<Proxy:Dirección> .... </Proxy>
<Proxy http://cnn.com/*>
Order allow,deny
Deny from all
</Proxy>
```

<ProxyMatch>: Esta directiva es similar a la directiva *proxy*, pero acepta expresiones regulares en la URL especificada. **[57]**

<IfDefine>: Directiva que será procesada o aplicada si se cumple con cierta condición al iniciar al servidor Web.

Ejemplo:

```
<IfDefine [!] nombre_del_parametro> ... </IfDefine>
httpd -DReverseProxy ...
# httpd.conf
<IfDefine ReverseProxy>
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule proxy_module modules/libproxy.so
</IfDefine>
```

<IfModule>: Directiva aplicada a los parámetros, si al iniciar el servidor el módulo especificado se encuentra cargado con LoadModule. **[56]**
Ejemplo:

```
<IfModule [!] nombre_del_modulo> ... </IfModule>
```

4.6. Iniciar, parar y reiniciar Apache

4.6.1. Iniciar Apache al iniciar el sistema

En primera instancia para lograr la ejecución de Apache se debe de contar con permisos de root o administrador.

En caso de que se desee ejecutar directamente el archivo *httpd*, se recomienda usar el script de control *apachectl* (toma los argumentos start, restart, y stop), el cual tiene como función asignar valores a las variables de entorno requeridas para que se ejecute correctamente *httpd* en el sistema y después llamar el binario *httpd. Apachectl*, pasando a *httpd* cualquier argumento que se le pase a través de la línea de comandos, tal que cualquier opción de *httpd* también se puede usar con *apachectl*.

Otra forma de iniciar el servidor apache después de reiniciar el sistema operativo, será necesario anexar una llamada a *apachectl* en los archivos de arranque. Logrando la ejecución de apache como root. **[58]**

4.6.2. Parar y reiniciar Apache

Para lograr esta operación será necesario de enviar una señal al proceso padre *httpd* que se encuentra en ejecución. Para conseguir el envió de mensajes se puede hacer uso del *comando kill* y *línea de comandos*, estos se menciona a continuación:

Comando kill: Este comando envía señales de tipo TERM, HUP, y USR1 directamente al proceso padre.

Ejemplo:

```
kill -TERM `cat /usr/local/apache2/logs/httpd.pid`
```

Línea de comandos: La línea de comandos permite enviar mensajes *(-k: stop, restart, y graceful*) al script de control *apachectl* para que después sean pasados a *httpd*.

Parar apache con la señal TERM o stop: La cual indica que inicie a cerrar los procesos hijos y finalmente cerrar el proceso padre (httpd):

```
apachectl -k stop
```

Parar apache con la señal USR1 o graceful: Le indica al proceso padre que le diga a los procesos hijos que finalicen después de ser finalizada la petición que tienen en ese momento.

```
apachectl -k graceful
```

Reiniciar apache con la señal HUP o restart: Envían una señal al proceso padre para que los procesos hijos finalicen. Esta opción es similar al la señal TERM. **[59]**

```
apachectl -k restart
```

Capítulo

5

Gestor de base de datos MySQL

MySQL es el sistema gestor de bases de datos de código abierto que en la actualidad es muy popular en todo el mundo. Gracias a que MySQL es de código abierto, las empresas que lo usan tienen menos gasto dedicado a este tipo de software en un 90%, y por ser desarrollado bajo la licencia GPL su uso es gratuito. Algunas de sus características más comunes que hacen que este gestor de base de datos sea empleado en las empresas, es su gran velocidad y flexibilidad de ser instalado en diferentes sistemas operativos.

Cuando se elige un gestor de base de datos, se debe de hacer de manera cautelosa de acuerdo a las necesidades requeridas. Por esta razón en el presente capítulo se muestran las características del gestor de base de datos, entre las cuáles se encuentran sus antecedentes históricos, tipos de datos, sentencias SQL, permisos, operaciones con cuentas de usuario, etc.

5.1. Surgimiento de MySQL

MySQL surge en un intento de conectar el sistema gestor de base de datos mSQL a las tablas propias de ISAM, usando sus propias rutinas, después de hacer varias pruebas a mSQL, se concluye que no era lo bastante eficiente para lo que se requería, por tal motivo se inicio el desarrollo de nuevas funciones, surgiendo un interfaz SQL, con interfaz compatible con mSQL.

Con respecto al nombre de MySQL no se tiene bien claro de donde proviene, ya que por diez años las herramientas y bibliotecas han llevado el nombre “my”, sin embargo se menciona que talvez el nombre de MySQL provenga del nombre de la hija del co-fundador Monty Widenius uno de los desarrolladores.

Por otra parte se menciona que el nombre del delfín del logotipo MySQL es “Shaka” (ver Figura 5.1), seleccionado por los fundadores de MySQL AB de una gran lista de nombres, del concurso ``Name the Dolphin''. Este nombre fue enviado por el desarrollador de software Open Source, Ambrose Twebaze de Swaziland, África. **[60]**

Figura 5.1: Logotipo de MySQL. **[60]**

5.2. Ventajas de MySQL

Una de las principales ventajas que tiene MySQL, es de ser un gestor de base de datos de distribución gratuita (licencia GPL). Por esta razón se enlistan las demás ventajas que caracteriza a MySQL sobre otros sistemas gestores de bases de datos:

- ✓ Mayor rendimiento.
- ✓ Mejores utilidades de administración.
- ✓ Integración perfecta con PHP.
- ✓ Sin límites en los tamaños de los registros.
- ✓ Mejor control de acceso de usuarios.
- ✓ Cliente/Servidor.
- ✓ Se pueden crear respaldos sin tener que cerrar los objetos bloqueados por usuarios.
- ✓ Gratuito.
- ✓ Multiplataforma. **[61]**
- ✓ Soporta gran cantidad de tipos de datos para las columnas.
- ✓ Dispone de API's de lenguajes (C, C++, Java, PHP, etc).
- ✓ Gran portabilidad entre sistemas.
- ✓ Soporta hasta 32 índices por tabla.
- ✓ Gestión de usuarios y passwords.
- ✓ Seguridad en los datos. **[62]**

5.3. Tipos de datos

Es importante que al elegir un gestor de base de datos, se tengan presentes los tipos de datos que puedan ser escritos (ver Tabla 5.1), para formar la base de datos y así tener una visión amplia del funcionamiento de la misma.

Tipo	Tamaño	Descripción
CHAR[Longitud]	Longitud bytes	Campo de longitud fija de 0 a 255 caracteres.
VARCHAR(Longitud)	Longitud de cadena + 1 byte	Campo de longitud fija de 0 a 255 caracteres.
TINYTEXT	Longitud de cadena + 1 byte	Cadena con una longitud máxima de 255 caracteres.
TEXT	Longitud de cadena + 2 bytes	Cadena con una longitud máxima de 65.535 caracteres.
MEDIUMTEXT	Longitud de cadena + 3 bytes	Cadena con una longitud máxima de 16.777.215 caracteres.
LONGTEXT	Longitud de cadena + 4 bytes	Cadena con una longitud máxima de 4.294.967.295 caracteres.
TINYINT[Longitud]	1 byte	Rango de -128 a 127 o de 0 a 255 sin signo.
SMALLINT[Longitud]	2 bytes	Rango de -32.768 a 32.767 o de 0 a 65.535 sin signo.
MEDIUMINT[Longitud]	3 bytes	Rango de -8.388.608 a 8.388.607 o de 0 a 16.777.215 sin signo.
INT[Longitud]	4 bytes	Rango de -2.147.483.648 a 2.147.483.647 o de 0 a 4.294.967.295 sin signo.
BIGINT[Longitud]	8 bytes	Rango de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 o de 0 a 18.446.744.073.709.551.615 sin signo.
FLOAT	4 bytes	Número pequeño con un punto decimal flotante.
DOUBLE[Longitud, Decimales]	8 bytes	Número grande con un punto decimal flotante.
DECIMAL[Longitud, Decimales]	Longitud + 1 o Longitud + 2 bytes	Un DOUBLE almacenado como cadena, que permite un punto decimal fijo.
DATE	3 bytes	En el formato AAAA-MM-DD.
DATETIME	8 bytes	En el formato AAA-MM-DD HH:MM:SS.
TIMESTAMP	4 bytes	En el formato AAAAMMDDHHMMSS; el rango aceptable termina en el año 2037
TIME	3 bytes	En el formato HH:MM:SS.
ENUM	1 o 2 bytes	Abreviatura de enumeración, lo que significa que cada columna sólo puede tener un valor entre varios posibles.
SET	1, 2, 3, 4 u 8 bytes	Como ENUM, con la diferencia de que cada columna puede tener más de uno entre un conjunto de valores posibles.

Tabla 5.1: Tipo de datos de MySQL.

A continuación se muestran las categorías principales que se pueden aplicar a toda base de datos, ya que al crearla será necesario de especificar el tipo de información que tendrá cada campo:

- ✓ Texto o cadena de caracteres.
- ✓ Números.
- ✓ Fechas y Horas. **[63]**

5.4. Obtención de MySQL

MySQL se puede obtener fácilmente, gracias a que se encuentra disponible en la página principal de MySQL (ver Figura 5.2). **[64]**

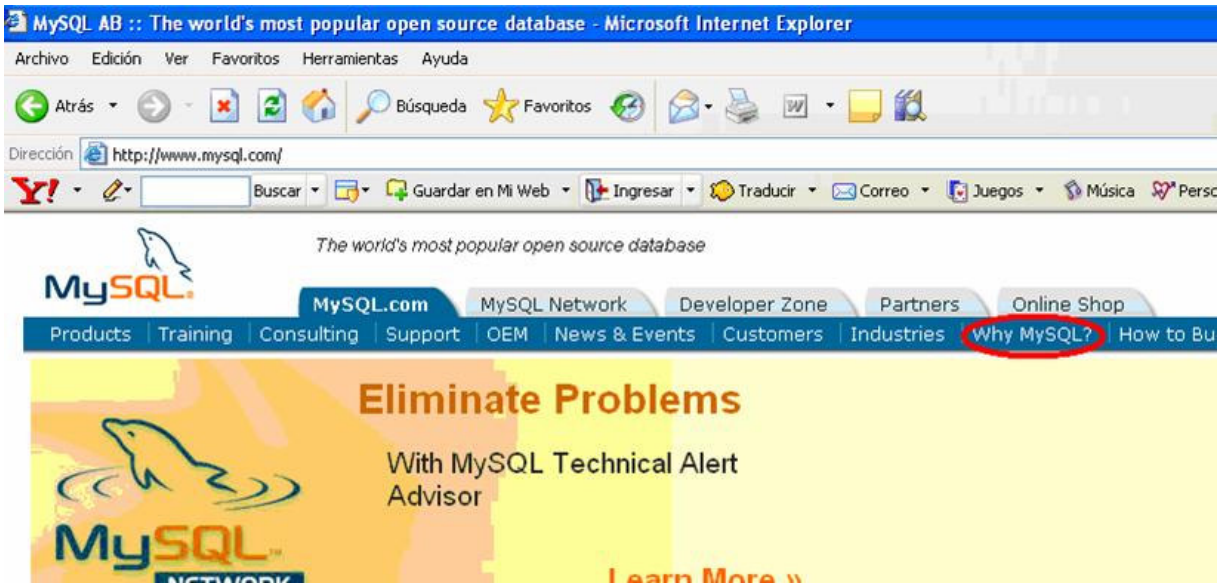


Figura 5.2: Página principal de MySQL. **[64]**

Para continuar con una descarga satisfactoria de MySQL, se mencionan los siguientes pasos:

Paso 1: Seleccionar la liga *Why MySQL?* (ver Figura 5.2), la cual despliega una nueva pantalla, donde mostrara una nueva liga *Whitepaper - MySQL Performance Benchmarks,* dar click para continuar (ver Figura 5.3). **[65]**

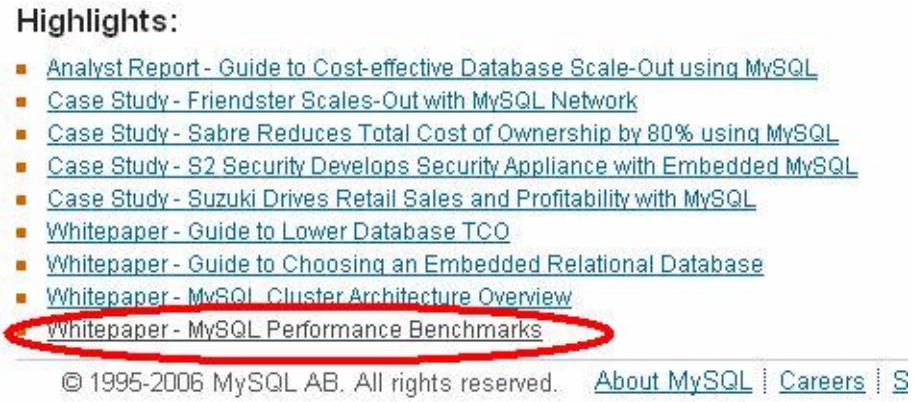


Figura 5.3: Seleccionando la opción Whitepaper-MySQL Performance Benchmarks. **[65]**

Paso 2: La liga anterior abrirá una nueva página, la cual pedirá que se llene una serie de requisitos (ver Figura 5.4). **[66]**

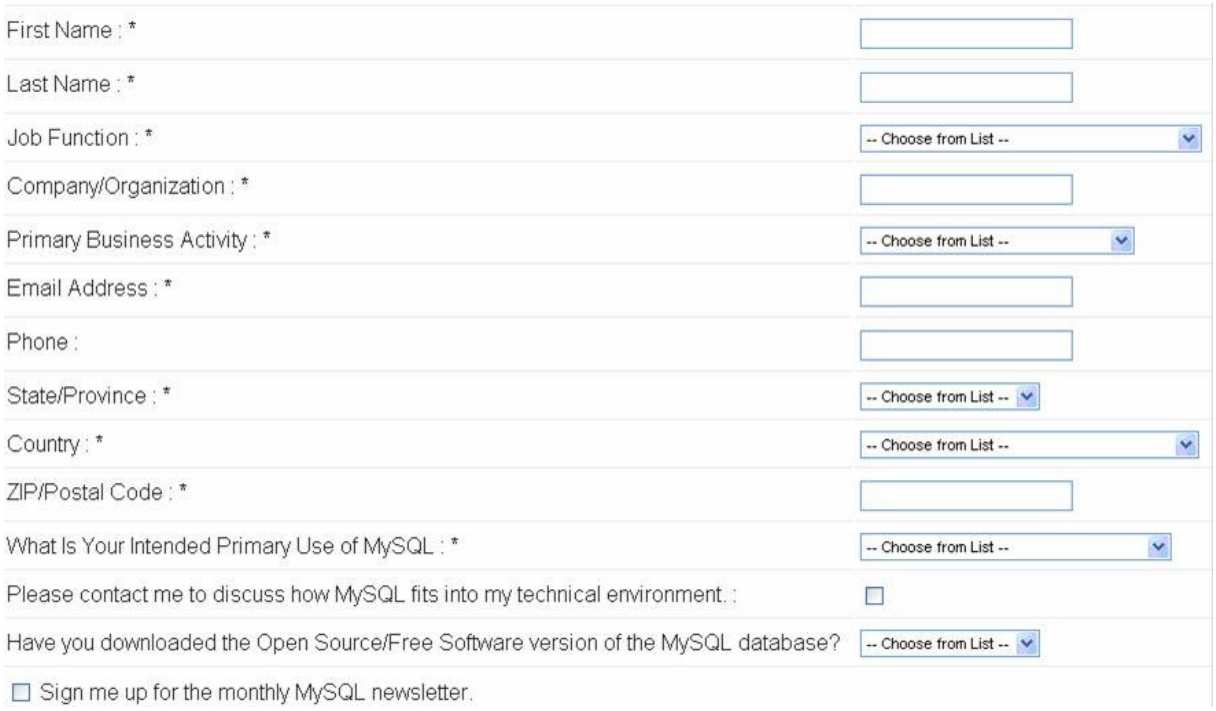


Figura 5.4: Formato de requisitos a llenar. **[66]**

Paso 3: Después de haberse aceptado el llenado del formato de la figura 5.4, se mostrara una página, donde se tendrá que seleccionar la plataforma adecuada para instalar MySQL (ver Figura 5.5), dar click *download*.

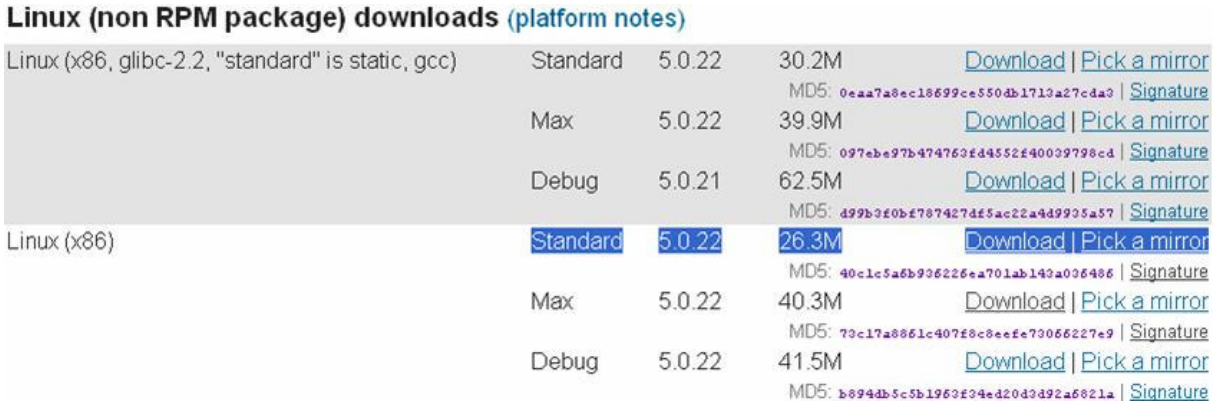


Figura 5.5: Selección de la plataforma de MySQL. **[67]**

Paso 4: Finalmente se descarga el archivo *.tar.gz* y *.tar*. **[67]**

5.5. Pasos para la instalación de MySQL

MySQL AB, recomienda que la mejor forma de instalar MySQL en Linux, debe ser por medio de RPM (Red Hat Pacage Manager), la página Web ofrece barios RPM, por si solo se quiera instalar el servidor o el cliente. En este caso como se usa

Debian GNU/Linux, se elijen los paquetes (binarios) mysql-client, mysql-server y mysql-common. En los siguientes pasos se instalan los archivos *.tar.gz* y *.tar*.

Paso 1: Se debe contar con los archivos.

```
mysql -versión.tar.gz
mysql -versión.tar
```

Paso 2: Se continua con desempaquetar los archivos.

```
gunzip mysql -versión.tar.gz
tar xvf mysql -versión.tar
```

Paso 3: Crear un enlace que haga referencia al siguiente directorio, para que los archivos de MySQL queden vinculados desde el directorio /user/local/mysql.

```
ln -s /usr/local/mysql-versión mysql
```

Paso 4: Crear un usuario nuevo y grupo MySQL, para que se ejecute y administre MySQL como usuario mysql en lugar de root.

```
groupadd mysql
useradd -g mysql mysql
```

Paso 5: Acceder al directorio mysql.

```
cd mysql
```

Paso 6: Configuración de archivos del código fuente de Mysql.

```
./configure -prefix=/usr/local/mysql
```

Paso 7: Compilación del archivo:

```
make
```

Paso 8: Instalación de archivos.

```
make install
```

Paso 9: Instalación de la base de datos predeterminada.

```
scripts/mysql_install_db
```

Paso 10: Haciendo cambios de permisos de los nuevos archivos.

```
chown -R root /usr/local/mysql/.
chown -R mysql /usr/local/mysql/data.
chown -R mysql /usr/local/mysql/.
```

Paso 11: Este paso es opcional, pero también es muy importante, debido a que si se requiere de que el servidor se ejecute de manera automática, solo será necesario de poner el *script* que ejecuta el demonio adecuadamente la estructura del directorio del comando *init* (/etc/rc.d):

```
cp support-files/mysql.server.sh /etc/rc.d/init.d/mysqld
chkconfig --add mysqld
/sbin/chkconfig --list mysqld
```

Nota: Si se usa un binario precompilado se omiten los pasos (6, 7, 8). **[M]**

5.6. Comprobación de la Instalación

Para continuar con el proceso de MySQL, se debe de verificar si el presente gestor de base de datos, fue instalado correctamente. En los siguientes temas se corrobora este proceso.

5.6.1. Ejecución de MySQL

La ejecución de MySQL (*daemon mysql*) es el primer paso para poder saber si fue instalado satisfactoriamente. En seguida se redactan los pasos para corroborar la instalación:

Paso 1: Iniciar una interfaz de línea de comandos.

Paso 2: Ir al directorio de MySQL.

```
cd /usr/local/mysql
```

Paso 3: Iniciar *safe_mysqld* para que se ejecute en segundo plano o se accede al directorio bin y se teclea safe_mysqld.

```
bin/safe_mysqld
o
safe_mysqld
```

La primera vez que se inicia *safe_mysqld,* detecta que mysql no esta iniciado y procede a iniciarlo.

Si todo funciona correctamente, se mostrara el siguiente mensaje. **[M]**

```
Starting mysqld daemon with databases from /usr/local/mysql/data
```

5.6.2. Detención de MySQL

Después de haber iniciado MySQL (*daemon mysql*) con éxito, su detención es fácil, ya que detener a MySQL no se realizara frecuentemente, solo en situaciones de actualización y mantenimiento. A continuación se redactan los pasos necesarios para realizar esta operación:

Paso 1: Iniciar una interfaz de comandos.

Paso 2: Cambiar al directorio *mysql/bin*.

```
cd /usr/local/mysql/bin
```

Paso 3: Teclear mysqladmin, si el sistema tiene una contraseña de root, será necesario agregar el nombre de usuario y contraseña. **[M]**

```
mysqladmin -u root -p shutdown
```

5.6.3. Estado real de MySQL

Paso 1: Iniciar una interfaz de línea de comandos.

Paso 2: Cambiar al directorio *mysql/bin*.

```
cd /usr/local/mysql/bin
```

Paso 3: Escribir.

```
mysqladmin -u root -p status
```

Después de dar enter, se introduce la contraseña de root. Si se tiene éxito se mostrara el tiempo que ha estado conectado el *daemon* mysql.

Nota:
Para ver la versión de MySQL.

```
mysqladmin -u root -p versión
```

Para ver si MySQL se esta ejecutando sin tener que ve los datos estadísticos. **[M]**

```
mysqladmin -u root -p ping
```

5.7. Uso del cliente mysql (monitor mysql)

El monitor mysql permite conectarse a *mysqld* que se encuentre en ejecución en la misma computadora o en una computadora diferente. El cliente MySQL permite distintos argumentos, como el nombre del usuario, la contraseña y el nombre del host, estableciéndose de la siguiente manera:

```
mysql -u nombre_usuario -p -h nombre_host
```

La opción *-p* hace que mysql pida la contraseña. En el monitor mysql todas las instrucciones deben finalizar con punto y coma (;).

Pasos para usar el cliente mysql.

Paso 1: Iniciar una interfaz de línea de comandos.

Paso 2: Cambiar al directorio mysql/bin.

```
cd  /usr/local/mysql/bin
```

Paso 3: Introducir.

```
mysql -u nombre_usuario -p
o
mysql -u nombre_usuario -p -h nombre_host
```

Paso 4: Selección de base de datos que se usara, en este caso se usara la base de datos (*test*) instalada por defecto durante la instalación de MySQL.

```
USE test;
```

En caso de que se conozca la base de datos con la que se desee trabajar, se teclea.

