

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE INFORMACIÓN PARA LA ASIGNACIÓN DE TUTORES EN EL ÁREA DE TUTORÍA PARA TODOS

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTAN:

JIMENA ARREGUÍN REYES
ALVARO ILICH PEDROZO LARA



DIRECTOR DE TESIS:

M. en I. CÉSAR ENRIQUE BENÍTEZ JOYNER

Cd. UNIVERSITARIA, D.F., FEBRERO 2006

AGRADECIMIENTOS.

A Mis Padres

Gracias por haberme proporcionado su apoyo, comprensión y todo su cariño durante el largo camino de mi preparación profesional, por enseñarme a superar los obstáculos que surgen en la vida, por alentarme a seguir a pesar de los momentos difíciles y a levantarme de los tropiezos que tenga en la vida con serenidad y seguridad, todo esto se ve reflejado con la realización de esta tesis, a cual se las dedico con mucho cariño y agradecimiento.

A Mis Hermanos

Gracias por brindarme la oportunidad de enseñarles que con esfuerzo y dedicación se logran los más grandes propósitos en la vida.

A Mi Novio

A Juan Carlos te agradezco los momentos de calma que me brindaste, tus palabras de aliento y la serenidad con la cual me escuchaban y todo el apoyo que me brindaste durante la realización de esta tesis.

A Mis Profesores

A cada uno de mis profesores por haberme transmitido el conocimiento necesario para desarrollarme y ser todo una persona profesional en cada una de las actividades que realice en la vida, por darme un consejo cuando era necesario para mejorar en mi desempeño académico.

A la Universidad

Por haberme abierto las puertas a un nuevo horizonte que me permitió ser parte importante de ella, proporcionándome todas las oportunidades para ser un digno representante de la Facultad de Ingeniería.

Al M. en I. CÉSAR ENRIQUE BENÍTEZ JOYNER,
Director de Tesis.

A la persona que me brindo la oportunidad de demostrar que una persona responsable, llega a tener éxito en la vida y en cada uno de los proyectos que se proponga uno siempre tener coraje y decisión para obtener el mejor resultado.

Gracias a Dios por acompañarme en mi largo camino hacia la culminación de esta Tesis, la cual se realizó con esfuerzo y cariño.

Jimena Arreguín Reyes.
Febrero del 2006.

AGRADECIMIENTOS

Es tiempo de agradecer a todas aquellas personas que para bien y para mal han formado parte de mi realidad, han contribuido a formar poco a poco a este ser humano.

También es tiempo de hacer un homenaje a mis padres que siempre confiaron en mí, que me han dado su ejemplo de superación y lo más importante su amor. Gracias Padres por la educación recibida por enseñarme las cosas que verdaderamente tienen valor en esta vida y por evolucionar conmigo.

Les doy las gracias a mis hermanas que siempre estuvieron ahí cuando las necesitaba y que siempre me han dado su cariño y alegría.

A mis tíos y primos les agradezco la inspiración, la fraternidad que tenemos y todos los momentos que hemos compartido.

Alejandro y Mariana gracias por ser incondicionales, por apoyarme siempre y enriquecerme con sus concejos e ideas.

A mis amigos de toda la vida, Guillermo y Jonathan, que aunque pase tiempo sin verlos, todo sigue siendo como cuando niños.

Un especial agradecimiento a mis amigos de la Facultad de Ingeniería, a Faustino, Juan Carlos, Hugo, Onesimo, por librar todas esas batallas donde hubo momentos muy difíciles y salimos adelante juntos.

Gracias al M. en I. César Enrique Benítez Joyner por habernos guiado en el trabajo de tesis, darnos ánimo y buenos concejos.

Muchas gracias a la Universidad Nacional Autónoma de México por haberme brindado la mejor educación que se imparte en América latina y por haberme formado con una visión crítica y humana.

El camino fue difícil, pero sin el no sería lo mismo este momento.

Alvaro Ilich Pedrozo Lara
Febrero 2006

Índice

Introducción 1

Capítulo 1. Definición del problema 4

1.1 Historia de “Tutoría para Todos” 5

1.2 Misión 5

1.3 Objetivos 6

1.4 Etapas 6

1.5 Definición del problema 7

1.6 Justificación del problema 7

Capítulo 2. Fundamentos teóricos 8

2.1 Concepto de una base de datos 9

2.2 Metodología para una base de datos 14

2.3 Definición de un Sistema Manejador de una Base de datos (DBMS) 15

2.3.1 Tipos de DBMS 15

2.3.2 Características de un DBMS 18

2.4 Modelado de una base de datos 21

2.4.1 Modelos lógicos basados en objetos 21

2.4.2 Modelos lógicos basados en registros 22

2.4.3 Modelo de Red 22

2.4.4 Modelo Jerárquico 23

2.4.5 Modelos Físicos de datos 23

2.5 Modelo Entidad – Relación 23

2.6 Reglas de normalización 26

2.6.1 Grados de normalización 27

2.6.2 Primera Forma Normal 28

2.6.3 Segunda Forma Normal 28

2.6.4 Tercera Forma Normal 28

2.7 Diccionario de datos 29

2.8 Diagramas de flujos de datos 30

2.8.1 Flujo de datos 31

2.8.2 Proceso 31

2.8.3 Nombre del proceso 32

2.8.4 Terminadores (fuentes o destinos de los datos) 32

2.9 Arquitectura cliente / servidor 33

2.9.1 Manejadores de base de datos 35

2.9.2 PostgreSQL 35

2.9.3 Oracle 38

2.9.4 Access 38

2.9.5 MS SQL Server 39

2.9.6 DB2 39

2.9.7 MySQL 39

Capítulo 3. Herramientas y Lenguajes de Programación 41

3.1 Servidor Apache 42

3.1.1 Arquitectura de Apache 2.0 43

3.1.2 Portabilidad en tiempo de ejecución 46

3.1.3 Requisitos del sistema 46

3.2 Manejadores de Base de Datos de Libre Distribución (MySql) vs. Licenciados 46

3.2.1 Oracle 46

3.2.2 MYSQL 48

3.2 .3 Lenguajes de Programación de Libre Distribución (PHP) vs. Licenciados 49

3.3 Migración a Linux 55

3.4 Instalación de Apache, PHP y MySql en Windows 57

3.4.1 Instalación del Servidor Apache 57

3.4.2 Instalación del lenguaje de programación PHP 59

3.4.3 Instalación del manejador de bases de datos MySql 61

3.4.4 Instalación de Apache, PHP y MySql en Linux 62

Capítulo 4. Propuesta de solución 65

4.1 Metodología utilizada 66

4.1.2 Determinando grado de rigor y tipo de proyecto 68

4.1.3 Regiones de tareas 72

4.2 Estudio de tiempos 74

4.3 Modelado del sistema 77

4.3.1 Modelo de objetos 77

4.3.2 Modelo dinámico 79

4.3.3 Modelo funcional 81

Capítulo 5. Diseño de la aplicación 82

5.1 Diseño de la aplicación 83

5.2 Diagrama de flujo de datos 87

5.3 Diseño de la interfaz 88

5.4 Diseño de la base de datos 91

5.5 Diseño de la base de datos conceptual 91

5.6 Modelo entidad-relación 93

Capítulo 6. Programación del sistema 99

6.1 Programación del sistema 100

6.2 Programación de la interfaz 100

6.3 Programación de la base de datos y sus tablas 108

6.4 Programación de las consultas 114

Capítulo 7. Pruebas, análisis de resultados y conclusiones 143

7.1 Pruebas y análisis de resultados 144

7.2 Conclusiones 154

Bibliografía 155

INTRODUCCIÓN

INTRODUCCIÓN

La realización de la presente tesis titulada **S**istema de **I**nformación para la **A**signación de tutores en el Área de **T**utoría para **T**odos (**SIATT**), tiene como objetivo principal el realizar el manejo de la información.