```
mysql -u nombre_usuario -p nombre_base_de_datos
```

Paso 5: Por ultimo se sale de mysql con: **[M]**

```
exit
o
quit
```

5.8. Configuración

Las configuraciones se pueden hacer durante la instalación de MySQL, pero en este capitulo se redactaran algunas de las opciones de configuración y los métodos para realizarlas.

Opciones del comando mysqld: Cuando se inicia el servidor existe la posibilidad de seleccionar opciones de programas mediante los métodos (línea de comandos, en un fichero opcional y en variables de entorno). Cuando se elije alguna de las opciones, es recomendable asegurase que el servidor use la misma opción cada vez que se inicie, puesto que *mysqld* lee opciones de los grupos *mysqld* y *server*.

Para ver las opciones de comandos que maneja *mysqld* solo será necesario, ejecutar *mysqld –help* o *mysqld –verbose*. Cabe mencionar que no se definen las opciones de comandos con las que cuenta mysqld, puesto que la lista es muy amplia.

Modo SQL del servidor: Facilita el uso de MySQL, ya que los modos se pueden aplicar de diferentes formas a varios clientes. Otra función principal de los modos es la definición de la sintaxis SQL que debe de soportar MySQL y que clase de chequeos de validación de datos debe realizar.

Para definir el modo SQL por defecto, solo se requiere de iniciar *mysqld* con la opción.

```
--sql-mode="modes".
```

Variables de sistema del servidor: Al iniciar el servidor *mysqld* conserva dos clases de variables, las primeras son las variables globales estas afectan las operaciones globales del servidor y las variables de sesión afectan las operaciones para conexiones individuales de clientes. Este tipo de variables pueden ser modificadas o reconfiguradas al iniciar el servidor usando la opción líneas de comando, o el ficheros de opciones. Para reconfigurar las variables, se ejecuta el comando *SET*:

- ✓ *SET GLOBAL var_name*: Cambia el nombre de una variable global.
- ✓ *SET SESSION var_name*: Cambia el nombre de una variable global.

Variables de sistema dinámicas: Estas variables pueden adquirir su valor usando *SELECT* y se pueden modificar cuando el servidor esta en ejecución, con *SET GLOBAL o SET SESSION*.

Variables de estado del servidor: Este tipo de variables están encargadas de proporcionar información sobre sus operaciones. La gran mayoría de las variables de estado son inicializadas a 0 por la sentencia *FLUSH STATUS*. **[60]**

5.9. Conexión al servidor

Para poderse conectar a un servidor que se ejecuta en una computadora separada, se debe de contar con un nombre y contraseña de usuario. El siguiente proceso

muestra como se realiza la conexión de un usuario al servidor.

```
mysql -h host -u user -p
Enter password: ********
```

Donde *host* representa el nombre de la computadora donde se esta ejecutando el servidor y *user* es el nombre del usuario.

Después de haber accedido de manera correcta al servidor, se mostrara el siguiente mensaje:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 25338 to server version: 5.0.9-beta-standard
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

El mensaje *mysql>* indica que se puede comenzar a trabajar con MySQL. Para cerrar la desconexión de MySQL, solo se tecleara QUIT o \q, como se especifica a continuación. **[60]**

```
mysql> QUIT
Bye
```

5.10. Seguridad de MySQL

En el presente tema se describen algunos puntos importantes con respecto a la seguridad de MySQL contra ataques o errores de uso, entre los cuales se encuentran; la intercepción pasiva de paquetes, alteración, reproducción de comandos (playback), y denegación de servicio. Por otro lado, para tener una mejor seguridad también se deben de tomar en cuenta los siguientes puntos al ejecutar MySQL:

- ✓ No proporcionar a nadie la cuenta de root.
- ✓ Uso de sentencias *GRANT* y *REVOKE* para controlar el acceso en MySQL.
- ✓ Asignar solo privilegios necesarios.
- ✓ Comprobar que no se pueda conectar al servidor con el comando *mysql -u root*.
- ✓ Comprobar con la sentencia *SHOW GRANTS* que tipos de accesos tienen los usuarios.
- ✓ Usar la sentencia REVOKE para bloquear privilegios innecesarios.
- ✓ Las contraseñas guardadas en la base de datos deben estar cifradas.
- ✓ No se deben de crear contraseñas que estén en un diccionario de datos.
- ✓ Hacer uso de firewall.
- ✓ Escaneo y bloqueo de puerto, principalmente el puerto 3306 usado por MySQL, se puede usar *telnet server_host 3306*.
- ✓ Revisar la información enviada por los usuarios.

- ✓ Proteger valores de tipo cadena y numéricos. Los valores numéricos se le pueden colocar comillas simples alrededor de las constantes numéricas.
- ✓ Comprobar el tamaño de los datos antes de enviarlos.
- ✓ Al enviar datos por Internet, estos deben estar cifrados. **[60]**

5.10.1. Contra ataques

Para hacer una conexión segura entre cliente y servidor será necesario hacer uso de SSH para cifrar la comunicación. Pero si se requiere de una seguridad sencilla, se puede lograr usando el protocolo comprimido, para que la información sea difícil de descifrar.

SSH realiza una conexión TCP/IP cifrada entre servidor y el cliente MySQL. Para lograr este tipo de conexión se debe de tomar en cuenta los siguientes consejos:

- ✓ Usar contraseñas para los usuarios de MySQL.
- ✓ Iniciar el servidor MySQL como usuario sin privilegios y no como usuario *root*.
- ✓ Verificar que el usuario que ejecute *mysqld* tenga privilegios de escritura y lectura.
- ✓ Dar privilegios *PROCESS* o S*UPER* y *FILE,* solo a usuarios administrativos.
- ✓ Si no es seguro el DNS, se pueden usar números IP en lugar de nombres de tablas de permisos.
- ✓ En caso de que sea necesario restringir el número de conexiones a una cuenta, se puede hacer uso de las variables *max_user_connections* o la sentencia *GRANT*. **[60]**

5.10.2. Tabla de opciones de mysqld que afecta la seguridad

En la tabla 5.2 se mencionan las opciones que afectan la seguridad de mysqld.

OPCIÓN	CONSECUENCIA
--allow-suspicious-udfs	Controla las funciones definidas por el usuario que tienen un símbolo *xxx* que puedan ser cargadas en la función principal.
--local-infile[={0|1}]	Si el servidor inicia con *--local-infile=0*, los clientes no pueden usar *LOCAL* en comandos *LOAD DATA.*
--old-passwords	Obliga al servidor a generar las contraseñas para las nuevas contraseñas.
--safe-user-create	Si esta opción se encuentra activada, los usuarios no podrán crear nuevos usuarios con el comando *GRANT*, amenos que tengan privilegios de *INSERT* para la tabla *mysql.user.*
--secure-auth	Desactiva la autentificación de cuentas que usen viejas contraseñas.
--skip-grant-tables	Obliga al servidor que no use el sistema de privilegios.
--skip-name-resolve	Los nombres de equipo no se resuelven.
--skip-networking	No admite conexiones TCP/IP.
--skip-show-database	Este comando permite a usuarios con privilegio *SHOW DATABASES*.

Tabla 5.2: Opciones que afectan la seguridad de mysqld. **[60]**

5.11. Sentencias SQL

Las sentencias SQL, son un método de acceso a las bases de datos, compuestas por sintaxis SQL y variables.

5.11.1. Sentencias de manipulación de datos

SELECT: Esta sentencia tiene dos funciones; la primera función es usada para recibir registros seleccionados desde una o más tablas y la segunda función es para recuperar registros computarizados sin referencia a alguna tabla.

HANDLER: Este tipo de comando es de bajo nivel ya que no da consistencia al abrir las tablas dejándolas con posibilidad de modificación. Cabe mencionar que este comando es muy similar al comando SELECT. Aun así existen varias ventajas para usar este comando, como se describen a continuación:

- ✓ Rapidez.
- ✓ No existe sobrecarga del chequeo de consultas.
- ✓ Facilidad de portar aplicaciones con interfaz ISAM a MySQL.
- ✓ Facilita las consultas a una base de datos, que no se pueda hacer con SELECT.
- ✓ Facilita el acceso directo a interfaces del motor de la tabla.

***INSERT*:** Este comando esta orientado a la inserción de registros en una tabla seleccionada. Otra característica del comando INSERT, es que hace uso de los comandos VALUES, SET y SELECT. Estos comandos se encargan de insertar registros de otras tablas.

INSERT DELAYED: En este tema se hablara acerca de DELAYED, ya que para el comando INSERT es una extensión de MySQL, para clientes que quieren conectarse rápidamente. Con respecto a los benéficos que puede ofrecer esta opción, es la inserción de varios clientes que se procesan juntos y se escriben en un bloque. Pero no todo es beneficio ya que DELAYED tiene algunas desventajas de uso, como:

- ✓ Trabaja sólo con tablas MyISAM y MEMORY en MySQL versión 5.0.
- ✓ Se usa sólo para comandos INSERT que definan una lista de valores en MySQL versión 5.0.
- ✓ El servidor no toma en cuenta a DELAYED para comandos INSERT DELAYED ...ON DUPLICATE UPDATE.

DELETE: Comando que usa la cláusula WHERE para borrar los registros que cumplan con esta, ya que si es usado el comando sin la cláusula, se eliminaran todos los registros.

UPDATE: El presente comando esta encargado de la actualización de las columnas en registros de las tablas. Este comando hace uso de las cláusulas SET, WHERE, ORDER BY y LIMIT. Don de SET se usa para indicar que columna será actualizada y los valores nuevos, WHERE define los registros que se van actualizar, ORDER BY

ordena los registros de acuerdo al orden especificado y LIMIT es el limite de registros a actualizar.

TRUNCATE: El comando TRUNCATE TABLE, es un comando derivado de la extensión de Oracle SQL. Actualmente es adoptado por MySQL. La función principal de este comando es vaciar completamente una tabla, por lo tanto se dice que es similar al comando DELETE.

LOAD DATA INFILE: Para poder usar este comando se debe de contar con permisos de FILE y tiene la función de leer registros desde un fichero de texto a una tabla a gran velocidad.

DO: *E*s el comando que se encarga de ejecutar la expresión donde sea usado, pero este no devuelve valor alguno. Este comando se usa principalmente con funciones que tiene efectos colaterales. **[60]**

5.11.2. Sentencias de definición de datos

CREATE DATABASE: Sentencia usada para la creación de bases de datos, siempre y cuando se tenga permiso de CREATE en la base de datos.

DROP DATABASE: Para usar esta sentencia será necesario contar con permisos DROP, ya que se usa para eliminar tablas y base de datos, en caso de que la base de datos este conectada también la conexión se borrara. Después de realizar este proceso se devolverá el número de tablas borradas.

ALTER DATABASE: El uso de esta sentencia solicita el premiso ALTER para poder modificar las características (guardadas en el fichero del directorio de la base de datos, como db.opt) globales de una base de datos.

CREATE TABLE: Sentencia que permite crear tablas, siempre y cuando se tengan permisos de tipo CREATE para tablas.

ALTER TABLE: La presente sentencia permite modificar una tabla, como; agregar o borrar columnas, crear o destruir índices, cambiar o renombrar el tipo de columnas, etc. El uso de ALTER TABLE, pide ALTER, INSERT, y permisos CREATE para las tablas.

RENAME TABLE: Permite renombrar una o varias tablas, mientras se cuente con permisos ALTER y DROP en la tabla original, y permisos CREATE e INSERT en la nueva tabla.

DROP TABLE: Facilita el borrado de una tabla con los datos que contiene, para poder borrar una tabla se necesita el permiso DROP de la tabla.

DROP INDEX: Sentencia para borrar índices de una tabla.

CREATE INDEX: Directiva que facilita crear índices en una tabla nueva creada con CREATE INDEX, además se pueden crear índices en tablas ya existentes. **[60]**

5.12. API's soportadas por MySQL

En la mayoría de las ocasiones se habla de las API's, para el uso de bases de datos y diferentes aplicaciones, pero para muchos usuarios es desconocido el significado de la abreviación API. Las siglas API tienen el significado en inglés de *Application Programming Interface*. En otras palabras, son los métodos que el desarrollador de cualquier aplicación ofrece a otros desarrolladores para que puedan interactuar con su aplicación. En seguida se mencionan las API's principales que usa MySQL (ver Tabla 5.3). **[60]**

LENGUAJE	NOMBRE DE LA API
C	API contenida en la librería mysqlclient
C++	MySQL++
PHP	mysq mysqli
Python	MySQLdb
Perl	mysqlperl
Tlc	MySQLtcl

Tabla 5.3: API's usadas por MySQL.

5.13. Controladores de MySQL

JDBC (Java Database Connectivity): Es un API que facilita realizar procesos sobre las bases de datos usando el lenguaje de programación Java como interfaz entre la base de datos y el usuario, sin importar que sistema operativo se este usando.

Controlador NET: Actualmente este tipo de controlador es usado por MySQL como controlador entre base de datos y la interfaz para el usuario. Un punto importante que se debe de mencionar, es que este controlador es para los lenguajes .NET que son programados en un cien por ciento en lenguaje de programación C.

ODBC (Open Database Connectivity): Este es un controlador de la familia MyODBC para interfaces de aplicaciones para acceder a los datos en un sistema gestor de base de datos. En otras palabras el controlador MyODBC es un intermediario entre bases de datos y aplicaciones, conformado por una arquitectura de cinco partes (ver Figura 5.6), las cuáles se describen a continuación:

- ✓ *Aplicación:* Permite el acceso a datos contenidos en el servidor.
- ✓ *Controlador administrador:* Librería encargada de la comunicación entre aplicación y controlador.
- ✓ *Conexión ODBC:* Realiza operaciones SQL al servidor MySQL.

- ✓ *ODBC.INI:* Encargado de configurar los archivos guardados entre el controlador *y el servidor.*
- ✓ *Servidor MySQL:* Lugar o espacio donde se aloja el sistema gestor de base de datos o base de datos. **[60]**

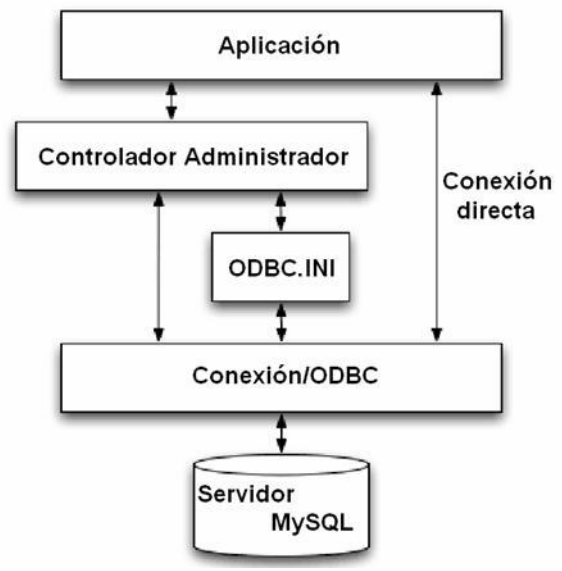


Figura 5.6: Arquitectura de MyODBC. **[60]**

5.14. Creación y uso de base de datos

En este tema se presentara la forma en que se selecciona o se crea una base de datos, así como la creación, eliminación y uso de una tabla entre otras operaciones, siempre y cuando se tengan los permisos adecuados.

1.- Para empezar a usar una base de datos, primero se debe verificar si existen en *mysql,* haciendo uso del comando SHOW DATABASES. Cabe mencionar que las sentencias se finalizan con punto y coma, pero USE y QUIT, no usan punto y coma.

2.- En caso de que no exista la base de datos, se creara con el comando CREATE DATABASE más el nombre de la base de datos "CREATE DATABASE nom_bd". En los sistemas Unix los nombres de bases de datos y tablas son sensibles a minúsculas y mayúsculas.

3.- Con el comando USE se puede seleccionar una base de datos "USE nom_bd". Otra forma de seleccionar una base de datos es inicializarla desde la línea de comandos al ejecutar *mysql*.

Ejemplo:

```
mysql -h host -u user -p nom_bd
Enter password: ********
```

Si se accede correctamente se mostrara lo siguiente.

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 25338 to server version: 5.0.9-beta-standard
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

Nota: Para salir del mysql solo se teclea QUIT o \q.

4.- Después de seleccionar una base de datos, se emplea *SHOW TABLES* para revisar si existen tablas.

5.- En caso de no encontrar tablas en una base de datos, se pueden crear con el comando CREATE TABLE "CREATE TABLE nom_tabla (variables o informacion)".

6.- Uno de los comandos utilizados para verificar si una tabla fue creada es SHOW TABLES. Cabe decir que también se puede usar DESCRIBE para realizar la misma operación "DESCRIBE nom_tabla", con la ventaja de que muestra el contenido de la tabla.

7.- Para iniciar con un proceso de inserción de datos en una tabla se puede utilizar la sentencia LOAD DATA e INSERT. Cuando se emplean estos comandos, es necesario crear un archivo ".txt", teniendo un registro por línea, cada valor será separado por un carácter de tabulación y con el orden que tiene las columnas creadas con la sentencia CREATE TABLE. En caso de usar valores nulos se usa \N.

Ejemplos:

```
LOAD DATA LOCAL INFILE '/path/hola.txt' INTO TABLE nom_tabla;
o
INSERT INTO nom_tabla
-> VALUES (information a introducir);
```

8.- Una forma facil y rapida de mostrar el contenido de una tabla, es hace uso de SELECT * FROM nom_tabla.

9.- En MySQL existen diferentes formas de seleccionar un registro, como:

- ✓ Por nombre: SELECT * FROM nom_tabla WHERE nom_columna = 'nom_registro
- ✓ Por columna: SELECT * FROM nom_tabla WHERE nom_columna > 'nom_registro
- ✓ Usando los operadores AND y OR, por separado o combinados. Aquí el operador AND tiene una prioridad mayor.

10.- Otro punto importante al usar tablas en MySQL es la facilidad de seleccionar columnas completas con SELECT. . . . . FROM, "SELECT nom_columna, nom_columna FROM nom_tabla".

11.- El ordenamiento de registros se puede lograr mediante el uso de SELECT con la cláusula ORDER BY, "SELECT nom_columna, nom_registro FROM nom_tabla ORDER BY nom_registro". Cabe mencionar que este tipo de ordenamiento también se puede hacer de manera descendente "SELECT nom_columna, nom_registro FROM nom_tabla ORDER BY nom_registro DESC".

12.- Por ultimo se escribe la forma de cómo se puede hace un conteo de registros contenidos en una tabla, haciendo uso de COUNT(*), para contar el numero de filas "SELECT COUNT(*) FROM nom_tabla". **[60]**

5.14.1. Ejemplos del uso de otros comandos en MySQL

Mostrando el número de versión de mysql y fecha actual.

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| VERSION()    | CURRENT_DATE |
+--------------+--------------+
| 5.0.7-beta-Max | 2005-07-11 |
+--------------+--------------+
1 row in set (0.01 sec)
```

Usando *mysql* como calculadora.

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-----------------+---------+
| SIN(PI()/4)     | (4+1)*5 |
+-----------------+---------+
| 0.70710678118655 | 25     |
+-----------------+---------+
1 row in set (0.02 sec)
```

Nota: En una línea de comandos se puede especificar más de una sentencia separada por punto y coma, ejemplo:

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| VERSION()    |
+--------------+
| 5.0.7-beta-Max |
+--------------+
1 row in set (0.00 sec)

+---------------------+
| NOW()               |
+---------------------+
| 2005-07-11 17:59:36 |
+---------------------+
1 row in set (0.00 sec)
```

Para poder cancelar un comando se usa \c: [60]

```
mysql> SELECT
-> USER()
-> \c
mysql>
```

5.15. Permisos

Cuando un usuario quiere acceder al servidor, este comprueba si el usuario tiene permisos para conectarse pidiendo nombre y contraseña. Otro tipo de permisos que se les asigna a los usuarios, son los de manipulación de la información en la base de datos. En la Tabla 5.4, se muestran los diferentes tipos de permisos que se les pueden asignar a los usuarios. **[60]**

Permisos	Usuario	Permite
CREATE		Crea bases de datos, tablas e índices.
DROP		Borra bases de datos y tablas.
GRANT OPTION		Da a los usuarios lo permisos que uno mismo posee sobre bases de datos, tablas y procedimientos almacenados.
REFERENCES		Comentario de bases de datos y tablas.
ALTER		Modifica tablas.
DELETE		Borra tablas.
INDEX		Permite crear o eliminar índices.
INSERT		Inserta tablas.
SELECT		Selección de tablas.
UPDATE		Actualiza tablas.
CREATE VIEW		Creación de vistas.
SHOW VIEW		Muestra vistas.
ALTER ROUTINE		Modifica o elimina procedimientos almacenados.
CREATE ROUTINE		Crea procedimientos almacenados.
EXECUTE		Procedimientos almacenados.
FILE		Da permiso para leer y escribir archivos en la máquina del servidor.
CREATE TEMPORARY TABLES	Administrador	Deja utilizar la palabra *TEMPORARY* en sentencias *CREATE TABLE*.
LOCK TABLES	Administrador	Facilita el uso de la sentencia *LOCK TABLES* para bloquear tablas con permisos *SELECT*.
CREATE USER	Administrador	Deja crear nuevos usuarios.
PROCESS	Administrador	Deja ver que subprocesos han sido iniciados por otros usuarios.
RELOAD	Administrador	Le dice al servidor que debe releer las tablas *grant* a memoria.
REPLICATION CLIENT	Administrador	Deja usar las sentencias *SHOW MASTER STATUS* y *SHOW SLAVE STATUS*.
REPLICATION SLAVE	Administrador	Permisos dados a los servidores esclavos por un servidor maestro.
SHOW DATABASES	Administrador	Facilita ver las bases de datos alojadas en el servidor.
SHUTDOWN	Administrador	Apaga el servidor.
SUPER	Administrador	Deja cerrar la conexión a otros usuarios.

Tabla 5.4: Permisos disponibles para los usuarios.

5.16. Operaciones con cuentas de usuarios en MySQL

Una vez que se tenga en ejecución MySQL, llega el momento de crear nuevos

usuarios para mejorara la seguridad de la base de datos. La creación de nuevos usuarios, como se menciona mas adelante, son necesarios para que no se tenga que recurrir al usuario root.

1.- Para lograr una conexión a mysql con permisos de administrador (Root), se logra mediante lo siguiente:

```
mysql ~user=root mysql
```

En caso de que se le haya proporcionado una contraseña a la cuenta de administrador, se tendrá que iniciar de la siguiente manera:

```
mysql -h host -u user -p
Enter password: ********
```

2.- Las cuentas de usuarios se pueden crear de diferentes formas con el comando GRANT, como:

- ✓ Cuenta que accede a una base de datos, desde una computadora local.
- ✓ Cuenta que accede a una base de datos, desde la computadora con nombre (whitehouse.gov).
- ✓ Cuenta con acceso a una base de datos, desde una computadora con dominio (server.domain).
- ✓ Cuenta con contraseña y permisos de súper usuario, con la facilidad de acceder desde una computadora local.
- ✓ Cuenta con contraseña y permisos de súper usuario, con la facilidad de acceder desde cualquier computadora.
- ✓ Cuenta sin contraseña y con permisos *RELOAD* y *PROCESS*, usada desde una computadora local.
- ✓ Cuenta sin contraseña y permisos, usada desde una computadora local.

3.- Para agregar contraseña a una cuenta de usuario, se puede hacer desde:

- ✓ La línea de comando, con el comando *mysqladmin*.
- ✓ Usando el comando *SET PASSWORD*.
- ✓ Uso del comando *GRANT USAGE,* globalmente (ON *.*).
- ✓ Asignación de contraseña al crear una cuenta.
- ✓ Agregando una contraseña a una cuenta ya existente, con el comando *UPDATE*.