En los siguientes capítulos se expondrá el razonamiento en el que se llevó a cabo la elaboración del sistema mediante un análisis y el planteamiento, tanto de los requerimientos como de las necesidades de la Coordinación de Atención Diferenciada para Alumnos (COPADI) en el Programa de **T**utoría para **T**odos, con ello se determina la interfaz y de los recursos para dar justificación del sistema.

En el **Capítulo 1** se conocerá el funcionamiento y el objetivo del Programa de **T**utoría para **T**odos y la problemática en el que se encuentra dicho programa, así como, cumplir con el objetivo.

En el **Capítulo 2** se conocerán las bases teóricas de las herramientas a utilizar, con respecto a la base de datos y analizar a las diferentes metodologías aplicadas en el desarrollo de un sistema informático y se verán las diferencias más importantes de los **M**anejadores de **B**ase de **D**atos (DBMS)

En el **Capítulo 3** se tendrá la perspectiva con respecto a la utilización del software bajo licencias contra aquel que no requiere licencia, comúnmente llamado **Software libre.**

En el **Capítulo 4** se dará conocer las metodologías a utilizar para el desarrollo del sistema (**SIATT),** así como los **tiempos empleados y el modelado del mismo.**

Esto permitirá llevar un mejor control y manejo de las herramientas ha emplear y aprovechar al máximo los recursos para cumplir con los requerimientos propios del sistema

En el **Capítulo 5** se realizaron los diseños con respecto al flujo de datos de acuerdo con el diseño de la base de datos, del modelo a utilizar, y también el diseño de la interfaz gráfica; esto se realizó para optimizar recursos y tiempo de desarrollo, considerando sea escalable el sistema **(SIATT).**

En el **Capítulo 6** se realizó la programación de las diferentes etapas, así como de las distintas consultas requeridas por el sistema **(SIATT)**, a fin de obtener un mejor rendimiento y aprovechamiento del lenguaje de programación.

En el **Capítulo 7** se realizarón las pruebas necesarias y el análisis de los resultados durante el desarrollo del sistema (**SIATT),** dando solución al problema planteado, de acuerdo con el objetivo del trabajo.

En el **Capítulo 8** se plantean las conclusiones a las que se llegó por el desarrollo del sistema y desempeño del mismo **(SIATT).**

CAPÍTULO 1

Definición del Problema

1.1 Historia de "Tutoría para Todos"

La Facultad de Ingeniería, presentaba una problemática en el desarrollo de los semestres curriculares que fueron el gran índice de deserción y un alto índice de reprobación; esto debido a que el estudiante no tenía un apoyo académico para un mejor entendimiento de aquellos temas de las diversas asignaturas correspondientes a las ciencas basicas en donde presentaba problemas al momento de la evaluación, por ese motivo se da origen al Programa **Tutoría para Todos. E**ste programa tiene sus orígenes en la década de los 80's, pero es hasta la época actual en donde se viene consolidando como una realidad.

Con el proposito de disminuir los niveles de reprobación y deserción escolar y de aumentar los niveles de aprovechamiento y de eficiencia Terminal, el programa **Tutoría para Todos** pretende ser un proceso educativo (Tutor-alumno) que busca mejorar el rendimiento y desarrollo académico, solucionar problemas escolares, desarrollar habitos de estudio, trabajo, reflexion y conviviencia social.

Pretende orientar de manera general a los alumnos en tono a las publicaciones academicas y/o funcionales que se presenten durante el ciclo escolar.

La Coordinación de Programas de Atención Diferenciada para Alumnos (COPADI) continuo apoyando a los estudiantes con diversos programas para alentar su desempeño escolar.

1.2 Misión

El Programa de Tutoría para Todos presenta el compromiso docente de los profesores de carrera y como un derecho que tienen los estudiantes que acceden a ella para estudiar una licenciatura en Ingeniería.

1.3 Objetivos

Los principales objetivos que presenta el Programa son:

- Establecer y consolidar un sistema de tutoría en el que el tutor considere al estudiante en lo individual.

- Lograr que la tutoría sea parte de la tarea propia de todo profesor de carrera y derecho de todos los alumnos.

- Proporcionar a los estudiantes orientaciones diversas en lo que se refiere a:

    a. Orientación para reafirmar la elección de carrera
    b. Campo de trabajo de la carrera elegida
    c. Formación técnica
    d. Formación humanista
    e. Visitas a obras y empresas de ingeniería
    f. Crecimiento personal y problemática existencial
    g. Inserción en el campo laboral y
    h. Sentido de pertenencia a la Facultad, y a la UNAM.

- Lograr que los estudiantes sean creativos, reflexivos, polifuncionales y emprendedores, que asuman su calidad de sujeto activo, protagonista de su propio aprendizaje y gestor de su proyecto de vida y dispuestos a estudiar toda la vida, aprender a aprender, aprender a emprender y aprender a ser.

1.4 Etapas

Las etapas que se consideran en este Programa son:

- En la primera etapa, se atiende a los alumnos durante su primer semestre curricular, en esta etapa se busca que el tutor sea profesor de carrera de la Facultad con amplia experiencia y de reconocido prestigio académico y/o profesional.

- En la segunda etapa, se buscará que los alumnos de 9° y 10° semestres cuenten con un tutor que sea profesional de la ingeniería, esto porque al estudiante en la etapa final de su licenciatura tenga la oportunidad de acercarse y conocer el futuro quehacer cotidiano.

1.5 Definición del problema

En el Área de **T**utoría para **T**odos, la problemática que se presenta en cada inicio de semestre, es la asignación de **tutor/profesor,** debido a que los tutores en cada semestre cambian de actividades o se encuentran realizando un proyecto en la Facultad de Ingeniería, provocando que las horas de asesoría asignadas sean diferentes al semestre anterior. Por otra parte, se presenta el conflicto por los distintos programas de becas, los cuales son: PRONABES (becas proporcionadas por la SEP) Y PARA (**P**rograma para los **A**lumnos de **A**lto **R**endimiento).

Se debe **consultar** un **reporte/lista** de los **alumnos** asignados a cada **tutor,** en este **reporte/lista** se deben cumplir con los requerimientos específicos, toda la información es almacenada en archivos de hojas de cálculo y este procedimiento se repite cada semestre, pero el procedimiento para la asignación de **tutor por alumno** realizado manualmente se desea cambiar por medio de un sistema que realiza dicha actividad; esto es, con el fin de ahorrar tiempo y aprovechar la tecnología existente.