4.- En caso de que se quiere eliminar una cuenta de usuario se usa la sentencia *DROP USER:* **[60]**

5.17. Prevención de fallas y recuperaciones

En el presente tema se explica brevemente el tema principiara de la creación de una copia de base de datos, así como el uso de sentencias para el mantenimiento de las

tablas alojadas en la base de datos, además se hará mención del uso de *myisamchk* para revisar y repara tablas.

5.17.1. Copia de seguridad de la base de datos

En esta sección se explicara la forma básica de hacer una copia de base de datos segura y consistente, siempre y cuando las tablas contenidas en dicha base de datos sean de tipo *MyISAM*. El hacer una copia de base de datos con MySQL es sencillo debido a que las tablas son guardadas como archivos, en este caso se explicara como se hace la copia de seguridad con los programas *mysqldump* y *mysqlhotcopy*.

***mysqldump*:** Programa usado para volcar y hacer copia de seguridad de base de datos o transferir datos a otro servidor SQL. En este caso se utiliza para crear copias de seguridad de base de datos:

```
mysqldump --tab=/path/to/some/dir --opt nom_bd
```

***mysqlhotcopy*:** Script de Perl, para hacer copias de seguridad rápidas de base de datos y tablas. Este script solo funciona el Unix y se ejecutar en la computadora donde está el directorio de base de datos. **[60]**

```
mysqlhotcopy nom_bd /path/to/some/dir
```

Nota:
--tab: Produce ficheros con datos separados por tabuladores.
--opt: Crea de manera rápidamente un fichero de volcado y se puede cargara en el servidor MySQL.
/path/to/some/dir: Ruta al Nuevo directorio.

5.17.2. Mantenimiento de tablas y recuperación de fallas

En el siguiente texto se redactan las diferentes opciones para comprobara y repara tablas *MyISAM* (archivos *.MYI* y *.MYD*). En primer instancia se hace mención de las sentencias SQL para repara las tablas, ya que en varias ocasiones resultan ser mas rápida y menos complejas de usar que *myisamchk*. La utilidad *myisamchk* (repara y optimizar), es muy segura y también se usa para conseguir información sobre las tablas de una base de datos. **[60]**

5.17.2.1. Sentencias para el mantenimiento de tablas

ANALYZE TABLE: Este comando funciona para tablas MyISAM, BDB y InnoDB. Permite analizar y almacenar la distribución de claves para una tabla.

CHECKSUM TABLE: Saca un checksum para saber si existe o no una tabla, si no existe devolverá un valor NULL.

CHECK TABLE: Comando encargado de verificar tablas para errores, funcionando para tablas MyISAM y InnoDB, donde la estadística de claves se actualiza en MyISAM.

OPTIMIZE TABLE: El presente comando funciona en tablas MyISAM, BDB y InnoDB. En tablas MyISAM puede emplearse para ordenar páginas, índices, repara tablas y actualiza estadísticas.

REPAIR TABLE: Este comando casi no se usa, solo si ocurre un desastre será usado para reparar todos los datos de una tabla dañada y funciona sólo en tablas MyISAM.

RESTORE TABLE: Funciona solo para tablas MyISAM, y se emplea para restaurar tablas de una copia de seguridad que se creó con BACKUP TABLE. Cabe mencionar que una restauración tarda más tiempo que la copia de seguridad. **[60]**

5.17.2.2. Aplicando myisamchk

El uso de *myisamchk,* requiere de diferentes opciones (ver Tabla 5.5) para que realice lo que se desee hacer con las tablas, en caso de que no se usen las opciones, *myisamchk* por defecto solo comprobara la tabla.

Sintaxis:

```
myisamchk [opciones] tbl_name
```

5.17.2.2.1. Comandos para comprobar tablas MyISAM

A continuación se hace mención de los comandos usados para comprobar tablas de tipo *MyISAM*.

myisamchk nom_tabla: No detecta errores que involucre al archivo de datos.

myisamchk -m nom_tabla: Revisa todos los índices y posteriormente lee todos los registros para verificar que las claves de los registros y el árbol de índices sean iguales.

myisamchk -e nom_tabla: Hace una comprobación completa y exhaustiva de todos los datos, buscando errores.

myisamchk -e -i nom_tabla: Hace una comprobación completa y exhaustiva de todos los datos, buscando errores e imprime información acerca de ellos. **[60]**

5.17.2.2.2. Reparar tablas

Con lo siguiente se puede hacer la comprobación de tablas:

```
myisamchk *.MYI o myisamchk -e *.MYI
```

OPCIÓN	NOTA
Generales	
--help, -?	Presenta un mensaje de ayuda y finaliza.
--debug=*debug_options*, -# *debug_options*	Escribe un registro de depuración. *debug_options por lo regular es:* 'd:t:o,*file_name*'.
--silent, -s	Modo silencioso y se escribe cuando ocurre algún error. Se aplica –s y –ss, para que *myisamchk* sea muy silencioso.
--verbose, -v	Modo explícito. Imprime información y si se usa barias veces (-vv, -vvv) aparece más información.
--version, -V	Enseña información sobre la versión.
--wait, -w	Finaliza un error de una tabla y si se encuentra bloqueada, espera hasta que se desbloquee.
Comprobación de tablas	
--check, -c	Checa si no hay errores en la tabla.
--check-only-changed, -C	Revisa las tablas que han cambiado desde la última comprobación.
--extend-check, -e	Comprueba las tablas lentamente y se usa en casos extremos.
--fast, -F	Comprueba las tablas que no se cierran correctamente.
--force, -f	Efectúa un proceso de reparación automático si *myisamchk* detecta un error en la tabla.
--information, -i	Imprime estadísticas con información de la tabla que se comprueba.
--medium-check, -m	Funciona mas rápido que *--extend-check* y detecta 99.99% de los errores en las tablas.
--read-only, -T	No señala la tabla como comprobada.
--update-state, -U	Guarda información en el archivo .MYI para mostrar que la tabla fue comprobada.
Reparación de tablas.	
--backup, -B	Hace una copia de seguridad del *archivo .MYD* con el formato file_name-time.BAK.
--character-sets-dir=*path*	Directorio donde los juegos de caracteres están instalados.
--correct-checksum	Corrige la información de *checksum* de la tabla.
--data-file-length=#, -D #	Longitud máxima del archivo de datos.
--extend-check, -e	Efectúa una reparación que intenta recuperar todos los registros posibles del archivo de datos.
--force, -f	Sobrescribe los archivos temporales en lugar de interrumpir la reparación de ellos.
--keys-used=#, -k #	Muestra qué índices se deben de actualizar.
--parallel-recover, -p, –r,-n	Crea las claves en paralelo, usando hilos de ejecución diferentes.
--quick, -q	Realiza una reparación rápida y no modifica el archivo de datos.
--recover, -r,	Hace una reparación rápida, únicamente que no repara claves únicas.
--safe-recover, -o	Realiza una reparación usando un método de recuperación antiguo que lee los registros en orden y actualiza los árboles de índices basándose en los registros encontrados.
--set-collation=*name*	Cambia la colación utilizada para ordenar los índices de las tablas.
--sort-recover, -n	Obliga a *myisamchk* usar orden para establecer las claves aunque los archivos temporales sean demasiado grandes.
--tmpdir=*path*, -t *path*	Dirección del directorio donde se deben guardar los archivos temporales.
--unpack, -u	Descomprime una tabla que fue comprimida con *myisampack*.

Tabla 5.5: Opciones de myisamchk.

Pasos que se debe de seguir para la reparación de tablas:

1.- Hacer una copia de seguridad del archivo de datos.

2.- Ejecutar -r (significa modo de recuperación).

```
myisamchk -r nom_tabla
o
myisamchk --safe-recover nom_tabla
```

Reparaciones de una tabla complicada: Si el primer bloque de 16KB en el archivo de indexación está destruido, si la información que contiene es errónea o si el fichero de indexación no se localiza.

1.- Guardar el archivo de datos en un lugar seguro.

2.- Crear un archivo de datos e indexación usando el archivo descriptor de la tabla:

```
mysql nom_bd
mysql> SET AUTOCOMMIT=1;
mysql> TRUNCATE TABLE nom_tabla;
mysql> quit
```

3.- Copiar el archivo viejo de datos sobre el fichero nuevo y guardar una copia. Regresar al paso 2 o teclear REPAIR TABLE nom_tabla USE_FRM, que realiza el proceso completo automáticamente. **[60]**

Capítulo

6

Preprocesador de hipertexto (PHP)

PHP (Hipertext Preprocessor) es un lenguaje de programación nuevo, usado como herramienta para el desarrollo de páginas Web. PHP facilita el diseño de páginas dinámicas de servidor, es decir, genera páginas bajo la petición de responder las demandas del cliente y no permite la automatización de gran cantidad de tareas. A pesar de que en la actualidad existen muchos lenguajes para el diseño de paginas Web, PHP se a convertido en uno de los lenguajes de lado del servidor.

Por tal motivo, en este capítulo se muestran algunas ventajas de PHP, ya que actualmente da grandes servicios para la creación de interfaces de bases de datos, así como a las páginas Web. Cabe mencionar que entre otros temas que se indican están los inconvenientes del uso PHP, tipos de datos, estructuras de control que usa, entre otros.

6.1. Un poco de historia de PHP

PHP fue desarrollado por Rasmus Lerdorf. Al principio PHP solo era un conjunto de *scripts* escrito en Perl que permitía el control de los accesos a páginas personales. A este conjunto de scripts les dio el nombre de Personal Home Page Tools. Conforme iba pasando el tiempo, Rasmus fue completando las funciones básicas de su herramienta con programas hechos en C. Fue hasta 1995 cuando se decide y libera el código fuente escrito en C para que todos lo pudieran usar y ayudar a mejorarlo. De esta manera surge PHP/FI. A finales de 1997 se libera PHP/FI 2.0, pasando a ser un proyecto de muchas personas. Pronto se convirtió en el 1% de los dominios que existían en Internet.

En junio de 1998 se liberó oficialmente PHP 3.0, como sucesor de PHP/FI 2.0, el cuál fue completamente reescrito por Andi Gutmans y Zeev Zuraski. Algunas de las principales características de PHP 3.0 que ayudo a que muchos desarrolladores se unieran y enviaran nuevos módulos de extensión para el mejoramiento de éste, fue su extensibilidad, daba a los usuarios finales una estructura sólida para muchas bases de datos, protocolos y API´s. Todo esto llevo a que PHP 3.0 ocupara el 10% de los servidores Web en Internet. En mayo del 2000 se libera la versión PHP 4.0, el cuál había sido reestructurado su núcleo, dando lugar al motor Zend (acrónimo de los apellidos Zeev y Andi), PHP 4.0 fue basado en este motor y acoplado con nuevas características adicionales.

Actualmente se trabaja en la modificación y mejora del motor Zend, para integrar las nuevas características a PHP5.

PHP (ver Figura 6.1), es un procesador de hipertexto (en ingles Hypertext Preprocessor), sin duda es uno de lo lenguajes de código abierto de alto nivel, alojado en código HTML que es ejecutado en el servidor. En otras palabras se dice que PHP es fácil de usar y a la vez brinda grandes ventajas a los desarrolladores de aplicaciones. **[L]**

Figura 6.1: Logotipo de PHP. **[70]**

6.2. Ventajas de usar PHP

Al hablar del lenguaje PHP, también se debe de mencionar las grandes ventajas que tiene sobre otros lenguaje de programación ya que PHP actualmente se a convertido en una de las mejores opciones para muchas necesidades. Por esta razón se mencionan estas ventajas por las cuáles los usuarios eligen PHP, para dar un interfaz a sus bases de datos:

- ✓ Es rápido cuando es usado como módulo de Apache
- ✓ Gratuito.

- ✓ Fácil de usar: Es fácil de entender y programar.
- ✓ Versátil: PHP funciona en 25 plataformas. Esto quiere decir que, se puede ejecutar en la mayoría de los sistemas operativos.
- ✓ Soporte amplio: Facilidad de que el usuario pueda subscribirse a una de las varias listas de discusión electrónicas que se encuentran en la página principal de PHP.
- ✓ Seguro: Cuando sus script son programados adecuadamente.
- ✓ Se puede personalizar: La licencia que tiene permite a los programadores modificar el software.
- ✓ PHP es completamente expandible.
- ✓ Interacción con varios gestores de bases de datos.
- ✓ PHP posee varios módulos para librerías, también se pueden crear API's cuando sean necesarias.
- ✓ Su sintaxis es similar a la del lenguaje C y a PERL.
- ✓ Soporta clases y herencia.
- ✓ Se puede combinar código PHP y HTML.
- ✓ Soporte de acceso a base de datos. **[N]**
- ✓ Soporte para ODBC y extensiones DBX.
- ✓ Generación de resultados en muchos formatos como XHTML, XML, ficheros de imagen, ficheros PDF y películas Flash.
- ✓ Soporta WDDX para intercambio de datos entre lenguajes de programación en Web.
- ✓ La extensión CORBA puede ser útil para acceder a objetos remotos.
- ✓ Soporte para comunicaciones con otros servicios usando protocolos, como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM y muchos más.
- ✓ Soporte para muchos servidores Web: Apache, Microsoft Internet Information Server, Netscape e iPlanet, caudium, etc. **[L]**

6.3. Inconvenientes del uso de PHP

En caso de que se este desarrollando una aplicación bajo una plataforma UNIX o Linux, es recomendable que se use PHP ya que es mas seguro en estas plataformas. Cabe mencionara que aún así PHP tiene algunas desventajas como las siguientes:

- ✓ Cuando se incrementan las solicitudes a un servidor, PHP será más inestable, ya que el servidor realiza todos los procesos.
- ✓ La legibilidad del código puede ser afectada al combinar sentencias HTML y PHP.
- ✓ El uso de la orientación a objetos aún es deficiente para aplicaciones grandes.
- ✓ No cuenta con un Debugger. **[68]**

6.4. Usos de PHP.

Gracia a que PHP es un lenguaje de propósito general se puede usar para desarrollar script de propósito general. Estos script son archivos de computadora que contiene instrucciones en PHP, las cuáles le dicen a la computadora que hacer.

Debido a que este lenguaje de programación es muy popular para sitios Web, a continuación se describe donde ese usado PHP.

PHP en aplicaciones Web: PHP resulta ser un lenguaje de script de lado del servidor; lo cuál significa que los script son ejecutados en el servidor. Gracias a que los script son ejecutados del lado del servidor, PHP puede crear dinámicamente el código HTML que genera la página Web, esto da origen a que los usuarios puedan ver páginas Web personalizadas, generando que los usuarios que visiten estas páginas solo vean la salida de los script y no el código de estos.

PHP en aplicaciones con bases de datos: La capacidad de este lenguaje de programación para interactuar con las bases de datos es muy buena, gracias a que soporta la mayoría de las bases de datos. PHP se encarga de la conexión y la comunicación de la base de datos, de manera que el usuario no necesita saber los detalles de cómo esta conectada la base de datos ni como intercambia mensajes; eso es por que solo basta con que se le indique a PHP el nombre de la base de datos y su ubicación, conecta la base de datos y da las instrucciones a la base de datos y le devuelve las respuestas.

PHP con el sistema de archivos: PHP funciona con el sistema de archivos, directorios y archivos que están en disco duro o en otra computadora. PHP puede escribir en un archivo de sistema de archivos, puede crearlo y puede leer el contenido de archivos. Otra de las opciones que puede hacer es crear directorios, copiar archivos, cambiarle el nombre, borrarlos, cambiar sus atributos, etc.

PHP para comandos del sistema: PHP tiene la facilidad de interactuar con el sistema operativo para hacer cualquier tarea que este realiza. Sí se ejecuta un comando del sistema operativo, se mostrara la salida del lo que hace dicho comando. La facilidad de ejecutar comandos del sistema incluye la habilidad de ejecutar cualquier programa en el sistema. **[N]**

6.5. Tipos de datos

PHP usa datos similares a los demás lenguajes de programación, en este caso los datos no son sensibles a mayúsculas y minúsculas, como se mencionan a continuación:

Booleanos: Este tipo de dato fue introducido a partir de PHP 4, también es uno de los más sencillos de definir ya que especifica un valor de verdad (True) o falso (False). Cabe mencionar que para usar este tipo de dato es necesario usar un operador para que devuelva un valor bolean.

Sintaxis:

```
<?php
$foo = True; // el valor TRUE es asignado a $foo
?>
```

Enteros (integer): Este es un dato de tipo numérico de -∞,-2, -1, 0, 1, 2, ∞. Estos datos pueden especificarse en las notaciones decimal, hexadecimal y octal, precedidos por el signo – o +.

Sintaxis:

```
<?php
$a = 1234; // número decimal
$a = -123; // número negativo
$a = 0123; // número octal (equivalente al 83 decimal)
$a = 0x1A; // número hexadecimal (equivalente al 26 decimal)
?>
```

Números de punto flotante: Números que también son conocidos como “flotantes”, “dobles” o “números reales”. Estos números tienen un valor desde una máximo de ~1.8e308 con una precisión de aproximadamente 14 dígitos decimales.

Sintaxis:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Cadenas: Las cadenas de caracteres en PHP se pueden especificar entre comillas simples (‘), dobles (") y sintaxis heredoc (<<<).

Matrices: Las matricen en PHP son mapas ordenados. Un mapa ordenado, es un dato que asocia valores (puede ser cualquier valor) con claves (puede ser integer o string), solo así el mapa ordenado podrá ser usado como matriz real, tabla asociativa, diccionario, pila, cola. También con PHP se pueden simular árboles.

Sintaxis:

```
<?php
$matriz = array("foo" => "bar", 12 => true);

echo $matriz["foo"]; // bar
echo $matriz[12];    // 1
?>
```

Recurso: Los recursos son variables especiales y fueron introducidos a partir de PHP 4, estos recursos son creados y usados por funciones especiales.

Objetos: Para poder usar objetos en PHP será necesario iniciarlos con la sentencia *new* (instancia el objeto a una variable).

Ejemplo:

```
<?php
class foo
{
   function hacer_foo()
   {
      echo "Haciendo foo.";
   }
}
$bar = new foo;
$bar->hacer_foo();
?>
```

Valor nulo (NULL): Este valor fue introducido a partir de PHP 4, el cuál representa una variable que no tiene valor. Para que una variable sea considerada con un valor nulo, se le debe de asignar la constante NULL o no asignarle algún valor. **[69]**

Sintaxis:

```
<?php
$var = NULL;
?>
```

6.6. Operadores

Los operadores son un conjunto de símbolos (ver Tabla 6.1a y 6.1b), que permiten limitar las operaciones. Entre los símbolos mas usados se encuentran los operadores suma, resta, multiplicación y división.

NOMBRE	OPERADOR	TIPO DE OPERADOR	NOTA
Comillas invertidas	``	Ejecución	PHP intentará ejecutar el contenido entre las comillas como sí fuera un comando del intérprete de comandos.
Pre-incremento Post-incremento Pre-decremento Post-decremento	++$a $a++ --$a $a--	Incremento/Decremento	Operadores que permiten realizar el incremento y decremento de valores.
Y O O exclusivo (Xor) No	and or xo !	Lógicos	
Unión Igualdad Identidad No-igualdad No-igualdad No-identidad	+ == === != <> !==	Matrices	El operador + adiciona la matriz del lado derecho a aquél al lado izquierdo, al mismo tiempo que cualquier llave duplicada NO es sobrescrita.

Tabla 6.1a: Tipos de operadores que usa PHP. **[69]**

NOMBRE	OPERADOR	TIPO DE OPERADOR	NOTA
Negación Adición Substracción Multiplicación División Módulo	- + - * / %	Aritméticos	El operador ("/") devuelve un valor flotante en todos los casos.
Igual (básico)	= += -= *= /= %= .= &= |= ^= <<= >>=	Asignación	En este caso solo existe el operador igual "=" como un operador básico, este se puede combinas para formar operadores de aritmética binaria, unión de matrices y de cadenas.
Y O Xor No Desplazamiento a la izquierda Desplazamiento a la derecha	& | ^ ~ << >>	Operadores Bit a Bit	Activar o desactivar bits individuales de un entero. Sí los parámetros tanto a la izquierda y a la derecha son cadenas, el operador actuara sobre los valores ASCII de los caracteres.
Concatenación Asignación sobre concatenación	. .=	Cadena	El operador de concatenación; devuelve el resultado de concatenar sus argumentos a lado derecho e izquierdo y el de asignación sobre concatenación; suma el argumento del lado derecho al argumento en el lado izquierdo.
Igual Idéntico Diferente Diferente No idénticos Menor que Mayor que Menor o igual que Mayor o igual que	== === != <> !== < > <= >=	Operadores de Comparación	Este tipo de operadores permite realizar la comparación de valores.
Ternario	?:	Ternario	Operador que evalúa el resultado de una sentencia.
Arroba	@	Control de Errores	Este operador se coloca a principio de cualquier expresión y en caso de que produzca un error, este será ignorado.

Tabla 6.1b: Tipos de operadores que usa PHP. **[69]**

6.7. Variables

Debido a que existe una gran cantidad de variable no se definirán, solo se hará

mención de cómo se deben de escribir. En PHP el nombre de las variables deben de empezar con una letra o una raya seguido de cualquier número de letras, números o rayas, estos nombres de variables son sensibles a mayúsculas y minúsculas.

6.8. Estructuras de Control

Las estructuras son para tener un mejor manejo en la programación de PHP ya que maneja varias sentencias. Los script de PHP están formados por varias sentencias, estas sentencias en ocasiones están conformadas por sentencias anidadas. Cabe mencionar que las sentencias se finalizan con punto y coma (;).

Sentencia if: Este tipo de sentencia permite ejecutar fragmentos de código, facilitando una gran flexibilidad para realizar ejecuciones condicionales ya que en ocasiones esta sentencia puede anidarse en otra sentencia *if*.

Sintaxis:

```
<?php
if (expr)
   sentencia
?>
```

***Sentencia else*:** Sentencia que extiende la sentencia *if* y verifica sí se cumple o no, determinada condición y sí no se cumple continua con la siguiente hasta finalizar las condiciones.

Ejemplo:

```
<?php
 if ($a > $b) {
    print "a es mayor que b";
 } else {
    print "a NO es mayor que b";
 }
?>
```

Bucle do..while: Con este bucle se comprueban las condiciones al final de cada iteración, garantizando la ejecución de la primera iteración en el bucle do.. while.

Ejemplo:

```
<?php
 $i = 0;
 do {
    print $i;
 } while ($i>0);
?>
```

Sentencia elseif: Esta sentencia esta formada por la combinación de *if* y *else*. También puede haber más de un *elseif* en una sentencia.

Ejemplo:

```
<?php
 if ($a > $b) {
    print "a es mayor que b";
 } elseif ($a == $b) {
    print "a es igual que b";
 } else {
    print "a es mayor que b";
 }
?>
```

Bucle while: Bucle que le indica a PHP que ejecute todas las sentencias que se encuentran anidadas dentro de la sentencia *while* hasta que sean evaluadas como verdadero, en caso de que *while* se valide como falso no se ejecutaran las sentencias que tenga.

Ejemplo:

```
<?php
 $i = 1;
 while ($i <= 10) {
    print $i++;
 }
 ?>
```

Bucle for: Este bucle esta formado por tres diferentes expresiones como se muestra a continuación:

Sintaxis:

```
for (expr1; expr2; expr3) sentencia
```

La primera expresión se ejecuta una sola vez, la segunda se evalúa al inicio de cada iteración, sí es verdadera se ejecuta la sentencia anidada y sí se evalúa falso termina el bucle. La tercera expresión se evalúa al final de cada iteración.

Ejemplo:

```
<?php
  for ($i = 1; $i <= 10; $i++) {
    print $i;
 }
?>
```

Opción break: Se utiliza para salir de los bucles *for*, *while*, o *switch*.