1.6 Justificación del problema

Con el desarrollo del sistema **(SIATT)** tratamos de optimizar el tiempo de asignación de **tutor/alumno**, que las **consultas** se realicen en el menor tiempo posible de forma correcta, también que cuando se requiera un **reporte/listado** se obtenga con la información necesaria y clara para su uso, que se genere el archivo de almacenamiento de cada uno de los procedimientos que se realizan con los datos, esto para, mejorar el control del manejo de la información y se tenga un respaldo de toda la información en caso de que se quiera hacer un estudio estadístico.

Para el manejo de datos tanto de **alumnos** y **tutor/profesor** se realizará el diseño de una base de datos para que no se pierda información, y centralizar toda aquella información que se necesite para el mayor rendimiento del sistema **(SIATT).**

CAPÍTULO 2

Fundamentos Teóricos

2.1 Concepto de una base de datos

Primero se hablaba de archivos y conjuntos de datos, ahora de grandes bases; algunos autores afirman que **las bases de datos son una recopilación de los grandes bancos de datos**, pero para definirlo correctamente se muestran a continuación algunos textos:

"Se entiende como un archivo de datos interrelacionados, recolectados, que satisfacen las necesidades de información de una comunidad determinada de usuarios. Cada unidad de información almacenada en una base de datos está compuesta por datos elementales, cada uno de los cuales representa características particulares de la entidad que se describe. Por ejemplo, una base de datos bibliográficos contendrá información sobre libros, reportes, artículos de revista, etcétera."[1]

"Una base de datos es una colección de datos estructurados según un modelo que refleje las relaciones y restricciones existentes en el mundo real. Los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas y, su definición y descripción han de ser únicas estando almacenadas junto a los mismos. Por último, los tratamientos que sufran estos datos tendrán que conservar la integridad y seguridad de éstos."[2]

"**Colección de datos correspondientes a las diferentes perspectivas de un sistema de información** (de una empresa o institución), existentes en algún soporte de tipo físico (normalmente de acceso directo), agrupados en una organización integrada y centralizada en la que figuran no sólo los datos en sí, sino también las relaciones existentes entre ellos, y de forma que se minimiza la redundancia y se maximiza la independencia de los datos de las aplicaciones que los requieren."[3]

"**Una base de datos es una colección de archivos interrelacionados creados con un DBMS** (Sistema Manejador de una Base de datos). El contenido de una base se obtiene combinando datos de todas las diferentes fuentes en una organización, de tal manera que los datos estén disponibles para todos los usuarios, y los datos redundantes puedan eliminarse, o al menos minimizarse."[4]

---

[1] Gil Rivera, Ma. del Carmen. *Las Bases de datos, importancia y aplicación en la educación*.
[2] Mota Herranz, Laura. *Bases de datos relacionales: teoría y diseño*. Universidad Politécnica de Valencia. Valencia. 1994. Pág. 9.
[3] Guilera Aguera, Llorenc. *Introducción a la informática*. Edunsa. Barcelona. 1988. Pág. 37.
[4] Alice Y., H. Tsai. *Sistema de base de datos: Administración y uso*. Prentice Hall Hisponoamericana S.A., 1990. Canada. Pág. 5.

Lo que conduce a que, las bases de datos son un conjunto de información relacionada no redundante, que está organizada, sistematizada y debe encauzarse en un propósito específico en beneficio de una comunidad. Del mismo modo tiene que cumplir con los objetivos de independencia entre las aplicaciones y las estructuras de datos, integridad en los datos y la seguridad de éstos ante los múltiples usuarios que la utilicen.

Cronología de las Bases de Datos.

Con el propósito de presentar diversos acontecimientos de más trascendencia del desarrollo de las bases de datos, se muestra la tabla 2.1

Año	Descripción
1960	Uso de los archivos separados ISAM (Index Sequencial Access Method) y VSAM (Virtual Storage Access Method) fueron sistemas administradores de archivos.
1964	El término Base de datos, primero usado en las Fuerzas Militares de EEUU aparece en las publicaciones. IBM introduce IDS (Almacén de Datos Integrado).
1969	IBM introduce IMS, una base de datos jerárquica (con el fin de programas de conquista especial, permite construir cadenas de registros entre ficheros y recorre dichas cadenas). En octubre el DBTG (Database Task Group) envió su primer informe a la PLC, proponiendo un Lenguaje de Descripción de Datos (DDL) para describir una base de datos y un Lenguaje de Manipulación de Datos (DML). Los primeros SGBD (**S**istemas **G**obernadores de **B**ases de **D**atos) se caracterizan por la separación de la descripción de los datos de la manipulación de estos por los programas de aplicación.
1970	E.F. Codd hizo un documento describiendo modelo relacional (definición de modelo relacional) basado en la simplicidad matemática del Álgebra Relacional. A mediados de los 70's surgió SEQUEL y se implantaron en forma **experimental.**
1971	Se produjo un informe (DBTG) sugiriendo por primera vez dos lenguajes de descripción de datos, uno para el esquema y otro para el subesquema. Se forma una comisión llamada Comisión de Lenguajes de descripción de Datos (DDLC).
1972	Finales de 1971 y principios de este año aparece el primer borrador del modelo Codasyl con una arquitectura de dos niveles: un esquema que proporciona la vista del sistema y un subesquema que proporciona la vista del usuario.
1973	La DDLC publicó su propuesta en Journal of Development, reconociendo que el proceso de desarrollo en lenguajes de bases de datos debería ser evolutivo, como en el COBOL.
1976	DER Chen. Surge el SQL. Chen introduce el modelo entidad-vínculo (ER).

1977	Oracle Corp. Se establece. Surge QUEL (Quero Lenguaje) del sistema INGRES. Surge el lenguaje QBE (Quero by Example) desarrollado por IBM.
1978	Oracle hace la primera implementación real del modelo relacional. La comisión técnica X3H2 ha comenzado a estandarizar el modelo de red. Se lanzó una definición borrador de un esquema de almacenamiento, pero jamás ha sido aprobada ninguna versión final.
1979	Primer RDBMS comercial Oracle V2.
1980	Se funda Informix (3° generación de DBMS), cerca de 10000 usando DBMS en EEUU. Para PC Paradox. Evolución del modelo relacional.
1981	IBM comenzó a comercializar el SQL. La pequeña compañía Ashton – Tate introduce DBASE II (no es de DBMS, es de archivos). Las facilidades de Codasyl incluyen un lenguaje de descripción de datos (DDL) de subesquema y un correspondiente Lenguaje de Manipulación de Datos (DML) Cobol, esto es, el Cobol JoD de 1981.
1982	Inician los esfuerzos de ANSI SQL.
1983	Oracle libera sus primeros DBMS que corren en mainframes, minicomputadoras y PC's. IBM lanzó al mercado su producto DATABASE2, más conocido por su abreviatura DB2. Se disolvió la principal comisión Codasyl que se ocupaba del modelo, porque terminó su objetivo al crear un modelo con éxito, que ANSI tomó a su cargo para estandarizar.
1984	Ashton – Tate introduce DB III. El trabajo sobre el modelo de res (ANSI, 1984 a, b, c d) basado en el modelo Codasyl, está casi completo, mientras que el modelo relacional (ANSI/SPARC 1984) está todavía en la etapa inicial (Gallagher, 1984).
1985	Comenzaron a conectar las computadoras por medio de redes de área local (LAN). IBM introduce DB2 para maxicomputadoras o mainframes bajo Sistema Operativo MVS.
1986	Informix pasa a ser una compañía pública. Oracle libera su primer DBMS con capacidades de distribución. Su publica el estándar ANSI SQL. El Dr. Date define las BD Distribuidas. SQL fue adaptado como el estándar ANSI (American Nacional Standarts Institute), par los lenguajes de DB relacionales. Los lenguajes usados son: **SQL** Structure Quero Lenguage, **QUEL** Query Lenguaje, **QBE** Quero By Example. DB orientadas a objetos. Primera generación Gbase francesa Grápale. Surge la primera generación de SGNDOO (Sistema Gobernador de DB Orientada a Objetos).

1987	ISO (Internacional Standards Organization) adopta el estándar SQL. Serv. Corp. Introdujo GemStone.
1988	Se libera SQL Server para OS/2 por parte de Microsoft y SyBase. Libera Oracle V6. Ashton – Tate introduce DBASE IV. Ontologic lanzá Vbase. Simbolics introdujo Static (apoyo a inteligencia artificial como LISP – Lenguaje de Procesamiento de Listas).
1989	ANSI -89 SQL. El estándar SQL fue actualizado. Se forma SQL Access Group. Se utiliza la arquitectura cliente/servidor y plataforma común. Segunda etapa de SGBDOO.
1990	Ashton –Tate introduce dBase IV Tercera generación de BD 0.0 (Integridad en forma más rápida y simple), sobre todo en áreas de investigación. Prog. O.O simula -67 y smalltalk -80.
1991	Modelo orientado a objetos (autores: Jeffcoate, Guilfoyle y Deutsch). Oracle alcanza el poder de 1000 TPS en una máquina de cómputo paralelo. Las bases de objetos (también conocidas como bases de datos orientadas a objetos o bases de datos objetuales). El ODMG (Object Management Group, hoy día Object Data Management). El uso creciente de Unified Modeling Lenguage (UML) para la especificación y diseño de sistemas de información propicia que sus datos se gestionen en bases de objetos de una manera más natural-tecnología de objetos del principio al final- que en SGBDs.
1992	ANSI -92 SQL Triggers (disparador : acciones ejecutadas automáticamente) SQL Access Group publica las especificaciones CLIIENTE Microsoft libera ODBC 1.0 para Windows. Se renueva el estandar para SQL, ANSI -92 SQL.
1993	Oracle implanta DB distribuidas. Access puede trabajar con datos dependientes de dBase, Paradox, Foxpro. Paradox 4.0 de Borland anuncia PAL (Paradox Applications Language for Windows) SyBase anuncio la versión 10.0, una versión completamente nueva de SQL Server y sin intervención de Microsoft. Libera Oracle 7 para UNIX. Se desarrolla un Ambiente de Desarrollo Cooperativo(CDE) de Oracle. Microsoft libera su versión de SQL Server de 32 para Windows NT.
1994	FoxPro Microsoft. Microsoft libera ODBC 2.0 para Windows. **Liberan Oracle 7 para PC** En abril Microsoft anunció que desarrollaría su propio producto servidor de bases de datos. Las versiones posteriores de Microsoft SQL Server traían recuerdos de SQL Server de SyBase. Microsoft permite portar las especificaciones de ODBC a otras plataformas no Windows por parte de Visigenic Software. ODBC 2.0 SDK es disponible para los UNIX de SUN, HP e IBM.

	SQL Access Group se fusiona con X/Open.
1995	Se anuncia DB2/2 de IBM. Oracle 7.1. SQL Server 6.0 de Microsoft (puede trabajar con Access, Lotus Approach y Borland Paradox). En este año Informix tenía el 34.5% del mercado RDBMS para UNIX. El SDK de ODBC está disponible para SCO UNIX, OS/2, Macintosh y Power Macintosh.
1996	Mejor DB Oracle 7.2. Oracle Parallel Server Incrementa Performance introduciendo operaciones en paralelo y asíncronas. Oracle Universal Server transfiere desde PC a poderosos servidores de redes. Soporta Web, multimedia. Oracle8 está en versiones Beta.. SyBase e Informix. Javasoft está preparando una especificación para unir aplicaciones Java con Bases de Datos relacionales. La subsidiaria de Sun Microsystems dispone de una herramienta que permite unir aplicaciones Java a BD ODBC. Java Database Connectivity (JDBC) crea un cambio uniforme para las aplicaciones Java accedan a diferentes bases de datos SQL.
1997	Herramientas OLAP (On Line Analitical Proccessing) ventajas de BD en bloques de tres dimensiones. Oracle se encuentra en la versión 8.i, SQL Server de SyBase en la 11 y SQL Server de Microsoft en la 6.5. Hoy en día las aplicaciones multimedia son vía Internet o Intranet para BD y envuelven un gran número de redes públicas o privadas. En los últimos años se han introducido nuevos conceptos en las bases de datos como: Datawarehousing (BD formada por BD que pueden manejar una cantidad enorme de información) BD Orienta a Objetos (En un inicio se tenían implementaciones de éstas que no cumplían del todo con la ideología de objetos, pero con el tiempo se han ido acoplando a ésta) JDBC (Java Database Connectivity) éste es un API (interfaz de Aplicación) que une a Java con las Bases de Datos.

Tabla 2.1 Cronología en las bases de datos

2.2 Metodología para una base de datos

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

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

El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un *esquema lógico* es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de **DBMS**.

Un *modelo lógico* es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño lógico depende del tipo de **DBMS** que se vaya a utilizar, no depende del producto concreto.

El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un *esquema físico* es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del **DBMS** concreto y el esquema físico se expresa mediante su lenguaje de definición de datos.

2.3 Definición de un DBMS (Sistema Manejador de una Base de datos)

Un sistema manejador de bases de datos (DBMS) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un **DBMS** es proporcionar una **forma de almacenar y recuperar** la información de una base de datos de manera que sea tanto *práctica* como *eficiente*.

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información.

La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información.

2.3.1 Tipos de DBMS

El criterio principal que se utiliza para clasificar los **DBMS** es el modelo lógico en que se basan. Los modelos lógicos empleados con mayor frecuencia en los **DBMS** comerciales actuales son:

- relacional
- de red
- jerárquico.

Algunos **DBMS** más modernos se basan en modelos orientados a objetos.

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

En **el modelo de red** los datos se representan como colecciones de registros y las relaciones entre los datos se representan mediante conjuntos, que son punteros en la implementación física. Los registros se organizan como un grafo: los registros son los nodos y los arcos son los conjuntos. El **DBMS** de red más popular es el sistema **IDMS.**

El modelo jerárquico es un tipo de modelo de red con algunas restricciones. De nuevo los datos se representan como colecciones de registros y las relaciones entre los datos se representan mediante conjuntos. Sin embargo, en el modelo jerárquico cada nodo puede tener un solo padre.