Ejemplo:

```
<?php
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list (, $val) = each ($arr)) {
   if ($val == 'stop') {
      break;
   }
   echo "$val<br>\n";
}
/* Uso de break. */
$i = 0;
while (++$i) {
   switch ($i) {
   case 5:
      echo "At 5<br>\n";
      break 1;  /* Salir del switch. */
   case 10:
      echo "Al 10; saliendo<br>\n";
      break 2;  /* Salir de switch y del while. */
   default:
      break;
   }
}
?>
```

Declare: La construcción declare define directivas de ejecución para un bloque de código.

Sintaxis:

```
declare (directiva) sentencia
```

Ejemplo:

```
<?php
// Estos son lo mismo:
// se puede usar este:
declare(ticks=1) {
   // script completo aqui
}

// o este:
declare(ticks=1);
// script completo aqui
?>
```

Opción continue: Esta opción es utilizada en las estructuras de bucle, para saltar el resto de la iteración actual y continuar con la siguiente iteración.

Ejemplo:

```
<?php
while (list ($key, $value) = each ($arr)) {
   if (!($key % 2)) { // skip odd members
      continue;
   }
   do_something_odd ($value);
}

$i = 0;
while ($i++ < 5) {
   echo "Outer<br>\n";
   while (1) {
      echo "  Middle<br>\n";
      while (1) {
         echo "  Inner<br>\n";
         continue 3;
      }
      echo "Esto nunca se imprime.<br>\n";
   }
   echo "Y esto tampoco.<br>\n";
}
?>
```

Sentencia switch: Esta sentencia se ejecuta línea por línea y se utiliza cuándo se requiere comparar una variable con diferentes valores ejecutando una parte del código.

Ejemplo:

```
<?php
if ($i == 0) {
   print "i es igual a 0";
} elseif ($i == 1) {
   print "i es igual a 1";
} elseif ($i == 2) {
   print "i es igual a 2";
}

switch ($i) {
   case 0:
      print "i es igual a 0";
```

```
    break;
  case 1:
    print "i es igual a 1";
    break;
  case 2:
    print "i es igual a 2";
    break;
}
?>
```

Otras estructuras de control:

- ✓ *Ticks:* Evento que ocurre por cada N sentencias de bajo nivel ejecutadas dentro del bloque *declare*.
- ✓ *return():* Finaliza rápidamente la ejecución de una función y regresa su argumento como valor de la función.
- ✓ *Las sentencias require(), include():* Incluye y evalúa un archivo especificado.
- ✓ *Las funciones require_once()* e *include_once():* Incluyen y evalúan el fichero especificado durante la ejecución del script. **[69]**

6.9. Adquisición de PHP

La descarga de PHP se puede realizar desde cualquiera de los miembros de la red del sitio PHP. Para realizar la descarga de PHP, solo se deben de seguir los siguientes pasos:

Paso 1: Se inicia con el acceso a la página principal de PHP. En esta página se selecciona la liga *downloads* (ver Figura 6.2). **[70]**

Figura 6.2: Elección de la liga downloads. **[70]**

Paso 2: Una vez accedido a la liga download, se descargan los archivos de la opción *Complete Source Code* (ver Figura 6.3). **[71]**

Figura 6.3: Escogiendo los archivos de PHP para código libre. **[71]**

Paso 3: Después se acceden a las ligas *PHP 5.1.4 (tar.bz2)* y *PHP 5.1.4 (tar.gz)*, se selecciona el servidor desde donde se descargaran los archivos de PHP (ver Figura 6.4). **[72]**

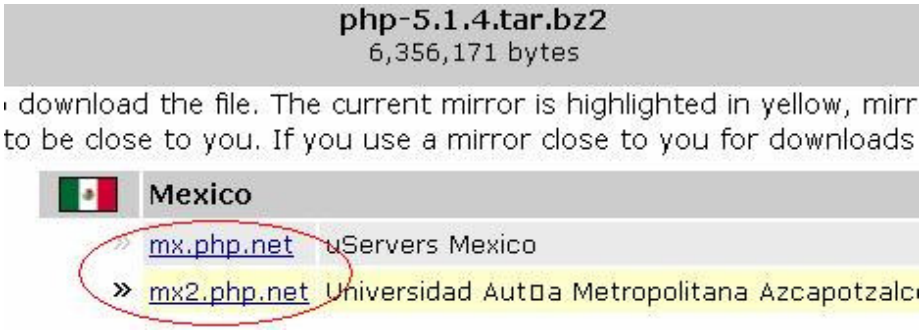


Figura 6.4: Eligiendo el servidor para descarga PHP. **[72]**

6.10. Antes de iniciar con la instalación.

Antes de iniciar con la instalación de PHP, primero se debe de ver que este instalado lo siguiente:

1.- Verificar que la versión de Apache sea 1.3.27 o más nueva, para ver la versión, solo se teclea en la línea de comando lo siguiente:

```
httpd -v
```

2.- Comprobar que el modulo *mod_so* este instado. En la mayoría de las ocasiones lo esta. Para mostrar la lista de todos los módulos y ver sí existe en esta lista, solo se teclea lo siguiente:

```
httpd -l.
```

3.- Comprobar sí esta instalada la utilidad apxs o apxs2 para uso con Apache 2. Este archivo se puede verificar sí existe en el directorio: **[N]**

```
/usr/sbin/apxs
```

6.11. Instalación de PHP como módulo de Apache

En el presente tema se describen los pasos pata realizara la instalación de PHP, en el sistema operativo Debian GNU/Linux.

Paso 1: Para iniciar con la instalación de PHP, se debe de contar con los archivos *.tar.gz* y *.tar* de PHP.

Paso 2: Se continua por desempaquetar los archivos.

```
gunzip php-versión.tar.gz
-tar xvf php-versión.tar
```

Paso 3: Crear un enlace a la ubicación donde se desempaqueto el archivo de PHP.

```
ln -s /usr/local/php-versión.x
```

Paso 4: Con la siguiente opción se accede al directorio de PHP.

```
cd /usr/local/php
```

Paso 5: En este paso se configurara PHP.

```
./configure –help
o
./configure –with-apxs2=/usr/local/apache2/bin/apxs –with-mysql
```

Paso 6: Se continua por compilar PHP.

```
make
```

Paso 7: Inicio de la Instalación de PHP.

```
make install
```

Paso 8: Se accede al directorio de PHP y se inicia la ejecución de *php.ini*.

```
cp php.ini-dist /usr/local/lib/php.ini
```

En este paso se puede editar el archivo *php.ini* para modificar opciones de PHP. En caso de que se requiera cambiar de lugar se usa lo siguiente:

```
–with-config-file-path=/some/path en el paso 5.
```

Paso 9: En el archivo *httpd.conf* se cargan los módulos necesarios, en este caso se cargaran los módulos de PHP 4 y 5.

```
Modulo de PHP 4:
    LoadModule php4_module modules/libphp4.so

Modulo de PHP 5
    LoadModule php5_module modules/libphp5.so
```

Paso 10: En el archivo *httpd.conf* se escribe lo siguiente:

```
AddType application/x-httpd-php .php .phtml
AddType application/x-httpd-php-source .phps
```

Paso 11: Reiniciar el servidor. **[69]**

```
/usr/local/apache-veción/bin/apachectl start
```

6.12. Soporte de PHP para bases de datos

Como ya se menciono anterior mente PHP tiene grandes ventajas; tal vez una de las mas grandes y favorables sea la capacidad de interactuar con diferentes bases de datos (ver Tabla 6.2), ya que facilita escribir interfaz vía Web para bases de datos.

BASE DE DATOS.	OPCIÓN DE CONFIGURACIÓN.
Adabas D	--with-adabas=*DIR*
dBase	--with-dbase
FilePro (read-only)	--with-filepro
mSQL	--with-msql=*DIR*
MySQL	--with-mysql=*DIR*
Oracle (OCI7 and OCI8)	--with-oracle=*DIR*.
PostgreSQL	--with-pgsql=*DIR*.
Solid	--with-solid=*DIR*
Sybase	--with-sybase=*DIR*
Velocis	--with-velocis=*DIR*
ODBC	--with-custom-odbc=*DIR*
Empress	?
Hyperwave	?
IBM DB2	?
Informix	?
Ingres	?
InterBase	?
FrontBase	?
Direct MS-SQL	?
Ovrimos	?
Unix dbm	?

Tabla 6.2: Bases de datos soportadas por PHP.

6.13. Seguridad

Al mencionar la palabra seguridad, se dice que es imposible tener una seguridad al cien por ciento, por esta razón se busca un equilibrio entre riesgos y funcionalidad.

Instalación como módulo de Apache: En este caso PHP toma los permisos del usuario de Apache, esto puede ocasionar problemas con respecto a la seguridad. Para realizar una mejor seguridad será necesario usar los mecanismos adecuados de autorización de Apache o se puede diseñar el modelo de acceso. Cuando la seguridad es estable PHP no podrá escribir archivos en los directorios del usuario, en otras palabras PHP no podrá acceder o modificar la base de datos.

Es importante que se tenga en cuenta que sí se escalan los permisos del usuario de Apache hasta el nivel de administrador puede ser extremadamente peligroso y comprometer todo el sistema. **[69]**

Seguridad del sistema de archivos: PHP puede manejar los archivos y directorios, esto permite controlar los archivos que pueden ser leídos del sistema de archivos. Se tienen que tener presentes los archivos con permisos de lectura global, para garantizar la seguridad de su contenido y que no represente algún riesgo para los usuarios con acceso al sistema de archivos.

Por otra parte, se recuerda que PHP fue diseñado para permitir acceso al nivel de usuarios del sistema de archivos, permitiendo la facilidad de escribir un script PHP que permita leer archivos del sistema, modificar conexiones tipo ethernet, enviar trabajos de impresión masivos, etc.

Datos enviados por el usuario: En ocasiones se dice que PHP es inseguro, pero este problema no radica en que PHP sea inseguro, sí no que dependerá de la forma en que el usuario escriba su código. En otras palabras cuando se requiera escribir código PHP se debe revisar para darle una mayor seguridad y no enviar una variable errónea.

Ocultar el código PHP: El ocultamiento de código PHP en ocasiones es una de las formas de seguridad más débiles, pero retrasan ataques que busquen debilidades en un sistema. En esta opción de seguridad se pueden destacar las siguientes:

1.- Establecer *expose_php = off* en el archivo *php.ini*, para reducir la información a posibles ataques.

2.- Configurar el servidor Web como Apache, para que procesen diferentes tipos de archivos con una directiva *.htaccess* o en el archivo de configuración de Apache, produciendo código de confusión.

Ejemplos:

Mostrar el código PHP como otro tipo de código.

```
AddType application/x-httpd-php .asp .py .pl
```

Mostrar el código PHP como código desconocido.

```
AddType application/x-httpd-php .bop .foo .133t
```

Mostrar el código PHP como código HTML. **[69]**

```
AddType application/x-httpd-php .htm .html
```

6.14. Ejemplo de la estructura de código PHP [69]

```
<html>
  <head>
    <title>Ejemplo</title>    //Título del código HTLM.
  </head>
  <body>

    <?php      //Inicio de código PHP.
    echo "Hola, &iexcl;soy un script PHP!";
    ?> //Fin del código PHP.

  </body>
</html>
```

6.15. Conexión a MySQL y selección de una base de datos

Para poder trabajar PHP con MySQL será necesario realizar una conexión al servidor usando la función mysql_connect():

```
$db_connection = mysql_connect (host, usuario, contraseña);
```

Donde la variable $db_connection, es el punto de referencia de PHP para la conexión, después de hacer la conexión satisfactoriamente, se selecciona la base de datos con la que se va a trabajar, escribiendo USE base_de_datos y la función mysql_select_bd():

```
mysql_select_db(base_de_datos);
```

A continuación se realiza un ejemplo de la conexión de MySQL haciendo uso de un archivo, donde los script PHP que se conecten a MySQL incluirán este archivo: **[M]**

Ejemplo: Escribir un documento PHP en un editor de texto:

```
1  <?php
2
3  //***** mysql_connect.inc *****
4  // ***** Script *****
5  // Desarrollado por:
6  // MySQL:
7  // Contacto:
8  // Fecha de creación:
9  // Última modificación:
10 // Este archivo contiene información de acceso a una base de datos.
11 // El archivo también establece una conexión con MySQL y
12 //  selecciona la base de datos.
13 // Información especifica de la base de datos:
14 DEFINE (DB_USER, "username");
15 DEFINE (DB_PASSWORD, "password");
16 DEFINE (DB_HOST, "localhost");
17 DEFINE (DB_NAME, "nombre_de_base_de_datos");
18
19 // Conexión a MySQL:
20 $db_connection = mysql_connect (DB_HOST, DB_USER,
   DB_PASSWORD);
21
22 // Seleccionar la base de datos:
23 mysql_select_db (DB_NAME);
24 ?>
```

Capítulo

7

Conformación de una base de datos distribuida

En los capítulos anteriores se ha comentado a cerca de las características de las bases de datos distribuidas y sistemas gestores de bases de datos; así como también se redactaron las herramientas necesarias para poder llegar a este último capítulo de todo el documento. En el presente capítulo se redacta acerca del diseño de una base de datos y los requisitos necesarios para su construcción, otro de los puntos mas importantes que se tocaran, es la descripción de dos ejemplos ilustrativos de las bases de datos distribuidas “bancaria y contabilidad”.

7.1. Introducción a la construcción de una base de datos distribuida

Como ya se tocaron los temas de base de datos, sistemas gestores de base de datos, así como los temas relacionados con una base de datos distribuida y las herramientas Open Source. En este capítulo se redacta como es construida una base de datos distribuida ya que estas deben de trabajar desde el punto de vista lógico, como si un sistema gestor de base de datos fuera ejecutado en una sola computadora administrando los datos.

Por otro lado hay que recordar que un sistema de bases de datos distribuidas esta formado por una serie de sitios o nodos conectados entre si mediante algún tipo de de red de comunicación; los cuales trabajan juntos, con el objetivo de que los usuarios de cualquier lugar pueda tener acceso a los datos desde cualquier punto de la red, como si los datos estuvieran en el nodo donde se encuentra el usuario.

Como resultado de lo anterior las partes de la base de datos distribuida son guardadas en distintas bases de datos "reales" ubicadas en diferentes nodos. Realmente es la unión lógica de esas bases de datos.

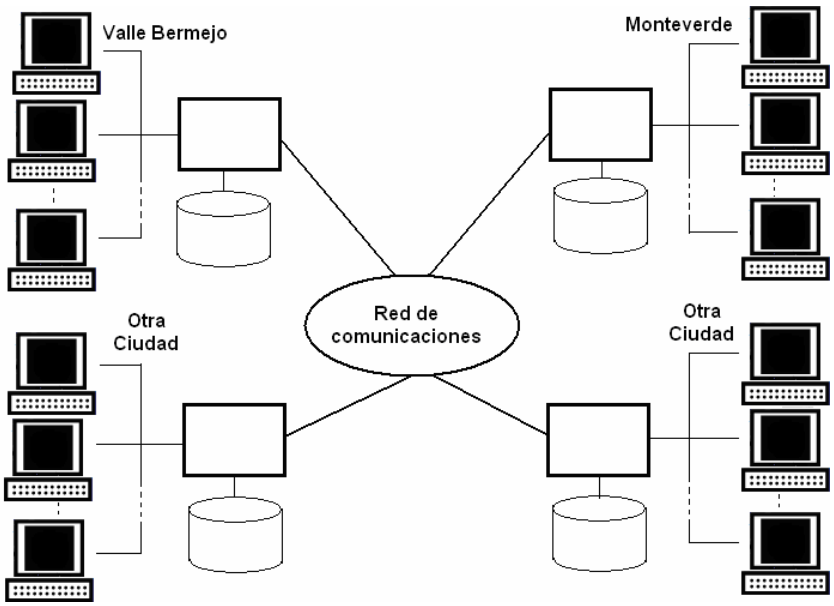


Figura 7.1: Sistema distribuido de base de datos representativo. **[O]**

Es frecuente que el conjunto de nodos se encuentren distribuidos físicamente y geográficamente (ver Figura 7.1), aún que basta con que estén dispersos lógicamente. Incluso dos sitios podrían estar en una sola computadora física. **[O]**

7.2. Diseño de una base de datos

Hay que resaltar que cuando se trabaja con un sistema gestor de base de datos como MySQL, el primer paso para construir y usar una base de datos, se deben de establecer y definir su estructura. El diseño de base de datos conocido como *modelo*

de base de datos es muy importante para lograr una administración a largo plazo. Usando el proceso de *normalización*, se logran eliminar las redundancias y otros problemas que pudieran perjudicar la integridad de la base de datos.

El conocimiento aquí plasmado ayudara a garantizar la viabilidad, rendimiento y fiabilidad de la base de datos bancaria, puesta como ejemplo mas adelante. **[M]**

7.2.1. Normalización

Desarrollada por el investigador de IBM llamado ***E. F. Codd*** a principios de los setentas. La normalización esta formada por un conjunto de reglas (formas normales), básicamente se usa las tres primeras formas normales las cuáles son suficientes para la construcción de la mayoría de las bases de datos.

Claves: Elementos de información con el objetivo de facilitar la identificación de una fila de información en una tabla o registro, las claves se pueden clasificar en:

- ✓ Claves principales: Es un identificador unívoco que debe cumplir con las reglas siguientes:

  1.- Poseer siempre un valor que no sea NULL.
  2.- Tener un valor que nunca cambie.
  3.- Poseer un valor univoco para cada registro de la tabla.

- ✓ Claves externas: Claves creadas para representar las claves principales. En la actualidad MySQL implementa claves externas cuando se utilizan tablas de tipo InnoDB, aunque en general ignora su existencia; por lo tanto, se dice que estas claves son una presencia teórica vinculante.

Cuando se requiere de asignar claves principales se deben de tomar en cuanta los siguientes puntos:

- ✓ Identificar los campos que tengan los tres criterios antes mencionados para dichas claves.
- ✓ En caso de que no exista una clave principal lógica, se tendrá que crear una.

Por otra parte hay que tener presente que MySQL solo admite una clave principal por tabla; sin embargo, es posible basar la clave en varias columnas. Lo correcto seria que la clave principal fuera un número entero, para lograr mejor rendimiento.

Relaciones: Al hablar de relaciones se refiere a la forma en que se relacionan los datos de una tabla con los de otra. Tales relaciones pueden ser *uno a uno*, *uno a varios* y *varios a varios*.

Una vez mencionado lo anterior se podrá comenzar con la normalización de la base de datos bancaria. **[M]**

7.2.2. Primera forma normal (FN1)

Para que la base de datos cumpla con esta forma normal, cada columna debe de contener un solo valor (atómico). Se dice que una tabla no cumple con esta forma normal cuando contiene un campo que no tiene relación con los demás.

Ajustando la base de datos a la FN1:

- ✓ Identificar los campos que tengan diversos elementos de información.
- ✓ Separar los campos encontrados, en campos independientes.
- ✓ Comprobar todos los campos creados en el paso anterior. **[M]**

7.2.3. Segunda forma normal (FN2)

Para lograr que la base de datos pase esta forma normal, debe cumplir primero con la FN1 y que cada columna de la tabla que no sea una clave que esté relacionada sólo con la clave principal. Una de las principales maneras de identificar que la base de datos no cumple con la FN2 es que existan muchos registros en una tabla que tengan valores idénticos en una columna.

Ajustando la base de datos a la FN2:

- ✓ Identificar los campos que no estén relacionados directamente con la clave principal.
- ✓ Crear las tablas necesarias.
- ✓ Crear y asignar claves principales, asegurando que cada tabla creada contenga su clave principal.
- ✓ Crear las claves externas que sean necesarias.

Otra manera de verificar que se cumple la FN2, es ver las relaciones entre las tablas, teniendo en cuenta que lo correcto seria reestructurar las relaciones varios a varios. **[M]**

7.2.4. Tercera forma normal

Para que la base de datos pase esta forma normal debe de haber cumplido con la FN2 y todas la columnas que no sean claves deben ser independiente de cualquier otra columna que tampoco se a clave. Se dice que los campos de una tabla que no sea clave, debe ser recíprocamente independientes.

Después de haber relazado las tres formas normales, probablemente ya no será necesario algún otro cambio en la base de datos. **[M]**

7.3. Ejemplo ilustrativo 1

Las bases de datos distribuidas normalmente se encuentran en varios lugares geográficos distintos, se administran de forma separada y poseen una interconexión más lenta. En estas se dan dos tipos de transacciones, las locales y las globales.

- ✓ Transacciones locales: Son las que acceden a los datos del único nodo en el cuál se inicio la transacción.
- ✓ Transacciones globales: Estas transacciones acceden a los datos localizados en un nodo diferente de aquél en el cuál se inicio la transacción o accede a datos de diferentes nodos.

Considerando el sistema bancario compuesto por cuatro sucursales localizado en cuatro diferentes ciudades (ver Figura 7.1). Cada sucursal tiene su propia computadora con una base de datos que alberga todas las cuentas abiertas en dicha sucursal. Así cada una de estas instalaciones se considera un nodo. También hay un único nodo que mantiene la información relativa a todas las sucursales del banco. Cada sucursal dispone de una relación *cuenta(Esquema-cuenta)*, donde:

esquema-cuenta = (nombre-sucursal, número-cuenta, saldo)

El nodo que contiene información acerca de las cuatro sucursales mantiene la relación *sucursal(Esquema-sucursal)*, donde:

esquema-sucursal = (nombre-sucursal, ciudad-sucursal, activos)

En la base de datos bancaria existen otras relaciones en los distintos nodos que serán ignorados para mostrar este ejemplo.

Para identificar los distintos tipos de transacciones, se considerara la transacción que suma 10.000 ptas. a la cuenta C-177, localizada en la sucursal Valle Bermejo. Las transacciones se consideran sí estas iniciaron en Valle Bermejo. Una transacción que transfiere 10.000 pts. desde la cuenta C-177 a la cuenta C-305, ubicada en la sucursal de Monteverde, es una transacción global, ya que como resultado de su ejecución se accede a datos de dos nodos diferentes.

Los siguientes puntos demuestran que esta base de datos puede ser un sistema de base de datos distribuido:

- ✓ Los diferentes nodos están informados de los demás.
- ✓ Aunque alguna relación este guardada en un diferente nodos, estos comparten un esquema global igual.
- ✓ Cada nodo da un entorno para la ejecución de transacciones locales y globales.
- ✓ En cada nodo se ejecuta el mismo software de gestión de bases de datos distribuidas.

Cabe mencionar que las transacciones globales son difíciles de realizar sí en los nodos se ejecuta un software diferente de gestión de base de datos. Estos sistemas son llamados *sistema de base de datos múltiples o sistemas distribuidos y heterogéneos de bases de datos*.

Continuando con la construcción de esta base de datos, primero se debe de tener en cuenta que una base de datos puede constituirse en un esquema E-R representado

por una serie de tablas, ya que cada conjunto de entidades de la base de datos y para cada conjunto de relaciones existe una única tabla a la que le asigna el nombre del conjunto de entidades o el conjunto de relaciones correspondientes. **[P]**

Para entender el diagrama E-R de la base de datos bancaria (ver Figura 7.3), primero se deben conocer las notaciones del diagrama E-R (ver Figura 7.2).