Una base de datos jerárquica puede representarse mediante un árbol: los registros son los nodos, también denominados segmentos, y los arcos son los conjuntos. El **DBMS** jerárquico más importante es el sistema **IMS**.

La mayoría de los **DBMS** comerciales actuales están basados en el modelo relacional, mientras que los sistemas más antiguos estaban basados en el modelo de red o el modelo jerárquico. Estos dos últimos modelos requieren que el usuario tenga conocimiento de la estructura física de la base de datos a la que se accede, mientras que el modelo relacional proporciona una mayor independencia de datos.

Se dice que el modelo relacional es declarativo (se especifica qué datos se han de obtener) y los modelos de red y jerárquico son navegacionales (se especifica cómo se deben obtener los datos).

El **modelo orientado a objetos** define una base de datos en términos de objetos, sus propiedades y sus operaciones. Los objetos con la misma estructura y comportamiento pertenecen a una clase, y las clases se organizan en jerarquías o grafos acíclicos. Las operaciones de cada clase se especifican en términos de procedimientos predefinidos denominados métodos.

Algunos **DBMS** relacionales existentes en el mercado han estado extendiendo sus modelos para incorporar conceptos orientados a objetos. A estos **DBMS** se les conoce como sistemas **objeto-relacionales.**

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

Un tercer criterio es el número de sitios en los que está distribuida la base de datos. Casi todos los **DBMS** son **centralizados**, sus datos se almacenan en una sola computadora o servidor. Los **DBMS** centralizados pueden atender a varios usuarios, pero el **DBMS** y la base de datos en sí residen por completo en una sola máquina. En los **DBMS distribuidos** la base de datos real y el propio software del **DBMS** pueden estar distribuidos en varios sitios conectados por una red. Los **DBMS distribuidos homogéneos** utilizan el mismo **DBMS** en múltiples sitios. Una tendencia reciente consiste en crear software para tener acceso a varias bases de datos autónomas preexistentes almacenadas en **DBMS distribuidos heterogéneos**.

Esto da lugar a los **DBMS federados o sistemas multibase de datos** en los que los **DBMS** participantes tienen cierto grado de autonomía local. Muchos **DBMS** distribuidos emplean una arquitectura cliente-servidor.

Por último, los **DBMS** pueden ser de **propósito general** o de **propósito específico**. Cuando el rendimiento es fundamental, se puede diseñar y construir un **DBMS** de propósito especial para una aplicación específica, y este sistema no sirve para otras aplicaciones.

Muchos sistemas de reservas de líneas aéreas son **DBMS de propósito especial** y pertenecen a la categoría de **sistemas de procesamiento de transacciones en línea** (**OLTP**), que deben atender un gran número de transacciones concurrentes sin imponer excesivos retrasos.

2.3.2 Características de un DBMS

Edgar Frank Codd, el creador del modelo relacional ha establecido una lista con los **ocho servicios** que debe ofrecer todo **DBMS**.

1. Debe proporcionar a los usuarios la capacidad de almacenar datos en la base de datos, acceder a ellos y actualizarlos. Ésta es la función fundamental y por supuesto, debe ocultar al usuario la estructura física interna (la organización de los archivos y las estructuras de almacenamiento).

2. Debe proporcionar un catálogo en el que se almacenen las descripciones de los datos y que sea accesible por los usuarios. Este catálogo es lo que se denomina diccionario de datos y contiene información que describe los datos de la base de datos (metadatos).

3. Un **DBMS** debe proporcionar un mecanismo que garantice que todas las actualizaciones correspondientes a una determinada transacción se realicen, o que no se realice ninguna. Una ***transacción*** es un conjunto de acciones que cambian el contenido de la base de datos. Una transacción en el sistema informático de la empresa inmobiliaria sería dar de alta a un empleado o eliminar un inmueble.

Una transacción un poco más complicada sería eliminar un empleado y reasignar sus inmuebles a otro empleado. En este caso hay que realizar varios cambios sobre la base de datos. Si la transacción falla durante su realización, por ejemplo porque falla el hardware, la base de datos quedará en un estado inconsistente.

Algunos de los cambios se habrán hecho y otros no, por lo tanto, los cambios realizados deberán ser deshechos para devolver la base de datos a un estado consistente.

4. Debe proporcionar un mecanismo que asegure que la base de datos se actualice correctamente cuando varios usuarios la están actualizando concurrentemente. Uno de los principales objetivos de los **DBMS** es el permitir que varios usuarios tengan acceso concurrente a los datos que comparten.

El acceso concurrente es relativamente fácil de gestionar si todos los usuarios se dedican a leer datos, ya que no pueden interferir unos con otros. Sin embargo, cuando dos o más usuarios están accediendo a la base de datos y al menos uno de ellos está actualizando datos, pueden interferir de modo que se produzcan inconsistencias en la base de datos.

El **DBMS** se debe encargar de que estas interferencias no se produzcan en el acceso simultáneo.

5.- Debe proporcionar un mecanismo capaz de recuperar la base de datos en caso de que ocurra algún suceso que la dañe.

Como ya se ha expresado, cuando el sistema falla en medio de una transacción, la base de datos se debe devolver a un estado consistente. Este fallo puede ser a causa de un fallo en algún dispositivo hardware o un error del software, que hagan que el **DBMS** aborte, o puede ser a causa de que el usuario detecte un error durante la transacción y la aborte antes de que finalice.

En todos estos casos, el **DBMS** debe proporcionar un mecanismo capaz de recuperar la base de datos llevándola a un estado consistente.

6.- Un **DBMS** debe proporcionar un mecanismo que garantice que sólo los usuarios autorizados pueden acceder a la base de datos. La protección debe ser contra accesos no autorizados, tanto intencionados como accidentales.

7. Debe ser capaz de integrarse con algún software de comunicación. Muchos usuarios acceden a la base de datos desde terminales. En ocasiones éstas terminales se encuentran conectadas directamente a la máquina sobre la que funciona el **DBMS** y en otras, las terminales están en lugares remotos, por lo que la comunicación con la máquina que alberga al **DBMS** se debe hacer a través de una red.

En cualquiera de los dos casos, el **DBMS** recibe peticiones en forma de mensajes y responde de modo similar. Todas estas transmisiones de mensajes las maneja el gestor de comunicaciones de datos. Aunque este gestor no forma parte del **DBMS**, es necesario que el **DBMS** se pueda integrar con él para que el sistema sea comercialmente viable.

8. Un **DBMS** 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. Se puede considerar como otro modo de proteger la base de datos, pero además de tener que ver con la seguridad, tiene otras implicaciones. La integridad se ocupa de la calidad de los datos.

Normalmente se expresa mediante restricciones, que son una serie de reglas que la base de datos no puede violar. Por ejemplo, se puede establecer la restricción de que cada empleado no puede tener asignados más de diez inmuebles. En este caso sería deseable que el **DBMS** controlara que no se sobrepase este límite, cada vez que se asigne un inmueble a un empleado.

Además, de estos ocho servicios, es razonable esperar que los **DBMS** proporcionen una serie de 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 **DBMS**. Algunas de ellas son:

- Herramientas para importar y exportar datos.
- Herramientas para monitorizar el uso y el funcionamiento de la base de datos.
- Programas de análisis estadístico para examinar las prestaciones o las estadísticas de utilización.