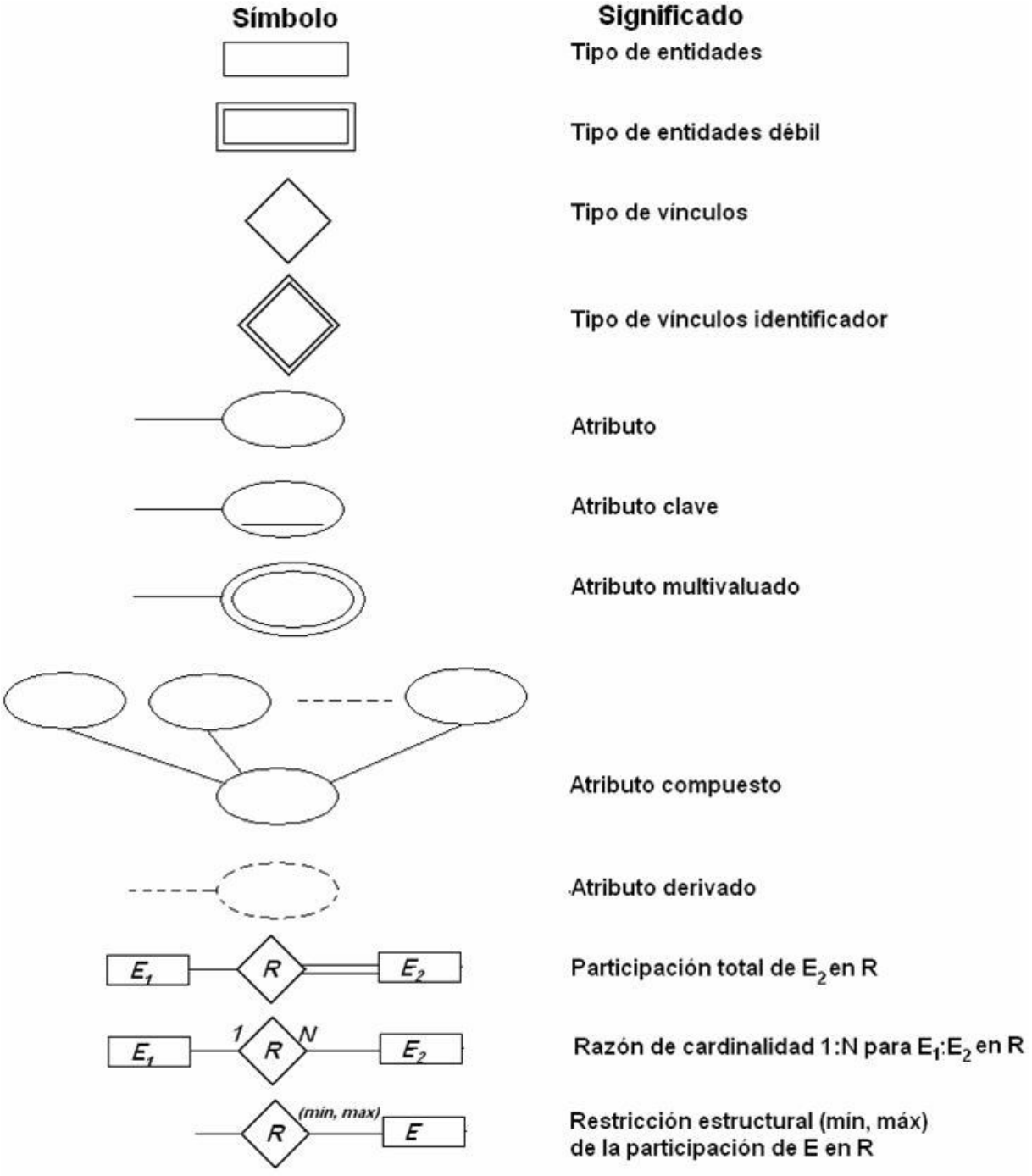


Figura 7.2: Resumen de la notación del diagrama E-R. **[P]**

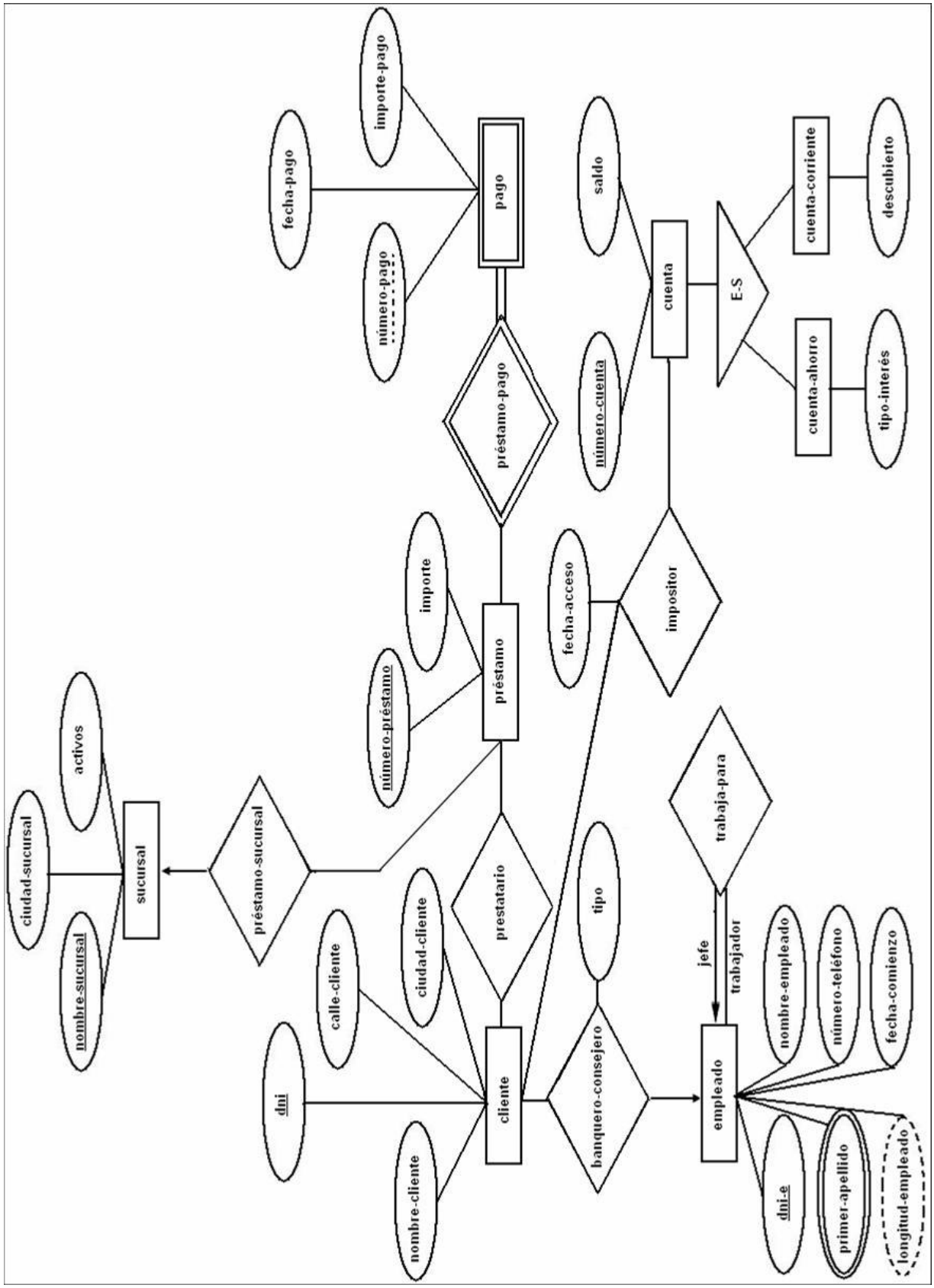


Figura 7.3: Diagrama E-R para el desarrollo bancario. **[P]**

7.3.1. Construyendo tablas

Después de mostrar el diagrama E-R de la base de datos bancaria se puede iniciar con la construcción de las tablas de la base de datos bancaria (ver Tablas 7.1, 7.2, 7.3). **[P]**

número-préstamo	importe
P-17	200.000
P-23	400.000
P-25	300.000
P-14	300.000
P-93	100.000
P-11	180.000
P-16	260.000

Tabla 7.1: Tabla préstamo.

nombre-cliente	dni	calle-cliente	ciudad-cliente
Santos	32112312	Mayor	Peguerinos
Gómez	19283746	Carretas	Cerceda
López	67789901	Mayor	Peguerinos
Pérez	55555555	Carretas	Cerceda
Rupérez	24466880	Ramblas	León
Abril	96396396	Preciados	Valsaín
Valdivieso	96396396	Goya	Vigo
Fernández	33557799	Jazmín	León
González	19283746	Arenal	La Granja

Tabla 7.2: Tabla cliente.

número-préstamo	número-pago	fecha-pago	importe-pago
P-17	5	10 mayo 1996	10.000
P-23	11	17 mayo 1996	15.000
P-15	22	23 mayo 1996	60.000
P-14	69	28 mayo 1996	100.0000
P-93	103	3 junio 1996	180.000
P-17	6	7 junio 1996	10.000
P-11	53	7 junio 1996	25.000
P-93	104	13 junio 1996	40.000
P-17	7	17 junio 1996	20.000
P-16	58	18 junio 1996	27.000

Tabla 7.3: Tabla Pago.

7.3.2. Relación entre tablas

A continuación se especifican las relaciones de la base de datos bancaria (ver Tablas 7.4, 7.5, 7.6, 7.7, 7.8, 7.9, 7.10, 7.11). Uno de los conjuntos principales de relaciones de esta base de datos implica los conjuntos de entidades:

cliente, con la clave primaria dni.
préstamo , con la clave primaria número-préstamo.

Debido a que el conjunto de relaciones no tiene atributos, la tabla prestatario posee las columnas *dni* y *número-préstamo* (ver Tabla 7.4).

dni	número-préstamo
32112312	P-17
19283746	P-23
67789901	P-15
55555555	P-14
24466880	P-93
19283746	P-11
96396396	P-17
33557799	P-16

Tabla 7.4: Tabla prestatario.

En la relación cuenta (ver Tabla 7.5) existen siete tuplas. Suponiendo que la *variable tupla t* [*nombre-sucursal*] para mostrar el valor de *t* en el atributo *nombre-sucursal.* Por lo tanto, *t* [*nombre-sucursal*] = <<Centro>> y *t* [*saldo*] = 100.000. De forma opcional, se puede decir que *t* [1] para mostrar el valor de la tupla *t* en el primer atributo (*nombre-sucursal*), *t* [2] para mostrar el *número-cuenta,* etc.

nombre-sucursal	número-cuenta	saldo
Centro	C-101	100.000
Becerril	C-215	140.000
Navacerrada	C-102	80.000
Collado Mediano	C-305	70.000
Galapagar	C-201	180.000
Moralzarzal	C-222	140.000
Galapagar	C-217	150.000

Tabla 7.5: Relación cuenta.

En la relación sucursal (ver Tabla 7.6) se dice que esta duplicada por utilizar atributos comunes en el esquema de relaciones, esta es una manera de relacionar las tuplas de relaciones diferentes.

nombre-sucursal	ciudad-sucursal	Activos
Centro	Arganzuela	1.800.000.000
Moralzarzal	La Granja	420.000.000
Navacerrada	Aluche	340.000.000
Becerril	Aluche	80.000.000
Collado Mediano	Aluche	1.600.000.000
Navas de la Asunción	Alcalá de Henares	60.000.000
Segovia	Cerceda	740.000.000
Cuenca	Villaverde	15.000.000

Tabla 7.6: Relación sucursal.

Por ejemplo sí se desea buscar información sobre las cuentas abiertas en la sucursal Arganzuela. Primero se tendrá que buscar en la relación *sucursal* para localizar los nombres de las sucursales localizadas en Arganzuela, en segundo término se

buscara en la relación *cuenta* para localizar la información sobre la cuenta buscada. Por tal razón se dice que en el modelo E-R el atributo *nombre-sucursal* representa el mismo conjunto de entidades en las dos relaciones.

Relación que describe la información sobre los clientes de la base de datos bancaria (ver Tabla 7.7).

nombre-cliente	calle-cliente	ciudad-cliente
Santos	Mayor	Peguerinos
Gómez	Carretas	Cerceda
López	Mayor	Pegueritos
Pérez	Carretas	Cerceda
Rupérez	Ramblas	León
Abril	Preciados	Valsaín
Valdivieso	Goya	Vigo
Fernández	Jazmín	León
González	Arenal	La Granja
Rodríguez	Yeserías	Cádiz
Amo	Embajadores	Arganzuela
Badorrey	Delicias	Valsaín

Tabla 7.7: Relación cliente.

Relación que tiene como objetivo describir la información entre clientes y las cuentas (ver Tabla 7.8). Esta relación se puede usar en caso de que un cliente tenga varias cuentas. **[P]**

nombre-cliente	número-cuenta
González	C-101
Gómez	C-215
López	C-102
Abril	C-305
González	C-201
Santos	C-217
Rupérez	C-222

Tabla 7.8: Relación impositor.

nombre-sucursal	número-préstamo	importe
Centro	P-17	200.000
Moralzarzal	P-23	400.000
Navacerrada	P-15	300.000
Centro	P-14	300.000
Becerril	P-93	100.000
Collado Mediano	P-11	180.000
Navacerrada	P-16	260.000

Tabla 7.9: Relación préstamo.

nombre-cliente	número-préstamo
Santos	P-17
Gómez	P-23
López	P-15
Sotoca	P-14
Pérez	P-93
Gómez	P-11
Valdivieso	P-17
Fernández	P-16

Tabla 7.10: Relación prestatario.

nombre-empleado	nombre-jefe
Arteaga	Benzoaga
Benzoaga	Erejalde
Conesa	Duarte
Duarte	Santos
Erejalde	Santos
Santos	Lara
Rupérez	Lara

Tabla 7.11: Relación jefe.

7.4. Ejemplo ilustrativo 2

En el presente ejemplo se diseñara y se crea una base de datos de contabilidad (ver Figura 7.4) orientada al mundo empresarial. El objetivo de esta base de datos será realizar un seguimiento de las facturas y gastos, con la facilidad de modificarse. Como primer pasó, se hará la normalización donde se asignaran claves. En este ejemplo solo se asignara una clave principal debido a que existe un dato unívoco, que siempre contendrá un valor que no cambiara nunca "*Número de factura*". **[M]**

Aplicando la primer forma normal (NF1): Continuando con este ejemplo de la base de datos, a hora se le aplicara la FN1 (ver Figura 7.5). Donde:

- ✓ Primero se identifican los campos que contengan más de un elemento de información. El resultado de esta identificación demuestra que "*Información del cliente"* y "*Tipo de gasto y descripción*", no se ajustan a la FN1.

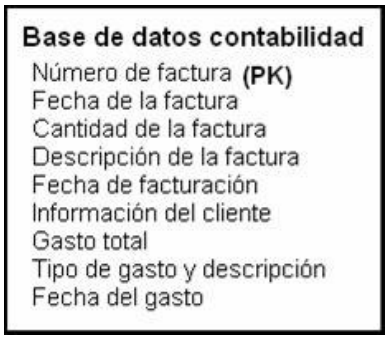


Figura 7.4: Creación de clave principal inicial. **[M]**

- ✓ El segundo paso consistirá en dividir los campos que no se ajustaron a la FN1, en campos independientes, donde:

"Información del cliente" será dividido en:
- Nombre del cliente
- Dirección del cliente
- Ciudad del cliente
- Estado del cliente
- Código postal del cliente
- Teléfono del cliente

"Tipo de gasto y descripción" se divide en:
- Tipo de gasto
- Descripción del gasto

- ✓ Por último se tendrá que comprobar que los campos creados pasan la FN1.

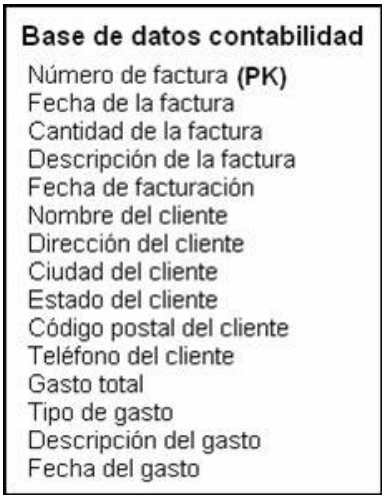


Figura 7.5: Una vez aplicada la FN1. **[M]**

Segunda forma normal (FN2): Como se menciono anteriormente, para hacer que la base de datos cumpla con esta forma normal, debe pasar primero la FN1 y que cada columna de la tabla que no sea una clave este relacionada con la clave principal.

- ✓ Para iniciar con este ajuste, primero se deben de identificar los campos que no se encuentren relacionados directamente con la clave principal. Una vez aplicado lo anterior se identifica que los campos "*Clientes, Facturas* y *Gastos*" no están relacionados con la clave principal.

- ✓ Verificar sí existen campos no relacionados con la clave principal, para crear las tablas necesarias. En este caso se crearan las tablas de los campos "*Clientes, Facturas* y G*astos*" (ver Figura 7.6).

- ✓ Una vez creadas las tablas, llega el turno de asignar o crear nuevas claves principales (ver Figura 7.7).

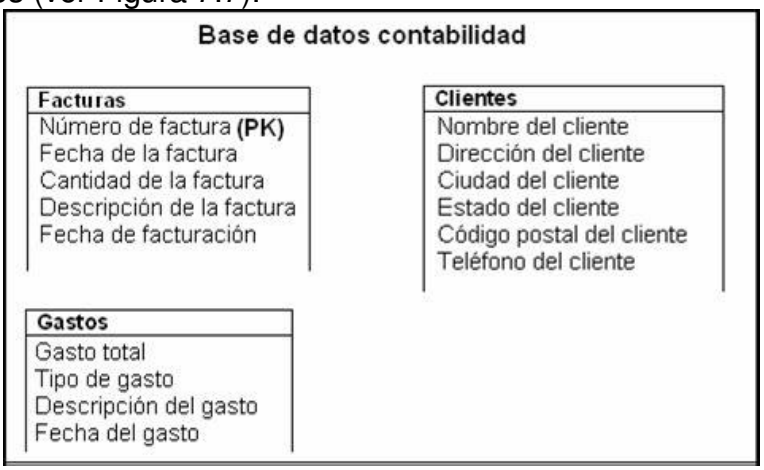


Figura 7.6: Creación de las tablas *Clientes, Facturas* y *Gastos*. **[M]**

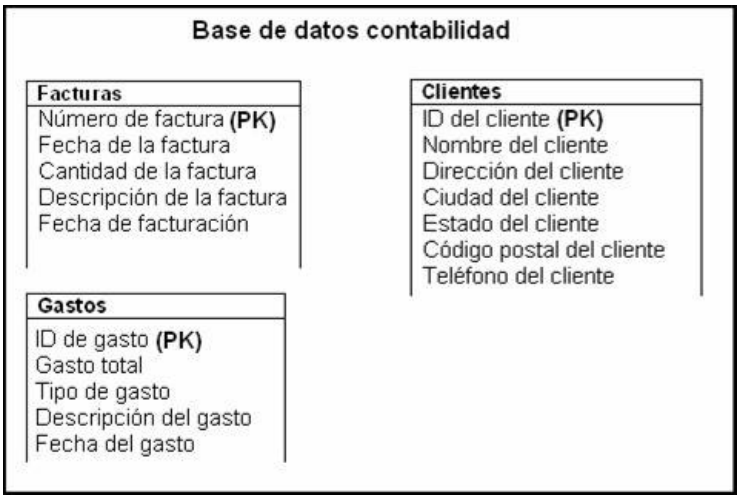


Figura 7.7: Asignación de claves principales. **[M]**

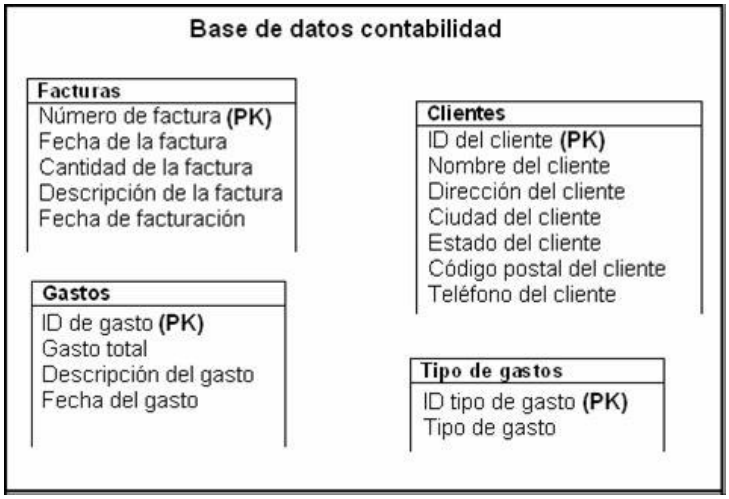


Figura 7.8: Creación de la tabla "*tipo de gastos*" y la creación de clave principal. **[M]**

- ✓ En este paso será necesario identificar nuevamente los campos que no tengan alguna relación con la clave principal y agregarle o crearle una clave principal (ver Figura 7.7); donde se puede identificar un problema con el campo "*Tipo de gasto*" ya que se podrían interpretar vario tipos de gastos, por esta razón se creara una nueva tabla (ver Figura 7.8).

- ✓ Para finalizar con la FN2, será necesario crear las claves externas (ver Figura 7.9) que permitan indicar los diferentes tipos de relaciones. **[M]**

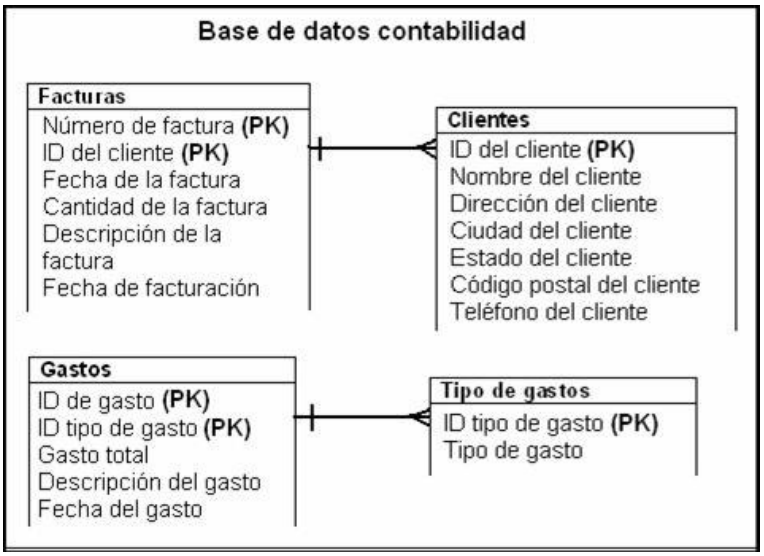


Figura 7.9: Incorporación de claves externas y relaciones entre datos y tablas. **[M]**

Tercera forma normal: Para que la base de datos contabilidad logre pasar esta tercer forma normal, será necesario que cumpla con la FN2 y que todas las columna que no sean claves, deben ser independiente de cualquier otra columna que tampoco sea clave.