- Herramientas para reorganización de índices.
- Herramientas para aprovechar el espacio dejado en el almacenamiento físico por los registros borrados y que consoliden el espacio liberado para reutilizarlo cuando sea necesario.

2.4 Modelado de una base de datos

El modelado de datos es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

Un **modelo** es una representación de la realidad que contiene las características generales de algo que se va a realizar. En base de datos, esta representación se elabora de forma gráfica.

Los modelos de datos se dividen en tres grupos:

- Modelos lógicos basados en objetos
- Modelos lógicos basados en registros
- Modelos físicos de datos

2.4.1 Modelos lógicos basados en objetos

Se usan para describir datos en los niveles conceptual y de visión, es decir, con este modelo representamos los datos de tal forma como nosotros los captamos en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Existen diferentes modelos de este tipo, pero el más utilizado por su sencillez y eficiencia es el modelo **Entidad-Relación**.

2.4.2 Modelos lógicos basados en registros

Se utilizan para describir datos en los niveles conceptual y físico.

Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre estos registros (ligas) o apuntadores. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

Los tres modelos de datos más ampliamente aceptados son:

- **Modelo Relacional**
- **Modelo de Red**
- **Modelo Jerárquico**

2.4.3 Modelo de Red

Este modelo representa los datos mediante colecciones de registros y sus relaciones se representan por medio de ligas o enlaces, los cuales pueden verse como punteros. Los registros se organizan en un conjunto de gráficas arbitrarias.

Ejemplo:



Figura 2.1 Modelo de Red

2.4.4 Modelo Jerárquico

Es similar al modelo de red en cuanto a las relaciones y datos, ya que éstos se representan por medio de registros y sus ligas. La diferencia radica en que están organizados por conjuntos de árboles en lugar de gráficas arbitrarias.

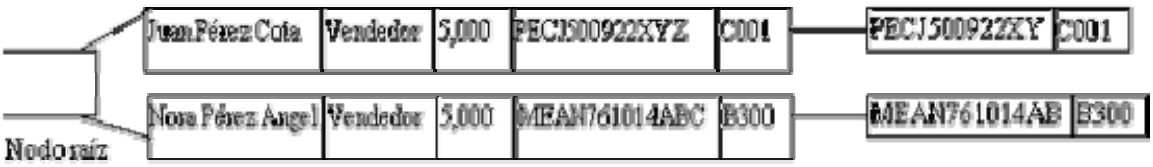


Figura 2.2 Modelo Jerárquico

2.4.5 Modelos Físicos de datos

Se usan para describir a los datos en el nivel más bajo, aunque existen muy pocos modelos de este tipo, básicamente capturan aspectos de la implementación de los sistemas de base de datos. Existen dos clasificaciones de este tipo que son:

- Modelo unificador
- Memoria de elementos

2.5 Modelo Entidad – Relación

Denominado por sus siglas como: E-R; Este modelo representa a la realidad a través de ***entidades***, que son objetos que existen y que se distinguen de otros por sus características, por ejemplo: un alumno se distingue de otro por sus características particulares como lo es el nombre, o el número de control asignado al entrar a una institución educativa, asimismo, un empleado, una materia, etc. Las entidades pueden ser de dos tipos:

- **Tangibles**: Son todos aquellos objetos físicos que podemos ver, tocar o sentir.

- **Intangibles:** Todos aquellos eventos u objetos conceptuales que no podemos ver, aun sabiendo que existen, por ejemplo:

la entidad materia, sabemos que existe; sin embargo, no la podemos visualizar o tocar.

Las características de las entidades en base de datos se llaman ***atributos,*** por ejemplo el nombre, dirección teléfono, grado, grupo, etc. son atributos de la entidad alumno; y los atributos de la entidad de empleado son: clave, número de seguro social, departamento, etc. A su vez, una entidad se puede asociar o relacionar con más entidades a través de ***relaciones.***

Un ejemplo del concepto anterior es:

consideremos una empresa que requiere controlar a los vendedores y las ventas que ellos realizan; de este problema se determina que los objetos o entidades principales a estudiar son el empleado (vendedor) y el artículo (que es el producto en venta), y las características que los identifican son:

Empleado:	*Artículo:*
Nombre	Descripción
Puesto	Costo
Salario	Clave
R.F.C.	

La relación entre ambas entidades se establece como Venta.

Ahora falta describir como se representa un modelo E-R gráficamente, la representación es sencilla, se emplean símbolos, los que son:

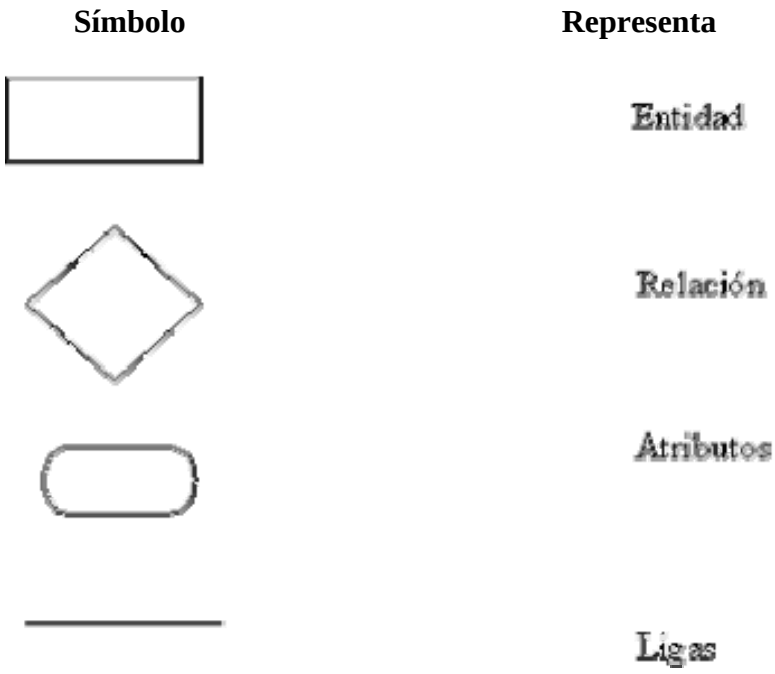


Figura 2.3 Simbología del diagrama E/R

Así el ejemplo anterior quedaría representado de la siguiente forma:

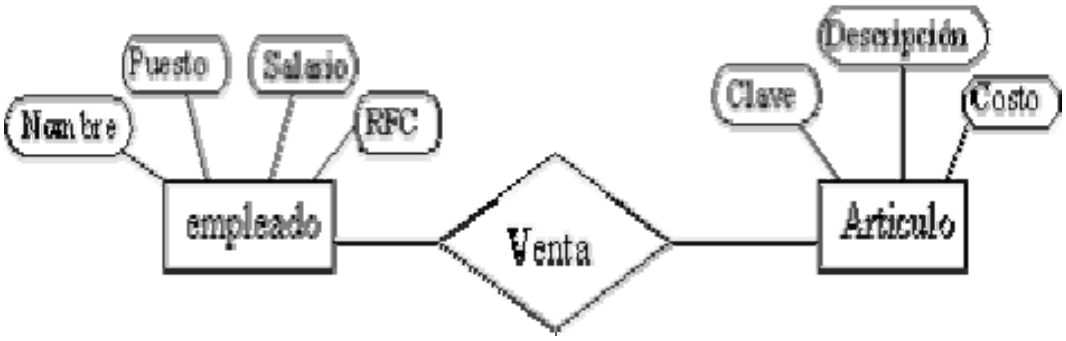


Figura2.4 Ejemplo diagrama E/R

2.6 Reglas de normalización

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener.

La normalización también se puede entender como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede. La normalización se adoptó porque el viejo estilo de poner todos los datos en un solo lugar, como un archivo o una tabla de la base de datos, era ineficiente y conducía a errores de lógica cuando se trataban de manipular los datos.

La normalización hace las cosas fáciles de entender. Los seres humanos tenemos la tendencia de simplificar las cosas al máximo. Lo hacemos con casi todo, desde los animales

hasta con los automóviles. Las guías que la normalización provee crean el marco de referencia para simplificar una estructura de datos compleja.

Otra ventaja de la normalización de base de datos es el consumo de espacio. Una base de datos normalizada ocupa menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia un menor uso de espacio en disco.

El proceso de normalización tiene un nombre y una serie de reglas para cada fase. Esto puede parecer un poco confuso al principio, pero poco a poco se va entendiendo el proceso, así como las razones para hacerlo de esta manera.

2.6.1 Grados de normalización

Existen básicamente tres niveles de normalización: Primera Forma Normal (1NF), Segunda Forma Normal (2NF) y Tercera Forma Normal (3NF). Cada una de estas formas tiene sus propias reglas. Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización. No siempre es una buena idea tener una base de datos conformada en el nivel más alto de normalización, puede llevar a un nivel de complejidad que pudiera ser evitado si estuviera en un nivel más bajo de normalización.

En la tabla siguiente se describe brevemente en que consiste cada una de las reglas, y posteriormente se explican con más detalle.

Regla	Descripción
Primera Forma Normal (1FN)	**Incluye la eliminación de todos los grupos repetidos.**
Segunda Forma Normal (2FN)	**Asegura que todas las columnas que no son llave sean completamente dependientes de la llave primaria (PK).**
Tercera Forma Normal (3FN)	**Elimina cualquier dependencia transitiva. Una dependencia transitiva es aquella en la que las columnas que no son llave son dependientes de otras columnas que tampoco son llave.**

Tabla 2.1 Formas Normales

2.6.2 Primera Forma Normal

La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas.

Poner la base de datos en la Primera Forma Normal resuelve el problema de los encabezados de columna múltiples. Muy a menudo, los diseñadores de bases de datos inexpertos harán algo similar a la tabla no normalizada. Una y otra vez, crearán columnas que representen los mismos datos.

La normalización ayuda a clarificar la base de datos y a organizarla en partes más pequeñas y más fáciles de entender. En lugar de tener que entender una tabla gigantesca y monolítica que tiene muchos diferentes aspectos, sólo tenemos que entender los objetos pequeños y más tangibles, así como las relaciones que guardan con otros objetos también pequeños.

2.6.3 Segunda Forma Normal

La regla de la Segunda Forma Normal establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un término que describe a aquellos datos que no dependen de la llave primaria de la tabla para identificarlos.

Una vez alcanzado el nivel de la Segunda Forma Normal, se controlan la mayoría de los problemas de lógica. Podemos insertar un registro sin un exceso de datos en la mayoría de las tablas.

2.6.4 Tercera Forma Normal

Una tabla está normalizada en esta forma si todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas. Como se mencionó que una dependencia transitiva es aquella en la que existen columnas que no son llave que dependen de otras columnas que tampoco son llave.

Cuando las tablas están en la Tercera Forma Normal se previenen errores de lógica cuando se insertan o borran registros. Cada columna en una tabla está identificada de manera única por la llave primaria, y no debe haber datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir.

La normalización es una técnica que se utiliza para crear relaciones lógicas apropiadas entre tablas de una base de datos. Ayuda a prevenir errores lógicos en la manipulación de datos. La normalización facilita también agregar nuevas columnas sin romper el esquema actual ni las relaciones.

Existen varios niveles de normalización: Primera Forma Normal, Segunda Forma Normal, Tercera Forma Normal, Forma Normal Boyce-Codd, Cuarta Forma Normal, Quinta Forma Normal o Forma Normal de Proyección-Unión, Forma Normal de Proyección-Unión Fuerte, Forma Normal de Proyección-Unión Extra Fuerte y Forma Normal de Clave de Dominio. Cada nuevo nivel o forma nos acerca más a hacer una base de datos verdaderamente relacional.
.

Se discutieron las primeras tres formas. Éstas proveen suficiente nivel de normalización para cumplir con las necesidades de la mayoría de las bases de datos. Normalizar demasiado puede conducir a tener una base de datos ineficiente y hacer a su esquema demasiado complejo para trabajar. Un balance apropiado de sentido común y práctico puede ayudarnos a decidir cuándo normalizar.

2.7 Diccionario de datos

Un **DBMS** debe proporcionar un catálogo en el que se almacenen las descripciones de los datos y que sea accesible por los usuarios. Este catálogo es lo que se denomina diccionario de datos y contiene información que describe los datos de la base de datos (metadatos).

Normalmente, un diccionario de datos almacena:

- 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.

- Estadísticas de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

Algunos de los **beneficios** que reporta el diccionario de datos son los siguientes:

- La información sobre los datos se puede almacenar de un modo centralizado. Esto ayuda a mantener el control sobre los datos, como un recurso que son.
- El significado de los datos se puede definir, 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.
- Las redundancias y las inconsistencias se pueden identificar más fácilmente ya que los datos están centralizados.
- Se puede tener un historial de los cambios realizados sobre la base de datos.
- El impacto que puede producir un cambio se puede determinar antes de que sea implementado, ya que el diccionario de datos mantiene información sobre cada tipo de dato, todas sus relaciones y todos sus usuarios.
- Se puede hacer respetar la seguridad.
- Se puede garantizar la integridad.
- Se puede proporcionar información para auditorias.

2.8 Diagramas de flujos de datos

El **DFD (**Diagramas de Flujos de Datos) es una de las herramientas del análisis estructurado moderno, más importante para el análisis de modelos gráficos, que permite visualizar un sistema como una red de procesos funcionales conectados entre sí por canales (flujo de datos) y depósitos de almacenamiento de datos.

Estos diagramas (figura 2.5) permiten ver como los datos fluyen a través de la organización, los procesos y transformaciones que sufren dichos datos y los diferentes tipos de salidas.

Componentes y símbolos utilizados

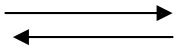
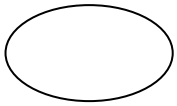
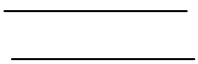
1. Flujo de datos 
2. Proceso 
3. Almacén 
4. Terminador 

Figura 2.5 Componentes

2.8.1 Flujo de datos

Representan la introducción de datos en un proceso o la obtención de datos de un proceso, además representan la actualización de datos en un archivo, una base de datos u otro medio de almacenamiento de datos.