Sí se han realizado correctamente los pasos anteriores probablemente no se harán cambios en la base de datos contabilidad, en caso de que sea necesario hacer un cambio, como por ejemplo:

Sí se quieren agregar los registros "*Nombre del contacto* y *Correo electrónico*" en la tabla *Factura* (ver Figura 7.10), se puede ver que la información se encuentra vinculada con las tablas "*Factura* y *Clientes*", lo recomendable es que estos datos se le asignen a la tabla *Clientes* para que la base de datos pueda cumplir con la tercera forma normal (ver Figura 7.11). **[M]**

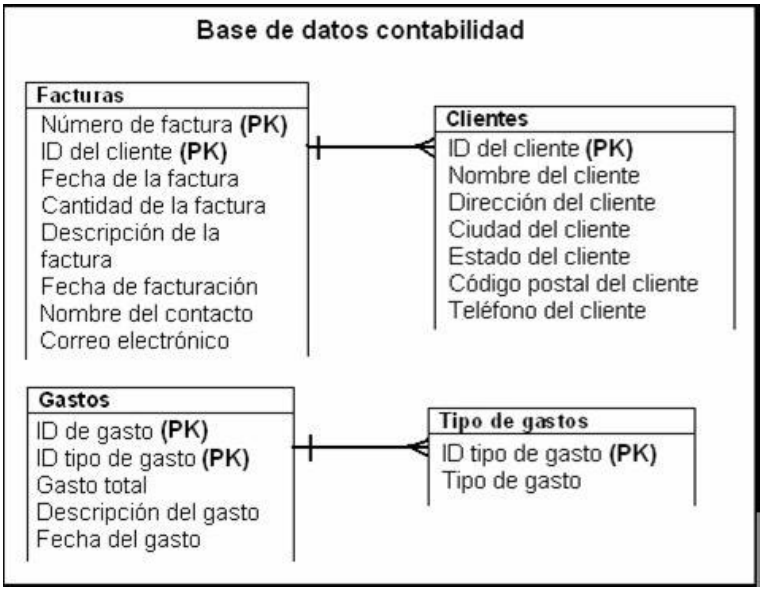


Figura 7.10: Alteración de la tabla *Facturas*. **[M]**

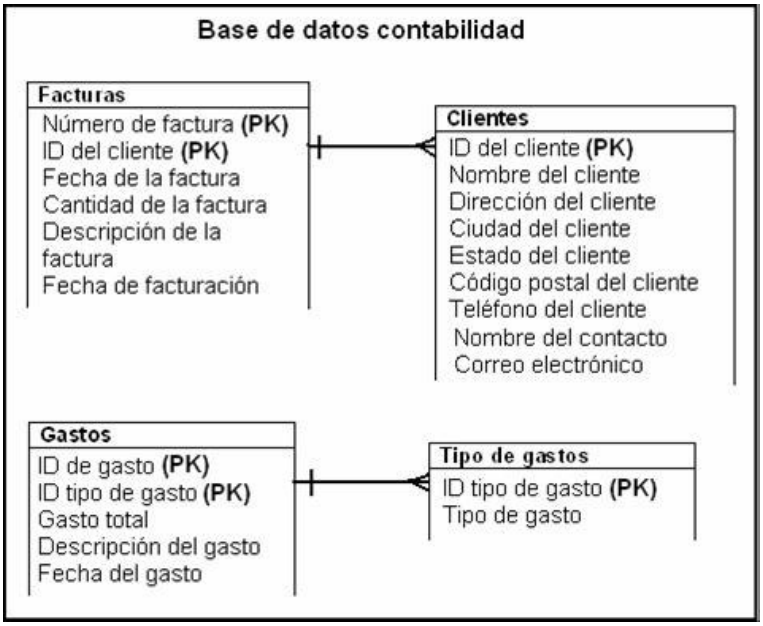


Figura 7.11: Incorporación de registros correctamente. **[M]**

Valores NULL y DEFAULT: La asignación de un valor NULL (vació). Lo ideal sería que todos los registros de la base de datos tuvieran algún valor, pero en la práctica no se generan de esta manera. Para imponer esta limitación a un campo, se pude añadir NOT NULL a la descripción del tipo de columna. En el caso donde gran parte de los registros tienen el mismo contenido, se puede presentar un valor predeterminado "DEFAULT" (ver Figura 7.12). **[M]**

Nombre de columna	Tabla	Tipo de columna
Número de factura	Factura	SMALLINT(4) UNSIGNED NOT NULL DEFAULT 0
ID del cliente	Factura	SMALLINT(3) UNSIGNED
Fecha de la factura	Factura	DATE NOT NULL
Cantidad de la factura	Factura	DECIMAL(3) UNSIGNED NOT NULL
Descripción de la factura	Factura	TINYTEXT
ID del cliente	Clientes	SMALLINT(3) UNSIGNED NOT NULL DEFAULT 0
Nombre del cliente	Clientes	VARCHAR(40) NOT NULL
Dirección del cliente	Clientes	VARCHAR(80)
Ciudad del cliente	Clientes	VARCHAR(30)
Estado del cliente	Clientes	CHAR(2)
Código postal del cliente	Clientes	MEDIUMINT(5) UNSIGNED
Teléfono del cliente	Clientes	VARCHAR(14)
Nombre del contacto	Clientes	VARCHAR(40)
Correo electrónico del contacto	Clientes	VARCHAR(60)
ID de gasto	Gasto	SMALLINT(4) UNSIGNED NOT NULL DEFAULT 0
ID tipo de gasto	Gasto	TINYINT(3) UNSIGNED
Gasto total	Gasto	DECIMAL(10,2) UNSIGNED NOT NULL
Descripción del gasto	Gasto	TINYTEXT
Fecha del gasto	Gasto	DATE
ID tipo de gasto	Tipo de gastos	TINYINT(3) UNSIGNED
Tipo de gasto	Tipo de gastos	VARCHAR(30)

Figura 7.12: Asignación de valores DEFAULT y descripciones NOT NULL. **[M]**

Creación de índices: Hay que recordar que los índices son un sistema usado por las bases de datos para mejorar su rendimiento global. Los índices que se crearán para la base de datos contabilidad serán del tipo "*INDEX, UNIQUE* y *PRIMARY KEY*" (ver Figura 7.13). **[M]**

Columna	Tipo de índice
Número de factura	PRIMARY KEY
ID del cliente	PRIMARY KEY
ID de gasto	PRIMARY KEY
ID tipo de gasto	PRIMARY KEY
Fecha de la factura	INDEX
Nombre del cliente	INDEX (o UNIQUE)

Figura 7.13: Índices de la base de datos contabilidad. **[M]**

Últimos pasos del diseño de la base de datos contabilidad: Para poder cumplir con el diseño de esta base de datos, se hace mención de las reglas para poder asignar de manera correcta los nombres de las tablas y columnas:

- ✓ Usar caracteres alfanuméricos.
- ✓ Limitar los nombres a menos de 64 caracteres.
- ✓ Usar el guión bajo (_) para separar las palabras.
- ✓ Escribir los nombres con letras minúsculas.
- ✓ Los nombres de las tablas van en plural y los de las columnas, en singular.
- ✓ Se recomienda hacer uso de las letras *id* en las columnas de claves principales y externa.
- ✓ En las tablas se deben de colocar, en primer lugar la clave principal y en seguida las claves externas.
- ✓ Los nombres de los campos deben ser unívocos.

Una vez cumplido estas reglas se dice que ha finalizado el diseño de la base de datos (ver Figura 7.14). **[M]**

Nombre de columna	Tabla	Tipo de columna
id_factura	Factura	SMALLINT(4) UNSIGNED NOT NULL DEFAULT 0
id_cliente	Factura	SMALLINT(3) UNSIGNED
fecha_factura	Factura	DATE NOT NULL
cantidad_factura	Factura	DECIMAL(3) UNSIGNED NOT NULL
descripción_factura	Factura	TINYTEXT
id_cliente	Clientes	SMALLINT(3) UNSIGNED NOT NULL DEFAULT 0
nombre_cliente	Clientes	VARCHAR(40) NOT NULL
dirección_cliente	Clientes	VARCHAR(80)
ciudad_cliente	Clientes	VARCHAR(30)
estado_cliente	Clientes	CHAR(2)
código_postal_cliente	Clientes	MEDIUMINT(5) UNSIGNED
teléfono_cliente	Clientes	VARCHAR(14)
nombre_contacto	Clientes	VARCHAR(40)
correo_electrónico_contacto	Clientes	VARCHAR(60)
id_gasto	Gasto	SMALLINT(4) UNSIGNED NOT NULL DEFAULT 0
id_tipo_gasto	Gasto	TINYINT(3) UNSIGNED
gasto_total	Gasto	DECIMAL(10,2) UNSIGNED NOT NULL
descripción_gasto	Gasto	TINYTEXT
fecha_gasto	Gasto	DATE
id_tipo_gasto	Tipo de gastos	TINYINT(3) UNSIGNED
tipo_gasto	Tipo de gastos	VARCHAR(30)

Figura 7.14: Nombres correctos de tablas y columnas. **[M]**

7.4.1. Creación de la base de datos y tablas

A continuación se hará uso de los comandos o sentencias que se mencionaron en el capítulo 5, estas sentencias se usarán para crear y usar la base de datos, así como para crear las tablas y verificar si estas fueron formadas de manera correcta.

Accediendo al monitor mysql.

```
mysql -u usuario -p
```

Creación de la base de datos.

```
CREATE DATABASE contabilidad;
```

Selección de la base de datos contabilidad.

```
USE contabilidad;
```

Creación de la tabla facturas.

```
CREATE TABLE facturas (
id_factura SMALLINT(4) UNSIGNED NOT NULL AUTO_INCREMENT,
id_cliente SMALLINT(3) UNSIGNED,
fecha_factura DATE NOT NULL,
cantidad_factura DECIMAL(10,2) UNSIGNED NOT NULL,
```

```
descripcion_factura TINYTEX,
PRIMARY KEY (id_factura),
INDEX (fecha_factura)
);
```

Creación de la tabla clientes.

```
CREATE TABLE clientes (
id_cliente SMALLINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
nombre_cliente VARCHAR(40) NOT NULL,
direccion_cliente VARCHAR(80),
ciudad_cliente VARCHAR(30),
estado_cliente CHAR(2),
codigo_postal_cliente MEDIUMINT(5) UNSIGNED,
telefono_cliente VARCHAR(14),
nombre_contacto VARCHAR(40),
correo_electronico_contacto VARCHAR(60),
PRIMARY KEY (id_cliente),
INDEX (nombre_cliente)
);
```

Creación de la tabla gastos.

```
CREATE TABLE gastos (
id_gasto SMALLINT(4) UNSIGNED NOT NULL AUTO_INCREMENT,
id_tipo_gasto TINYINT(3) UNSIGNED,
gasto_total DECIMAL(10,2) UNSIGNED,
descripcion_gasto TINYTEXT,
fecha_gasto DATE,
PRIMARY KEY (id_gasto)
);
```

Creación de la tabla tipos_gastos.

```
CREATE TABLE tipos_gastos (
id_tipo_gasto TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
tipo_gasto VARCHAR(30),
PRIMARY KEY (id_tipo_gasto)
);
```

Confirmar la existencia de las tablas. **[M]**

```
SHOW TABLES;
```

O

```
SHOW COLUMNS FROM facturas;
```

7.4.2. Conexión a MySQL y selección de la base de datos

Uno de los puntos importantes para poderse conectar a MySQL será la creación de un archivo PHP; en un editor de texto, el cuál será guardado con el nombre y extensión mysql_connect.inc o .php.

```
<?php
// Archivo que establece la conexión con MySQL y selecciona la base de datos contabilidad.
// Información de la base de datos:
DEFINE (DB_USER, "nombre de usuario");
DEFINE (DB_PASSWORD, "password");
DEFINE (DB_HOST, "localhost");
DEFINE (DB_NAME, "contabilidad");
// Conexión a MySQL.
$db_connection = mysql_connect (DB_HOST, DB_USER,
    DB_PASSWORD);
// Selección de la base de datos:
mysql_select_db (DB_NAME);
?>
```

Una vez creado el archive PHP, este debe de ser guardado fuera del directorio Web, para ser cargado en el servidor. **[M]**

7.4.3. Archivo para añadir registros a la base de datos

La presente página PHP se crea para logras añadir registros a la base de datos contabilidad a través de un navegador Web. **[M]**

```
<html>
<head>

 <title>agregar un tipo de gasto</title>
</head>
<body>
<?php

//página para insertar registros en la tabla tipo_gastos.

//Sí el formulario a sido remitido, gestionarlo.
if (isset($HTTP_POST_VARS['submit'])) {

   // Comprueba que sean llenados todos los registros requeridos.
   if (strlen($HTTP_POST_VARS['tipos_gastos']) > 0 )
      {

        //Incluye la página de conexión.
```

```
        require_once("../mysql_connect.inc");

        // Escriba la consulta.
        $query = "INSERT INTO tipos_gastos VALUES (NULL,
        '{$HTTP_POST_VARS[' tipos_gastos']}')";

        //Ejecución de la consulta.
        $query_result = mysql_query ($query);

        //Imprimir mensajes necesarios.
        if ($query_result)
                        {
                          echo '<b><font color="green">Los tipos de gastos se
                         agregaron agregados!</font></b>';
          }
          else
           {
             echo '<b><font color="red">El tipo de gastos no fue
    agregados!</font></b>';
           }

              //Liberar recursos y cerrar la conexión con la base de datos.
              mysql_close();

              // Imprimir un mensaje si no se puede introducir la categoría.
            } else
                      {
                         echo '<b><font color="red">Se le olvido llenar todas la
                         información!</font></b>';
                      }
      } else
           {
              //Cerrar el formulario HTML.
              ?>
                    Agregar una nueva categoría en la tabla tipos_gastos:<br/>
                   <form action="add_expense_category.php" method="post">
                   <input type="text" name=" tipos_gastos" size="30"
                  maxlength="30"/></p>
                   <input type="submit" name="submit" value="Submit"/>
              <?php
} // Fin del condicional "submit" principal.

?>

</body>
</html>
```

7.4.4. Archivo para hacer consultas a la BD

```
<html>
<body>
<?php

$db_connection = mysql_connect("DB_HOST, DB_USER, DB_PASSWORD");

mysql_select_db("DB_NAME", $db_connection);

$result = mysql_query("SELECT numero_factura, id_cliente FROM facturas",
$db_connection);

if ($row = mysql_fetch_array($result)){
echo "<table border = '1'> \\n";
echo "<tr> \\n";
echo "<td><b>Número de factura</b></td> \\n";
echo "<td><b>ID Clinte</b></td> \\n";
echo "</tr> \\n";
do {
echo "<tr> \\n";
echo "<td>".$row["numero_factura"]."</td> \\n";
echo "<td>".$row["id_cliente"]."</td>\\n";
echo "</tr> \\n";
} while ($row = mysql_fetch_array($result));
echo "</table> style="margin-left: 50">echo "</table> \\n";
} else {
echo "¡ La base de datos está vacía !";
}
?>
</body>
</html>
```

En este script se introducen dos novedades, la más obvia es la sentencia de control while(), que tiene un funcionamiento similar al de otros lenguajes, ejecuta una cosa mientras la condición sea verdadera. En esta ocasión while() evalúa la función mysql_fetch_array(), que devuelve un array con el contenido del registro actual (que se almacena en $row) y avanza una posición en la lista de registros devueltos en la consulta SQL. De esta forma se logra referenciar a los campos por su nombre ($row["id_cliente"]).

Con la sentencia if/else, se asigna a $row el primer registro de la consulta, y en caso de no haber ninguno (else) se muestra un mensaje ("No se ha encontrado..."). Mientras que con la sentencia do/while, se asegura que se muestren todos los registros devueltos por la consulta en caso de haber más de uno. Hay que destacar la utilización del punto (.), como operador para concatenar cadenas. **[73]**

7.4.5. Mapeo de una base de datos

El mapeo de una base de datos consiste en la búsqueda de una base de datos con la cual de se va a trabajar, esto se logra mediante el uso de una base de datos la cual es la principal (ver Figura 7. 15)

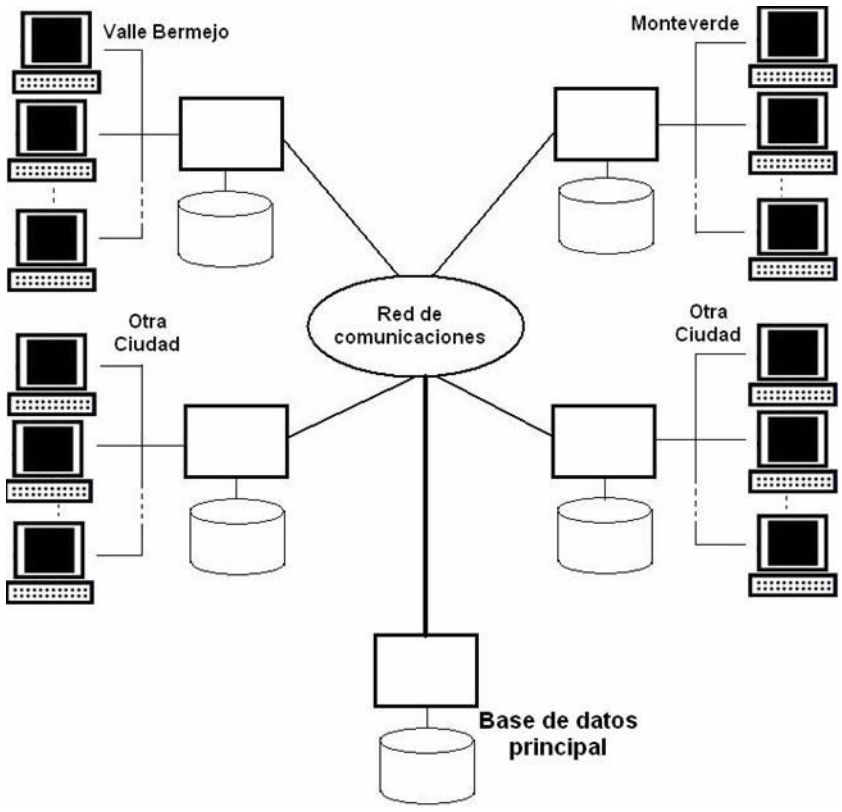


Figura 7.15: Agregando una base de datos principal. **[O]**

Haciendo referencia al ***ejemplo ilustrativo 1*** "sistema bancario" en el cual se le agrega una base de datos principal para que pueda existir un *mapeo de base de datos*. Esta base de datos permitirá elegir la base de datos con la que se desee trabajar, donde la tabla de dicha base de datos (ver Tablas 7.12) contiene información para conectar la base de datos con la que se quiera trabajar.

user_id	usuario	pasword	host_name	db_name	nombre_sucursal
1	Miguel_G	*****	vab	valle_bermejo	Centro
2	Mauro_G	*****	mov	monteverde	Moralzarzal
3	Alejandra_C	*****	otc	otra_ciudad	Navacerrada
4	Ruben_L	*****	otc	otra_ciudad	Becerril
. . .	. . .	. . .	. . .	. . .	Collado Mediano
. . .	. . .	. . .	. . .	. . .	Navas de la Asunción
. . .	. . .	. . .	. . .	. . .	Segovia
. . .	. . .	. . .	. . .	. . .	Cuenca

Tabla 7.12: Tabla de la base de datos principal.

7.4.6. Usar varias bases de datos en el mismo script

En ocaciones por especificaciones del programa es necesario el uso de diferentes bases de datos en el mismo programa. PHP proporciona la instrucción mysql_select_db() que permite escoger la base de datos que se requiera usar para ejecutar los SQL.

Ejemplo:

```
<?php
 $conexio = mysql_connect("localhost","usuario","passwod");

 $sql="SELECT * FROM usuario_valle_vermejo";


 //Seleccionamos trabajar con la BD valle_mermejo

 mysql_select_db("valle_mermejo");

 $result=mysql_query($sql);
 $row= mysql_fetch_array($result);
 $ultim_id=$row["user_id"];

 $sql="SELECT * FROM usuario WHERE id>".$ultim_id." ORDER BY id LIMIT 0,5";

 //Seleccionamos trabajar con la BD monteverde

 mysql_select_db("monteverde");
 $result=mysql_query($sql);

 ....
 ....
 ....
?>
```

mysql_select_db() establece la base activa que estará asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto. Si no hay ningún enlace abierto, la función intentara establecer un enlace como si se llamara a mysql_connect(). **[73]**

Conclusiones

Conclusiones.

El surgimiento de nuevas tecnologías, y en especial de tecnologías de información, ha abierto nuevas corrientes para el desarrollo de sistemas cuyas capacidades de guardado y distribución de datos, es un todo para las empresas e instituciones de diferentes áreas, destinados al mejoramiento y buen desempeño del manejo de información que operan, surgiendo las bases de datos distribuidas, por esta razón se describieron los principales requisitos para realizar las bases de datos distribuidas.

En este documento se presentan los requisitos principales para realizar una base de datos distribuida con software de código abierto (Open Source). Se ha estudiado las ventajas, desventajas, diseño, arquitectura, consultas, transacciones y concurrencia de una base de datos, sistema gestor de base de datos y bases de datos distribuida; así como las principales características de Debían GNU/Linux, servidor Apache, MySQL y PHP.

A lo largo de este documento se ha intentado dar una visión general y genérica de los problemas y características que contiene el diseño de una base de datos distribuida. Se ha hecho hincapié en las técnicas de fragmentación horizontal y vertical através de métodos y algoritmos usados en la realización del diseño. Con el objeto de que el lector no tenga problemas para su comprensión, las técnicas son sencillas y se ha procurado incluir algunos ejemplos para facilitar la comprensión de estos.

Con respecto a los sistemas gestores de bases de datos se ase mención de los tipos que existen y su clasificación, así como sus características principales y la incorporación de las dos herramientas fundamentales, para la definición y la manipulación de los datos. El lenguaje de definición de datos (DDL, Data Definition Language), proporciona los medios suficientes para definir los datos con precisión y la definición de la estructura. El otro lenguaje es el de manipulación de datos (DML, Data Manipulation/ Management Language), encargado de proveer a los usuarios el acceso y manipulación de los datos. Pueden diferenciarse en procedimentales y no procedimentales, estos se encargan de la recuperación, inserción, eliminación y modificación de datos almacenados en la base de datos.

Haciendo referencia al software de código abierto (Open Source), se concluye que este posee grandes ventajas como:

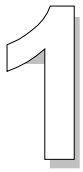
> ***Acceso al código fuente:*** con la ventaja de estudiarlo o modificarlo, para corregir errores, adaptarlo o añadir más prestaciones. De esta manera es viable para la educación.
>
> ***Gratuito:*** Esto quiere decir que no se tendrá que pagar alguna cantidad elevada, por concepto de licencias de uso, en caso de algún pago, esté será por concepto de gastos de empaquetamiento, distribución y valores añadidos, ya que por el software de forma binaria o en la forma de código fuente, puede obtenerse libremente.

Evitar monopolios de software propietario: No depender de un único fabricante del software. Esto tiene importancia cuando se trata de una organización como, una empresa o institución, los cuales no pueden ponerse en manos de una única solución y pasar a depender exclusivamente de ella.

Un modelo de avance: No basado en la ocultación de información, sino en la comparición del conocimiento, para alcanzar avances de manera rápida, eficiente, con mejor calidad, con la ventaja de decisiones tomadas por el consenso de una comunidad, y no en los caprichos de empresas desarrolladoras de software propietario.

Para concluir con esta parte, vale la pena decir que el uso de software de código abierto para la creación y el manejo de bases de datos distribuidas, en un futuro no muy lejano ganara terreno ante el software con propietario. En la actualidad el software de código abierto ha ido creciendo a pasos agigantados, por las aportaciones de diversos desarrolladores de todo el mundo, de esta manera se va haciendo mas robusto, eficiente.

Anexo



Licencia Pública General GNU (GPL)

Licencia Pública General GNU (GPL)

Términos y condiciones para la copia, distribución y modificación

- ✓ Esta Licencia se aplica a cualquier programa u otro tipo de trabajo que contenga una nota colocada por el tenedor del copyright diciendo que puede ser distribuido bajo los términos de esta Licencia Pública General. En adelante, «Programa» se referirá a cualquier programa o trabajo que cumpla esa condición y «trabajo basado en el Programa» se referirá bien al Programa o a cualquier trabajo derivado de él según la ley de copyright. Esto es, un trabajo que contenga el programa o una porción de él, bien en forma literal o con modificaciones y/o traducido en otro lenguaje. Por lo tanto, la traducción está incluida sin limitaciones en el término «modificación». Cada concesionario (permisos) será denominado «usted».

  Cualquier otra actividad que no sea la copia, distribución o modificación no está cubierta por esta Licencia, está fuera de su ámbito. El acto de ejecutar el Programa no está restringido, y los resultados del programa están cubiertos únicamente si sus contenidos constituyen un trabajo basado en el Programa, independientemente de haberlo producido mediante la ejecución del programa. El que esto se cumpla, depende de lo que haga el programa.

- ✓ Usted puede copiar y distribuir copias literales del código fuente del Programa, según lo has recibido, en cualquier medio, supuesto que de forma adecuada y bien visible publique en cada copia un anuncio de copyright adecuado y un repudio de garantía, mantenga intactos todos los anuncios que se refieran a esta Licencia y a la ausencia de garantía, y proporcione a cualquier otro receptor del programa una copia de esta Licencia junto con el Programa.

- ✓ Puede cobrar un precio por el acto físico de transferir una copia, y puede, según su libre albedrío, ofrecer garantía a cambio de unos honorarios. Puede modificar su copia o copias del Programa o de cualquier porción de él, formando de esta manera un trabajo basado en el Programa, y copiar y distribuir esa modificación o trabajo bajo los términos del apartado 1, antedicho, supuesto que además cumpla las siguientes condiciones:

  a. Debe hacer que los ficheros modificados lleven anuncios prominentes indicando que los ha cambiado y la fecha de cualquier cambio.

  b. Debe hacer que cualquier trabajo que distribuya o publique y que en todo o en parte contenga o sea derivado del Programa o de cualquier parte de él sea licenciada como un todo, sin carga alguna, a todas las terceras partes y bajo los términos de esta Licencia.

  c. Si el programa modificado lee normalmente órdenes interactivamente cuando es ejecutado, debe hacer que, cuando comience su ejecución para

ese uso interactivo de la forma más habitual, muestre o escriba un mensaje que incluya un anuncio de copyright y un anuncio de que no se ofrece ninguna garantía (o por el contrario que sí se ofrece garantía) y que los usuarios pueden redistribuir el programa bajo estas condiciones, e indicando al usuario cómo ver una copia de esta licencia. (Excepción: si el propio programa es interactivo pero normalmente no muestra ese anuncio, no se requiere que su trabajo basado en el Programa muestre ningún anuncio).

Estos requisitos se aplican al trabajo modificado como un todo. Si partes identificables de ese trabajo no son derivadas del Programa, y pueden, razonablemente, ser consideradas trabajos independientes y separados por ellos mismos, entonces esta Licencia y sus términos no se aplican a esas partes cuando sean distribuidas como trabajos separados. Pero cuando distribuya esas mismas secciones como partes de un todo que es un trabajo basado en el Programa, la distribución del todo debe ser según los términos de esta licencia, cuyos permisos para otros permisos se extienden al todo completo, y por lo tanto a todas y cada una de sus partes, con independencia de quién la escribió.

Por lo tanto, no es la intención de este apartado reclamar derechos o desafiar sus derechos sobre trabajos escritos totalmente por usted mismo. El intento es ejercer el derecho a controlar la distribución de trabajos derivados o colectivos basados en el Programa.

Además, el simple hecho de reunir un trabajo no basado en el Programa con el Programa (o con un trabajo basado en el Programa) en un volumen de almacenamiento o en un medio de distribución no hace que dicho trabajo entre dentro del ámbito cubierto por esta Licencia.

- ✓ Puede copiar y distribuir el Programa (o un trabajo basado en él, según se especifica en el apartado 2, como código objeto o en formato ejecutable según los términos de los apartados 1 y 2, supuesto que además cumpla una de las siguientes condiciones:

  a. Acompañarlo con el código fuente completo correspondiente, en formato electrónico, que debe ser distribuido según se especifica en los apartados 1 y 2 de esta Licencia en un medio habitualmente utilizado para el intercambio de programas, o

  b. Acompañarlo con una oferta por escrito, válida durante al menos tres años, de proporcionar a cualquier tercera parte una copia completa en formato electrónico del código fuente correspondiente, a un coste no mayor que el de realizar físicamente la distribución del fuente, que será distribuido bajo las condiciones descritas en los apartados 1 y 2 anteriores, en un medio habitualmente utilizado para el intercambio de programas, o

c. Acompañarlo con la información que recibiste ofreciendo distribuir el código fuente correspondiente. (Esta opción se permite sólo para distribución no comercial y sólo si usted recibió el programa como código objeto o en formato ejecutable con tal oferta, de acuerdo con el apartado b anterior).

Por código fuente de un trabajo se entiende la forma preferida del trabajo cuando se le hacen modificaciones. Para un trabajo ejecutable, se entiende por código fuente completo todo el código fuente para todos los módulos que contiene, más cualquier fichero asociado de definición de interfaces, más los guiones utilizados para controlar la compilación e instalación del ejecutable. Como excepción especial el código fuente distribuido no necesita incluir nada que sea distribuido normalmente (bien como fuente, bien en forma binaria) con los componentes principales (compilador, kernel y similares) del sistema operativo en el cuál funciona el ejecutable, a no ser que el propio componente acompañe al ejecutable.

Si la distribución del ejecutable o del código objeto se hace mediante la oferta acceso para copiarlo de un cierto lugar, entonces se considera la oferta de acceso para copiar el código fuente del mismo lugar como distribución del código fuente, incluso aunque terceras partes no estén forzadas a copiar el fuente junto con el código objeto.

- ✓ No puede copiar, modificar, sublicenciar o distribuir el Programa excepto como prevé expresamente esta Licencia. Cualquier intento de copiar, modificar sublicenciar o distribuir el Programa de otra forma es inválida, y hará que cesen automáticamente los derechos que te proporciona esta Licencia. En cualquier caso, las partes que hayan recibido copias o derechos de usted bajo esta Licencia no cesarán en sus derechos mientras esas partes continúen cumpliéndola.

- ✓ No está obligado a aceptar esta licencia, ya que no la ha firmado. Sin embargo, no hay hada más que le proporcione permiso para modificar o distribuir el Programa o sus trabajos derivados. Estas acciones están prohibidas por la ley si no acepta esta Licencia. Por lo tanto, si modifica o distribuye el Programa (o cualquier trabajo basado en el Programa), está indicando que acepta esta Licencia para poder hacerlo, y todos sus términos y condiciones para copiar, distribuir o modificar el Programa o trabajos basados en él.

- ✓ Cada vez que redistribuya el Programa (o cualquier trabajo basado en el Programa), el receptor recibe automáticamente una licencia del permiso original para copiar, distribuir o modificar el Programa, de forma sujeta a estos términos y condiciones. No puede imponer al receptor ninguna restricción más sobre el ejercicio de los derechos aquí garantizados. No es usted responsable de hacer cumplir esta licencia por terceras partes.

- ✓ Si como consecuencia de una resolución judicial o de una alegación de infracción de patente o por cualquier otra razón (no limitada a asuntos relacionados con patentes) se le imponen condiciones (ya sea por mandato judicial, por acuerdo o por cualquier otra causa) que contradigan las condiciones de esta Licencia, ello no le exime de cumplir las condiciones de esta Licencia. Si no puede realizar distribuciones de forma que se satisfagan simultáneamente sus obligaciones bajo esta licencia y cualquier otra obligación pertinente entonces, como consecuencia, no puede distribuir el Programa de ninguna forma. Por ejemplo, si una patente no permite la redistribución libre de derechos de autor del Programa por parte de todos aquellos que reciban copias directas o indirectamente a través de usted, entonces la única forma en que podría satisfacer tanto esa condición como esta Licencia sería evitar completamente la distribución del Programa.

  Si cualquier porción de este apartado se considera inválida o imposible de cumplir bajo cualquier circunstancia particular ha de cumplirse el resto y la sección por entero ha de cumplirse en cualquier otra circunstancia.

  No es el propósito de este apartado inducirle a infringir ninguna reivindicación de patente ni de ningún otro derecho de propiedad o impugnar la validez de ninguna de dichas reivindicaciones. Este apartado tiene el único propósito de proteger la integridad del sistema de distribución de software libre, que se realiza mediante prácticas de licencia pública. Mucha gente ha hecho contribuciones generosas a la gran variedad de software distribuido mediante ese sistema con la confianza de que el sistema se aplicará consistentemente. Será el autor/donante quien decida si quiere distribuir software mediante cualquier otro sistema y una licencia no puede imponer esa elección.

  Este apartado pretende dejar completamente claro lo que se cree que es una consecuencia del resto de esta Licencia.

- ✓ Si la distribución y/o uso de el Programa está restringida en ciertos países, bien por patentes o por interfaces bajo copyright, el tenedor del copyright que coloca este Programa bajo esta Licencia puede añadir una limitación explícita de distribución geográfica excluyendo esos países, de forma que la distribución se permita sólo en o entre los países no excluidos de esta manera. En ese caso, esta Licencia incorporará la limitación como si estuviese escrita en el cuerpo de esta Licencia.

- ✓ La Free Software Foundation puede publicar versiones revisadas y/o nuevas de la Licencia Pública General de tiempo en tiempo. Dichas nuevas versiones serán similares en espíritu a la presente versión, pero pueden ser diferentes en detalles para considerar nuevos problemas o situaciones. Cada versión recibe un número de versión que la distingue de otras. Si el Programa especifica un número de versión de esta Licencia que se refiere a ella y a «cualquier versión posterior», tienes la opción de seguir los términos y condiciones, bien de esa versión, bien de cualquier versión posterior publicada

por la Free Software Foundation. Si el Programa no especifica un número de versión de esta Licencia, puedes escoger cualquier versión publicada por la Free Software Foundation.

- ✓ Si quiere incorporar partes del Programa en otros programas libres cuyas condiciones de distribución son diferentes, escribe al autor para pedirle permiso. Si el software tiene copyright de la Free Software Foundation, escribe a la Free Software Foundation: algunas veces hacemos excepciones en estos casos. Nuestra decisión estará guiada por el doble objetivo de de preservar la libertad de todos los derivados de nuestro software libre y promover el que se comparta y reutilice el software en general.

AUSENCIA DE GARANTÍA

- ✓ Como el programa se licencia libre de cargas, no se ofrece ninguna garantía sobre el programa, en toda la extensión permitida por la legislación aplicable. Excepto cuando se indique de otra forma por escrito, los tenedores del copyright y/u otras partes proporcionan el programa «tal cuál», sin garantía de ninguna clase, bien expresa o implícita, con inclusión, pero sin limitación a las garantías mercantiles implícitas o a la conveniencia para un propósito particular. Cualquier riesgo referente a la calidad y prestaciones del programa es asumido por usted. Si se probase que el Programa es defectuoso, asume el coste de cualquier servicio, reparación o corrección.

- ✓ En ningún caso, salvo que lo requiera la legislación aplicable o haya sido acordado por escrito, ningún tenedor del copyright ni ninguna otra parte que modifique y/o redistribuya el Programa según se permite en esta Licencia será responsable ante usted por daños, incluyendo cualquier daño general, especial, incidental o resultante producido por el uso o la imposibilidad de uso del Programa (con inclusión, pero sin limitación a la pérdida de datos o a la generación incorrecta de datos o a pérdidas sufridas por usted o por terceras partes o a un fallo del Programa al funcionar en combinación con cualquier otro programa), incluso si dicho tenedor u otra parte ha sido advertido de la posibilidad de dichos daños.

FIN DE TÉRMINOS Y CONDICIONES

Glosario

Glosario

.MYD: Extensión de archivos de MySQL.

.MYI: Extensión de archivos de MySQL.

Atributos: Es cada una de las cualidades, propiedades o características de un elemento.

Affero GPL: Es íntegramente una GPLv2 con una cláusula nueva (sección 2(d)): añade la obligación de distribuir el software si éste se ejecuta para ofrecer servicios a través de una red de ordenadores.

Apple: Casa fabricante de ordenadores, creadora de ordenadores como los Apple II, Lisa, Macintosh e iMac.

Archivos .deb: Formato de archivo con el que se empaquetan los binarios de Debian.

Apt (Advanced Packaging Tool): Acrónimo para sistemas de gestión de paquetes creado por el proyecto Debian. APT simplifica, la instalación y eliminación de programas en los sistemas Linux.

ASCII (American Standard Code for Information Interchange): Código estándar utilizado para representar letras, símbolos y ciertos códigos.

AGP (Advanced Graphics Port): Tipo de slot dedicado en exclusiva a tarjetas gráficas.

AMD (Advanced Micro Devices): compañía mundial productora de microprocesadores y uno de los más importantes fabricantes de memoria flash.

ATAPI: Interfaz de programación para sistemas de bus AT. Es un estándar que siguen la mayoría de los lectores de CdRom.

Athlon XP: Es el nombre que recibe una gama de microprocesadores compatibles con la arquitectura x86, diseñados por AMD.

apachectl: Script para controlar el proceso demonio, con comandos simples como apachectl start y apachectl stop.

BACKUP TABLE: Comando obsoleto en MySQL 5.0. Copia al directorio de base de datos el mínimo número de ficheros de tablas necesarias para restaurar la tabla, tras volcar cualquier cambio almacenado en el buffer a disco.
API´s (Interfaz de Programación de Aplicaciones): Conjunto de especificaciones de comunicación entre componentes software.

APACHE: Servidor de páginas Web. Hoy por hoy líder del mercado de servidores, por delante de soluciones propietarias.

BDB: Tablas de mysql 5.0, que pueden tener hasta 31 índices y son más grandes que las MyISAM.

BDD: Conjunto de información guardada en diferentes lugares de forma organizada la cuál posteriormente se puede extraer uno o varios datos necesarios.

BIOS (Basic Input-Output System): Programa incorporado en un chip de la placa base que se encarga de realizar las funciones básicas de manejo y configuración del ordenador.

Bash (Bourne Again Shell): Interprete de comandos. Se encarga de interpretar las órdenes que se le den para su proceso por el kernel.

Bibliotecas de C: Conjunto de procedimientos y funciones (subprogramas) agrupadas en un archivo con el fin de ser aprovechadas por el lenguaje de programación.

Bachgman: Modelo de Datos de Red.

Backups: Copia de seguridad de una base de datos.

BUS: Facilita la transferencias internas de datos que se dan en un sistema computacional en funcionamiento.

debian-installer: Comando para instalar Debian GNU/Linux.

BOOTP (Bootstrap Protocol): Protocolo de red UDP utilizado por los clientes de red para obtener su dirección IP automáticamente. Normalmente se realiza en el proceso de arranque de los ordenadores o del sistema operativo.

Cardinalidad: Es el número de tuplas en una tabla.

Cobol (Common Business Oriented Language): Uno de los primeros lenguajes de programación de alto nivel, de raíces antiguas con instrucciones muy sencillas, pero que obligaban a los programadores de las primeras computadoras a que sus programas fuente se convirtieran en verdaderos libros debido a su longitud en líneas de código.

Clase: Descripción del conjunto de objetos que comparten los mismos atributos, funcionamiento, relaciones y semántica.

C++: Lenguaje de programación orientado a objetos, basado en el lenguaje C.

CPU (Central Processing Unit): Unidad donde se ejecutan las instrucciones de los programas y se controla el funcionamiento de los distintos componentes del ordenador.

C: Lenguaje de programación transportable que se puede usar para desarrollar software.

checksum: Forma de verificar y proteger la integridad de los datos al ser enviados.

cfdisk: Lee la tabla de particiones de las unidades de disco.

CD (Compact Disc): Disco Óptico de 12 cm. de diámetro para almacenamiento binario. Su capacidad "formateado" es de 660 Mb.

COM: Es la extensión que corresponde a un tipo de fichero ejecutable bajo Ms-Dos.

COMMIT: Mecanismo de sincronización y asíncrono, usado para escribir ciertas funciones.

CGI (Common Gateway Interface): Importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web.

CORBA (Common Object Request Broker Architecture): Estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

Cyrix: Era una corporación Norteamericana que manufacturaba microprocesadores compatibles con la tecnología Intel. Se encargó de generar una nueva línea de procesamiento 6x86 comparable con los chips de Intel.

DB: Conjunto de información guardada de forma correlativa y organizada de la cuál posteriormente se puede extraer uno o varios datos necesarios.

DB2: Sistema manejador de base de datos relacionales y funciona en varios sistemas operativos. Es un producto de IBM.

Disquetes: Dispositivo de almacenamiento de datos formado por una pieza circular de un material magnético que permite la grabación y lectura de datos, fino y flexible, encerrado en una carcasa cuadrada de plástico.

DHCP: Protocolo de configuración dinámica de servidores.

dBase: Sistema de Gestión de Base de Datos Relacional desarrollado por Ashton-Tate para ordenadores personales.
Dispositivos: Componentes necesarios que conforman las computadoras pala lograr su fin.

DVD: Formato de almacenamiento óptico, que puede ser usado para guardar datos, incluyendo películas con alta calidad de video y sonido.

Disco duro: Es un dispositivo de almacenamiento, que nació como evolución del disquete. Tiene una capacidad mucho mayor y es mucho más rápido, pero no está diseñado para ser llevado de un sitio a otro.

daemon: Clase especial de programa que corre en segundo plano en vez de ser controlado directamente por el usuario; cabe mencionar, que funciona sin tener relación con una terminal o consola y, consecuentemente, sin interactuar con un humano.

Debugger: Es un programa que asiste en la depuración de un programa.

Dominio: Es la representación de una empresa o entidad en el mundo de Internet. Se compone de un nombre seguido de una extensión que define el tipo de dominio.

EISA (Extended Industry Standard Architecture): Es una arquitectura de bus para ordenadores compatibles con las computadoras IBM.

Ensamblador (ASM): Lenguaje de programación de bajo nivel, muy cercano al código máquina. Su sintaxis depende por completo del tipo de ordenador que se esté usando.

Eric Steven Raymond: También conocido como ESR, es el autor de La Catedral y el Bazar. Es una de las figuras más conocidas del Movimiento del Software Abierto.

Expat: Esta orientado a XML 1.0 y sus librerías están escritos en C, incluidas en fuentes abiertas como Apache HTTP Server, Mozilla, Perl, Python y PHP.

Flash: Producto de software propiedad de Macromedia que permite crear animaciones vectoriales de gran calidad, con la ventaja de que ocupan muy poco espacio.

Free Software Foundation: Organización creada a partir del esfuerzo de Richard Stallman y otros entusiastas del software libre con el propósito de difundir este movimiento.

FreeBSD: Sistema operativo libre para ordenadores personales basado en CPU's de arquitectura Intel, incluyendo procesadores 386, 486, y Pentium. También son soportados por los procesadores AMD y Cyrix.

Forum: Oferta de un sitio que permite a los visitantes enviar y leer mensajes. Ideal para la animación de un sitio y la interactividad con los visitantes.
FTP (File Transfer Protocol): Protocolo de transferencia de archivos ideal para transferir datos por la red.

Firewall: Equipo de hardware o software utilizado en las redes para prevenir algunos tipos de comunicaciones prohibidos.

FORTRAN (Formula Translation): Lenguaje de programación desarrollado en los años 50 y activamente utilizado desde entonces.

GB: Unidad de medida de la capacidad de memoria y de dispositivos de almacenamiento informático.

Gcc (GNU Compiler Collection): Colección de compiladores, diseñados dentro del Proyecto GNU, funciona en muchas arquitecturas y sistemas operativos.

GNU FDL: Licencia copyleft para contenido libre, diseñada por la Fundación del Software Libre (FSF) para el proyecto GNU.

GIMP (GNU Image Manipulation Program): Programa de manipulación de imágenes del proyecto GNU. Se publica bajo GNU General Public License.

GNOME (GNU Network Object Model Environment): Parte del proyecto GNU y parte del movimiento de software libre de código abierto. GNOME es un sistema de escritorio que funciona en UNIX. La versión actual corre sobre Linux, FreeBSD, IRIX y Solaris.

GDB (GNU Debugger): Depurador de la GNU, publicado bajo GNU General Public License.

HP (Hewlett-Packard): Empresa dedicada a fabrica y comercializa hardware y servicios relacionados con la informática.

HTTP (Hyper Text Transfer Protocol): El hipertexto es el contenido de las páginas Web, y el protocolo de transferencia es el sistema mediante el cuál se envían las peticiones de acceder y respuesta de la página Web.

Host: Ordenador que permite a los usuarios comunicarse con otros sistemas centrales de una red.

Hostname: Nombre que se le da a una computadora.

Hz: Abreviatura de Hertzio, unidad de frecuencia en el Sistema Internacional, que equivale a un ciclo por segundo.

IDS (Integrated Data Store): Sistema de detección de intrusos. Busca anomalías que indiquen un riesgo potencial. Puede tomar medidas de protección.

Informix: Gestor de base de datos creado por Informix Software Inc. Incluye un sistema manejador de base de datos relacional basado en SQL.

Intel: Es una plataforma formada por un conjunto de componentes integrados que hace posibles modelos de uso concretos.

IBM (International Business Machines Corporation): Empresa dedicada a la fabricación y comercialización de hardware, software y servicios relacionados con la informática.

IMAP (Internet Message Access Protocol): Protocolo de red de acceso a mensajes electrónicos almacenados en un servidor

ISA (Industry Standard Architecture): Bus creado por IBM en 1980 en Boca Raton, Florida para ser empleado en los IBM PCs.

IDE (Integrated Drive Electronics): Electrónica de unidad integrada, que denota la interfaz estándar usada para conectar fundamentalmente unidades de disco y CD-ROM a un ordenador.

IP (Internet Protocol): Encargado de dar un direccionamiento a los datagramas de información y la administración del proceso de fragmentación de dichos datagramas.

Imagen ISO: Es un archivo que contiene la copia exacta de la imagen de un CD-ROM.

Java: Lenguaje de programación orientado a objeto parecido al C++. Usado en WWW para la telecarga y telejecución de programas en el ordenador cliente.

Jabber: Es un protocolo libre basado en el estándar XML para mensajería instantánea.

Kernel: También conocido como núcleo, es la parte fundamental de un sistema operativo.

KDE (K Desktop Environment): Entorno de escritorio gráfico e infraestructura de desarrollo para sistemas Unix y en particular Linux.

LAN (Redes de área local): Son de cobertura pequeña, velocidades de transmisión muy elevadas, utilizan redes de difusión en vez de conmutación, no hay nodos intermedios.

LaTeX: Es un conjunto de macros de TeX, escritas por Leslie Lamport (LamportTeX) en 1984, con la intención de facilitar el uso del lenguaje creado por Donald Knuth, (TeX), al cuál no modifica sino complementa.

Login: Acción de introducir el nombre a través del teclado para acceder a otro ordenador.

LDAP (Lighweight Directory Access Protocol): Servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

LILO (LInux LOader): Cargador de Linux. Es un programa que se instala en el MBR del disco duro y que permite elegir entre distintos sistemas operativos que se encuentren instalados en distintas particiones del disco duro.

Linux: Es un sistema operativo y un núcleo. Es uno de los paradigmas del desarrollo de software libre, donde el código fuente está disponible públicamente y cualquier persona puede libremente usarlo, modificarlo y/o redistribuirlo.

Multiprocesador: Sistemas que utilizan más de un procesador para acelerar los procesos ya que se ejecutan en paralelo.

Mozilla: Navegador Web y una plataforma de desarrollo libre de código abierto para la WWW. Habitualmente se le llama navegador.

ML: Lenguaje de programación de propósito general de la familia de los lenguajes de programación funcional desarrollado por Robin Milner.

Minix: Es un clon del sistema operativo Unix distribuido junto con su código fuente y desarrollado por el profesor Andrew S. Tanenbaum en 1987.

Microsoft Internet Information Server: Servicios de Internet Información Server (IIS) simplifica la publicación de información en Internet o en la Intranet.

MB (Megabyte): Unidad de medida de cantidad de datos informáticos. Es un múltiplo binario del byte, que equivale a 220 (1 048 576) bytes.

Mandrake: Distribución Linux enfocada a principiantes o usuarios medios, propiedad de Mandrakesoft.

Módulos: Módulos partes de un sistema, de computador responsable de una tarea bien definida.

Microchannel: Arquitectura estándar de placas base, basada en ISA, con ranuras de ampliación de 32 bits, desarrollada por IBM y hoy en día abandonada.

MD5 (Message-Digest Algorithm 5): Algoritmo de reducción criptográfico de 128 bits ampliamente usado. El código MD5 fue diseñado por Ronald Rivest en 1991.

Mascara de red: Una máscara de red es un conjunto de cuatro números separados por puntos.
MAGIC: Es un de 4GL multiplataforma que no posee código sino que está orientado a tablas y eventos.

MyODBC: Controlador para el acceso a bases de datos usado por el gestor de bases de datos MySQL.

MySQL: MySql es un gestor de Bases de Datos multiusuario que gestiona bases de datos relacionales poniendo las tablas en ficheros diferenciados.

Netscape: Software encargado de optimizar las páginas Web para un explorado especifico.

NCSA: Servicio de Investigación ubicado en la Universidad de Illinois donde fue desarrollado el navegador Mosaic. Origen de la mayor parte de documentos de dominio público en Internet.

NNTP (Network News Transport Protocol): Protocolo de transferencia de noticias. Es el Protocolo de red utilizado por el Usenet internet service. Es un Protocolo de red basado en tiras de textos enviados sobre canales TCP de 7 bit ASCII.

Nokia: Empresa dedicada al desarrollo de telefonía celular.

NFS (Network File System): Sistema de archivos distribuido para un entorno de red de área local. Posibilita que distintos sistemas conectados a una misma red accedan a ficheros remotos como si se tratara de locales.

ODBC (Open DataBase Connectivity): Estándar de acceso a bases de datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible el acceder a cualquier dato de cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos almacene los datos.

Oracle: Sistema manejador de base de datos relacional, fabricado por Oracle Corporation.

OpenOffice: Suite ofimática de software libre y código abierto que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos..

OpenLDAP (open source implementation of the Lightweight Directory Access Protocol): Aplicación de código abierto, incluido en las distribuciones Linux.

OpenSSL: Aplicación de open source de los protocolos SSL y TLS. Sus bibliotecas están escritas en C.

Open Source Initiative (OSI): Organización dedicada a la promoción del software abierto. Fue fundada en febrero de 1998 por Bruce Perens y Eric S. Raymond.

OS2 (Operating System 2): Sistema operativo desarrollado por IBM. Es multitarea, con un entorno gráfico, 32 bits.

Paradox: Base de datos relacional para entorno MS Windows, anteriormente disponible para MS-DOS y Linux.

Particionamiento: Palabra usada para describir la división de cierta cosa u objeto.

Perl (Practical Extraction and Report Language): Lenguaje de programación desarrollado por Larry Wall, inspirado en herramientas de UNIX.

PervasiveSQL: Motor de base de datos embebible que sustenta la integridad de los datos, el alto rendimiento, flexibilidad, escalabilidad y un bajo costo de propiedad.

PDF (Portable Document Format): Formato para el almacenamiento de documentos, desarrollado por la empresa Adobe Systems.

PHP (Hypertext Preprocessor): Lenguaje de programación interpretado, se utiliza para la programación de páginas Web activas, y se destaca por su capacidad de mezclarse con el código HTML.

PHP/FI: Lenguaje de programación, que permite el acceso a MySQL desde una página HTML.

PCMCIA: Dispositivo normalmente utilizado en computadoras portátiles para expandir las capacidades de este.

Plug and Play (Enchufar y usar): Tecnología que logra que las PC compatibles sean fáciles de configurar y mantener.

PCI (Peripheral Component Interconnect): Es un bus de computadora estándar para conectar dispositivos periféricos a la tarjeta madre de la computadora.

Python: Lenguaje de programación interpretado e interactivo, capaz de ejecutarse en una gran cantidad de plataformas. Fue creado por Guido van Rossum en 1990.

Protocolos: Conjunto de reglas que controlan la secuencia de mensajes que ocurren durante una comunicación entre entidades que forman una red.

Programa procedural: Es un paradigma de la programación basado en el concepto de modularidad, usado para la codificación de algún programa.

POP3 (Post Office Protocol 3): Tercera versión del protocolo diseñado para la gestión, el acceso y la transferencia de mensajes de correo electrónico entre dos máquinas.

PC (Personal Computer): Término para identificar una computadora personal.
PostgreSQL: Es un sistema manejador de base de datos relacional de tipo open source.

Root: Nombre del súper usuario de un sistema determinado, el cuál pude ver, manipular y eliminar archivos de cualquier directorio del sistema.

ReiserFS: Sistema de archivos de propósitos generales, diseñado e implementado por un equipo liderado por Hans Reiser. Actualmente funciona bajo Linux.

Red Hat: Compañía responsable de la creación y mantenimiento de la distribución del sistema operativo Linux que lleva el mismo nombre de Red Hat Linux.

Red: Conjunto de nodos interconectados entre si.

Regla de minimalidad: Una relación o fragmento es dividido por lo menos en dos partes las cuales se acceden de distinta forma por al menos una consulta de usuario.

RAM (Random Access Memory): Se usa para mantener los programas mientras se están ejecutando, y los datos mientras se procesa

safe_mysqld: Aplicación de MySQL encargado de verificar si mysqld se encuentra en ejecución y si no es así inicia el daemon.

Samba: Programa de ordenador que imita el protocolo de archivos compartidos de Microsoft Windows para sistemas de tipo UNIX, para que los ordenadores con Linux o Mac OS X se vean como servidores. Samba es software libre.

Simula: Primer lenguaje de programación orientado a objetos, el cuál fue planteado para la simulación de actividades.

Sendmail: Agente de transporte de correo (MTA - Mail Transport Agent) en Internet, cuya tarea consiste en "encaminar" los mensajes de correos de forma que estos lleguen a su destino.

SGBD: Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan.

SQL (Structured Query Language): Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas.

Sun: Compañía de Software y Hardware popular por sus estaciones de trabajo y últimamente por haber creado Java.

Servidor FTP: Ordenador que sirve cualquier tipo de fichero, a través del Protocolo de Transferencia de Ficheros a clientes FTP o a navegadores Web que lo soporten.
SMP (symetric multi-processing): Se trata de un tipo de arquitectura de ordenadores que se encuentra en las computadoras personales y servidores de baja gama.

SNMP (Simple Network Management Protocol): Protocolo simple de gestión de redes, es aquél que permite la gestión remota de dispositivos de red, tales como switches, routers y servidores.

SCSI (Small Computer System Interface): Interfaz estándar para la transferencia de datos entre periféricos en el bus de una computadora.

SuSE: Una Distribución GNU/Linux, más sencillas de instalar y administrar, ya que cuenta con varios asistentes gráficos para completar diversas tareas.

Servidor DHCP: Provee los parámetros de configuración a las computadoras conectadas a la red informática que los requieran (máscara, puerta de enlace y otros) y también incluye un mecanismo de asignación de direcciones de IP.

Servidores BOOTP: Encargado de asignan la dirección IP de un conjunto de direcciones a cada cliente con un cierto tiempo de uso (lease time). Originalmente está definido en el RFC 951.

Servidor DNS: Se encarga de traducir nombres de sistema (como por ejemplo www.yahoo.com) en sus correspondientes direcciones IP. Para poder navegar por Internet, es preciso que tengas acceso a un servidor DNS.

Script: Serie secuencial de instrucciones que permite realizar tareas sencillas y repetitivas, generalmente son interpretadas en tiempo de ejecución, aunque hay sistemas que permiten compilar los guiones.

SSH: Es un paquete de utilidades para poder conectarse y ejecutar programas en una máquina remota.

SSL (Secure Sockets Layer): Es un protocolo diseñado por la empresa Netscape Communications, que permite cifrar la conexión, incluso garantiza la autentificación. Se basa en la criptografía asimétrica y en el concepto de los certificados.

Telnet: Es un protocolo para que dos computadoras separadas se puedan conectar y trabajar una con otra, como si estuvieran conectadas directamente. Una de ellas es el usuario y la otra el servidor.

Tcl (Tool Command Language): Es un lenguaje de script creado por John Ousterhout, de fácil aprendizaje y potente. Se usa principalmente en programas rápidos, aplicaciones "script", entornos gráficos y pruebas.

Tuplas: Es una hilera o fila en una tabla.

Tasksel: Es el Instalador de Tareas de Debian que se ofrece durante la instalación. Ofreces ciertas configuraciones predefinidas en función a tareas, como entorno de escritorio, desarrollo en C, servidor de ficheros, entre otros.

Unix: Sistema operativo portable, multitarea y multiusuario; desarrollado en principio por un grupo de empleados de los laboratorios Bell de AT&T

VESA Local Bus: Bus definido por Video Electronics Standards Association.

VGA (Video Graphics Array): Estándar de presentación de video de IBM, que se originó con sus modelos PS/2. VGA se ha convertido en el estándar mínimo de presentación para computadores personales.

VMS (Virtual Memory System): Sistema de Memoria Virtual es un método de administración de la memoria que ejecuta el Sistema Operativo, que consiste en desocupar localidades en el módulo RAM para escribirlos en un archivo dinámico en el disco duro.

WDDX (Web Distributed Data eXchange): Mecanismo que facilita el intercambio de datos entre computadoras.

Web: Se utiliza para denominar uno de los servicios más importantes de la red Internet. Son páginas que utilizan un lenguaje especial llamado HTML, que permite presentar en pantalla texto y gráficos en el formato deseado.

Win32: Conjunto de API´s para la Interfaz de programación de aplicaciones, disponibles para los sistemas operativos Microsoft Windows

XHTML (eXtensible Hyper Text Markup Language): Lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas Web.

XML (eXtensible Markup Language): Lenguaje de descripción de páginas de Internet diseñado con la intención de reemplazar al estándar actual HTML.

XITAMI: Es un software Web de código abierto.

X Window: Sistema desarrollado a mediados de los años 80 en el MIT para dotar de una interfaz gráfica a los sistemas Unix.

XFree86: Producto derivado de The XFree86 Project, Inc. el cuál es una implementación libre de X Window System. Corre en sistemas UNIX, BSD, Sun Solaris, SGI IRIX, Linux, OS/2 y Cygwin (para Windows).

Zope: Servidor de aplicaciones Web escrito en el lenguaje de programación Python. Usa una interfaz de usuario basada en páginas Web.

Referencias

Referencias bibliográficas

[A] Adoración de Miguel Castaño, Mario G. Piattini Velthuis, *"Fundamentos y Modelos de Bases de datos",* Editorial Alfaomega, Segunda edición, 1999.

[B] Adoración de Miguel Castaño, Mario Gerardo Piattini Velthuis, *"Concepción y diseño de base de datos del modelo E/R al modelo relacional"*, Editorial Addison Wesley Iberoamericana RA-MA, 1993.

[C] Jeffrey D. Ullman, *"Principles of database and knowledge-base systems"*, Stanford University, Computer Science Press, Volumen 1: Classical database systems, 1988.

[D] Henry F. Korth, Abraham Silberschatz, *"Fundamentos de Bases de Datos"*, Editorial Mac Graw Hill, Segunda Edición, 1993.

[E] C. J. Date, *"Introducción a los sistemas de bases de datos"*, Editorial Prentice Hall, Séptima edición, 2001.

[F] M. Tamer Özsu, Patrick Valduriez, *"Principles of distributed database systems"*, Editorial Prentice Hall, Second edition, 1991.

[G] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, *"Fundamentos de Bases de Datos"*, Editorial Mc Graw Hill, Cuarta Edición, 2002.

[H] Kenneth E. Kendall, Julie E. Kendall, *"Análisis y diseño de sistemas"*, Editorial Prentice Hall Hispanoamericana, Tercera edición.

[I] Alfredo Catalina Gallego/Miguel Catalina Gallego, *"UNIX/Linux"*, Editorial Mc Graw Hill, Segunda edición, 2004.

[J] Francisco García Jiménez, *"Guía de campo Linux"*, Editorial Alfaomega, 2005.

[K] Jack Tackett Jr, Steven Burnett, *"Edición especial Linux"*, Editorial Prentice Hall, Cuarta edición, 2000.

[L] Abraham Gutiérrez Rodríguez, Ginés Bravo García, *"PHP4 a través de ejemplos"*, Editorial Alfaomega RA-MA, 2004.

[M] Larry Ullman, *"Guía de aprendizaje MySQL"*, Abarca versiones 3 y 4, Editorial Prentice Hall, 2004.

[N] Por Janet Valade, *"PHP 5 para dummies"*, Publicado por ST Editorial, 2004.

[O] C. J. Date, *"Introducción a los sistemas de bases de datos",* Volumen 1, Editorial Addison wesley Iberoamericana, Quinta edición.

[P] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, *"Fundamento de base de datos"*, Editorial Mc Graw Hill, Tercera edición, 1998.

Referencias electrónicas

[1] Cuestionario.
URL: http://www.angelfire.com/empire2/bashvictorpage/

[2] Las bases de datos en salud como fuente principal para los servicios de BINASSS.
URL: http://www.una.ac.cr/bibl/v10n2/art6.html

[3] Base de datos.
URL: http://members.fortunecity.es/siddartha/master1.htm

[4] Ensayos, Ventajas y retos en el uso de bases de datos distribuidas, Francisco de Asís López Fuentes, Septiembre-Diciembre del 2002.
URL: http://mixteco.utm.mx/temas-docs/ensayo1t18.pdf

[5] Diseño de bases de datos distribuidas.
URL: http://www.cs.cinvestav.mx/SC/prof_personal/adiaz/Disdb/Cap_3.html

[6] Bases de Datos Distribuidas, Diseño en las BDD.
URL: http://ji.ehu.es/Illarramendi/apuntesabd/BDD2004.PDF

[7] Universidad de castilla-la mancha, escuela superior de informática, Base de datos distribuidas, Distribución 1, 3 de mayo del 2006.
URL: http://alarcos.inf-cr.uclm.es/doc/bbddavanzadas/DISTRIBUCION1.pdf

[8] Tema 9. Bases de Datos Distribuidas (BDD), Bases de Datos distribuidas y arquitectura, cliente-servidor Elmasri/Navathe 02.
URL: http://www.unirioja.es/cu/arjaime/Temas/09.Distribuidas.pdf

[9] Distribución de bases de datos.
URL: http://www.cs.cinvestav.mx/SC/prof_personal/adiaz/Disdb/Cap_1.html

[10] Manejador de Bases de Datos.
URL: http://www.itlp.edu.mx/publica/tutoriales/basedat1/tema1_9.htm

[11] Web, Base de datos DBMS.
URL: http://www.unalmed.edu.co/~mstabare/Dbms.htm

[12] Modelos de datos, conceptos y clasificación.
URL: http://alarcos.inf-cr.uclm.es/doc/bda/doc/trab/T9900_FLorenzo.pdf

[13] Apuntes de ficheros y bases de datos, Introducción, Ventajas e inconvenientes de los sistemas de bases de datos, Mercedes Marqués.
URL: http://www3.uji.es/~mmarques/f47/apun/node7.html

[14] Apuntes de ficheros y bases de datos, sistemas de bases de datos, funciones de los sistemas de gestión de bases de datos.
URL: http://www3.uji.es/~mmarques/f47/apun/node39.html

[15] Métodos de optimización de consultas para el lenguaje SQL, Sistema de gestión de bases de datos relacionales, Tipos de SGBD, Mario Cisterna Neira, Santiago-Chile, 2002.
URL: http://macine.epublish.cl/tesis/index-3_3_.html

[16] Instituto Tecnológico de Informática, Línea de investigación (I + D + I), Sistemas distribuidos.
URL: http://www.iti.upv.es/about_iti/lines/sidi_bd

[17] Sistemas distribuidos.
URL: http://gsyc.escet.urjc.es/simple_com/phd-thesis-es/node11.html

[18] Redes, Sistemas Distribuidos y Paralelismo.
URL: http://www.dcc.uchile.cl/web/article-27565.html

[19] Historia del Proyecto GNU.
URL: http://www.gnu.org/gnu/gnu-history.es.html

[20] La Definición de Software Libre.
URL: http://www.gnu.org/philosophy/free-sw.es.html

[21] Utilización de software libre como única tecnología para el desarrollo de portales Web, Daniel Gayo Avello, Benjamín López Pérez, Luís Vinuesa Martínez, José E. Labra Gayo. M. Cueva Lovelle, Universidad de Oviedo, Departamento de Informática, Oviedo, España, 33007, 2005.
URL: http://www.di.uniovi.es/~dani/publications/sisoft2001.PDF

[22] ¿Qué es el software libre?.
URL: http://gugs.sindominio.net/faqs/faqlinux.html

[23] ¿Qué es Software Libre?.
URL: http://www.molinux.info/index.php?option=com_content&task=view&id=22&Itemid=46

[24] Diversas licencias y comentarios sobre ellas.
URL: http://www.gnu.org/licenses/license-list.es.html#GPLCompatibleLicenses

[25] "La Historia del Proyecto Debían en una tabla", Javier Viñuales.
URL: http://www.debian.org/international/spanish/contrib/tabla-historia.txt

[26] El Proyecto Debían: algo más que una distribución de GNU/Linux, Por Miquel Vidal.
URL: http://espora.org/biblioweb/proyecto-debian.html

[27] Sobre Debían.
URL: http://gsyc.escet.urjc.es/~grex/robles-barahona-hispalinux2003.pdf

[28] ¿Qué es Debían?
URL: http://es.tldp.org/DEBIAN/%257Ejfs/debian/doc/es/debian-guide-es/html/node10.html

[29] Contrato Social de Debían.
URL: http://www.debian.org/social_contract

[30] Contrato social de Debían, Las directrices de software libre de Debían (DFSG).
URL: http://www.debian.org/social_contract#guidelines.

[31] Debían Policy, Manual, About this manual.
URL: http://www.debian.org/doc/debian-policy/

[32] El Linux Standard Base.
URL: http://www.linuxbase.org/

[33] El Filesystem Hierarchy Standard.
URL: http://www.pathname.com/fhs/

[34] Proyecto, Debían Jr.
URL: http://www.debian.org/devel/debian-jr/

[35] Linux al fin.
URL: http://www.tucuman.linux.org.ar/textos/historia.html

[36] Tux.
URL: http://www.isc.tamu.edu/~lewing/linux

[37] Instalación de Debían GNU/Linux 3.0 para Intel x86, Capitulo 1-Bienvenido a Debían,
URL: http://www.debian.org/releases/woody/i386/ch-welcome.es.html

[38] Manual del usuario de Debían GNU/Linux 2.2 “Potato”, Guía, bienvenido a Debían, ¿Qué es Debían GNU/Linux?, 2005.
URL: http://es.tldp.org/DEBIAN/%257Ejfs/debian/doc/es/debian-guide-es/html/node12.html

[39] Conseguir Debían.
URL: http://www.es.debian.org/distrib/

[40] El sistema operativo GNU/Linux, Licencias, ¿Qué es el copyleft?
URL: http://www.gnu.org/licenses/licenses.es.html#WhatIsCopyleft

[41] Instalación de Debian GNU/Linux 3.0 para Intel x86, Capítulo 2 - Requisitos de Sistema.
URL: http://www.debian.org/releases/woody/i386/ch-hardware-req.es.html

[42] Guía de instalación de Debían GNU/Linux, Requerimientos de sistema, Requisitos de memoria y espacio en disco.
URL: http://www.es.debian.org/releases/stable/i386/ch02s05.html.es

[43] Guía de instalación de Debían GNU/Linux, ¡Haga copias de seguridad de su información actual!.
URL: http://www.es.debian.org/releases/stable/ia64/install.pdf.es

[44] Guía de instalación de Debían GNU/Linux**,** Utilización del menú de configuración de la BIOS.
URL: http://www.es.debian.org/releases/stable/i386/install.pdf.es

[45] Instalación de Debian, Preparando la instalación.
URL: http://www.nautopia.net/archives/es/linux_distribuciones/instalacion_debian/instalacion_de_debian.php

[46] Instalación de debian, Preparando la instalación.
URL: http://www.nautopia.net/archives/es/linux_distribuciones/instalacion_debian/debian_2.php

[47] Instalación de debian, Preparando la instalación.
URL: http://www.nautopia.net/archives/es/linux_distribuciones/instalacion_debian/debian_3.php

[48] Desarrollo de aplicaciones web, El nacimiento de Apache.
URL: http://www.uoc.edu/masters/esp/img/873.pdf

[49] Tutórales, Apache, ¿Qué es apache?
URL: http://www.servidores.unam.mx/tutoriales/apache.html

[50] Servidor "libre" de http, Características.
URL: http://es.tldp.org/Presentaciones/200102linuxcol/linux-serv-internet/linux-serv - internet/apache-10.html

[51] Página principal de Apache.
URL: http://httpd.apache.org/

[52] Página para descargar Apache.
URL: http://httpd.apache.org/download.cgi

[53] Documentación del servidor HTTP Apache 2.0, Manual de referencia, Compilación e Instalación.
URL: http://httpd.apache.org/docs/2.0/es/install.html

[54] Configuración de Apache.
URL: http://geneura.ugr.es/~gustavo/apache/

[55] Carles Mateu, Software libre, Desarrollo de aplicaciones web, Configuración de Apache.
URL: http://www.uoc.edu/masters/esp/img/873.pdf

[56] Documentación del servidor HTTP Apache 2.0, Manual de referencia, Módulos, Índice de Módulos, Funcionalidad básica de Apache.
URL: http://httpd.apache.org/docs/2.0/es/mod/core.html

[57] Documentación del servidor HTTP Apache 2.0, Guía de usuario, Sección de configuración.
URL: http://httpd.apache.org/docs/2.0/es/sections.html

[58] Documentación del Servidor HTTP Apache 2.0, Manual de Referencia, Iniciar Apache.
URL: http://httpd.apache.org/docs/2.0/es/invoking.html

[59] Documentación del Servidor HTTP Apache 2.0, Manual de Referencia, Parar y reiniciar Apache.
URL: http://httpd.apache.org/docs/2.0/es/stopping.html

[60] Manual de referencia de MySQL 5.0.
URL: http://downloads.mysql.com/docs/refman-5.0-es.a4.pdf

[61] Ventajas de MySQL.
URL: http://www.elguruprogramador.com.ar/foros/mensaje.asp?id=9330

[62] PostGreSQL vs. MySQL, MySQL, Daniel Pecos.
URL: http://www.netpecos.org/docs/mysql_postgres/x57.html

[63] Tipos de datos en MySQL, Artículo publicado el 2005-10-21 00:55:20, Por Larry Ullman.
URL: http://www.dise-web.com/articulos/sql/tipos-de-datos-mysql/?v=1

[64] Página principal de MySQL.
URL: http://www.mysql.com/

[65] Liga *Whitepaper - MySQL Performance Benchmarks*.
URL: http://www.mysql.com/why-mysql/

[66] Página para hacer el llenado de requisitos para descargar MySQL.
URL: http://www.mysql.com/why-mysql/white-papers/performance.php

[67] Página para seleccionar la plataforma adecuada para instalar MySQL.
URL: http://dev.mysql.com/downloads/mysql/5.0.html

[68] Ventajas de PHP.
URL: http://ascii.eii.us.es/docs/2002-03/php/php4.html

[69] Manual de PHP.
URL: http://www.php.net/manual/es/

[70] Página principal de PHP.
URL: http://www.php.net/

[71] Página de la liga Download (descargar) PHP.
URL: http://www.php.net/downloads.php

[72] Selección del servidor para descargar PHP.
URL: http://mx2.php.net/get/php-5.1.4.tar.bz2/from/a/mirror

[73] Índice códigos de programación (códigos PHP)
URL: http://www.webnova.com.ar/codigos-fuentes.php?recurso=8