Es una vía por la cual transitan paquetes de datos de composición conocida.

Los datos pueden viajar por cualquier vía dada.

2.8.2 Proceso

Generalmente, el proceso (función o transformación) viene representado por un círculo o por una burbuja, y son acciones que se toman sobre los datos, como por ejemplo, calcular, comparar, imprimir, señalar, marcar, autorizar, almacenar, validar, informar, producir, entre otros.

Los procesos muestran una parte del sistema que transforma entradas en salidas, esto es, muestra cómo es que una o varias entradas se transforma en una o varias salidas.

2.8.3 Nombre del proceso

El nombre de un proceso consiste en una frase VERBO-OBJETO, y describe lo que hace; como por ejemplo:

CALCULAR-IMPUESTO
AUTORIZAR- FIRMA
AUTORIZAR-FACTURA
AUTORIZAR-ORDEN-DE-COMPRA
VALIDAR- PROVEEDOR
GENERAR-REPORTES

Los procesos también, pueden ser descritos (aunque no es recomendable) con el nombre de una persona o un grupo de personas, computadora o un aparato mecánico, de cualquier modo la palabra clave es **Quién** o **Qué** lo está efectuando.

2.8.4 Terminadores (fuentes o destinos de los datos)

1.- **Agentes internos** (entradas al sistema o fuentes), se refieren a tareas efectuadas dentro de la empresa, pero que no forman parte del ámbito del sistema y además le suministran entradas o reciben salidas de él. Ejemplos: otros departamentos, empleados o sistemas de información.

2.- **Agentes Externos** (salidas de un sistema o destinos) son aquellos que son claramente exterior a la empresa. Ejemplos: clientes, proveedores y los organismos gubernamentales.

2.8.5 Almacenes de datos

Un almacén es un inventario de datos y describen cosas sobre las que la empresa desea almacenar datos (figura 2.5).

Es el punto de unión más común entre los modelos de datos y los modelos de procesos.

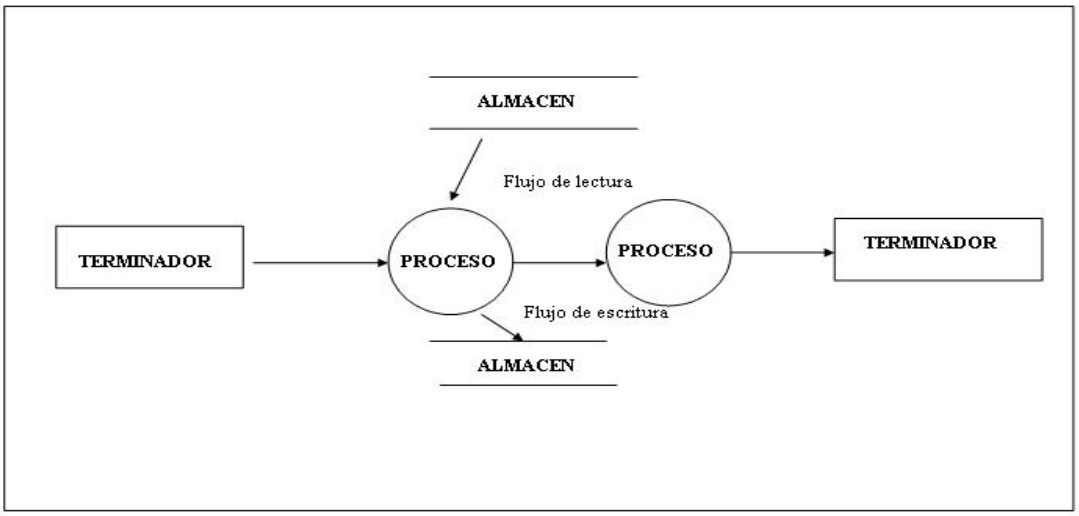


Figura 2.5 Diagrama de Flujo

2.9 Arquitectura cliente / servidor

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor, al proceso que responde a las solicitudes.

Los principales componentes del esquema cliente/servidor son entonces los clientes, los servidores y la infraestructura de comunicaciones.

En este modelo, las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario.

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

Los clientes realizan generalmente funciones como:

- Manejo de la interface del usuario.

- Captura y validación de los datos de entrada.
- Generación de consultas e informes sobre las bases de datos.

Por su parte los servidores realizan, entre otras, las siguientes funciones:

- Gestión de periféricos compartidos.
- Control de accesos concurrentes a bases de datos compartidas.
- Enlaces de comunicaciones con otras redes de área local o extensa.
- Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste, le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura de comunicaciones, que proporciona los mecanismos básicos de direccionamiento y transporte.

La mayoría de los sistemas cliente/servidor actuales, se basan en redes locales y por lo tanto utilizan protocolos no orientados a conexión, lo que implica que las aplicaciones deben hacer las verificaciones. La red debe tener características adecuadas de desempeño, confiabilidad, transparencia y administración.

Entre las principales características de la arquitectura cliente / servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interface única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interface externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

En el sistema cliente /servidor de bases de datos, El cliente envía mensajes que son representados en solicitudes SQL hacia el servidor de bases de datos. Los resultados de cada orden de SQL son devueltos al cliente.

El **DBMS** se encarga de recolectar los datos desde su base de datos, no envía los registros completos, teniéndose un uso mucho más eficiente de la capacidad de procesamiento distribuida. Es usual que se generen aplicaciones en el cliente y en el servidor.

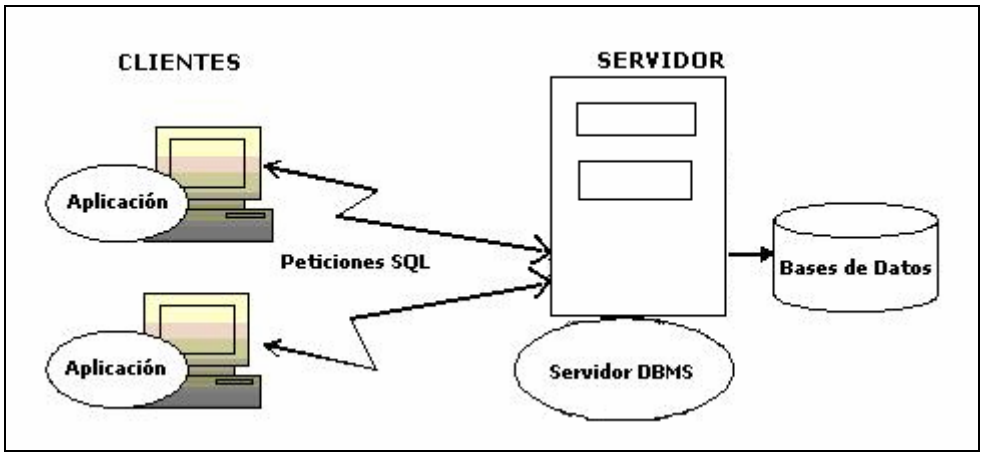


Figura 2.6 Arquitectura cliente servidor

2.9.1 Manejadores de base de datos

Como ejemplo se estableció que un sistema manejador de bases de datos (**DBMS**) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. A continuación se dan las características de los más usados.

2.9.2 PostgreSQL

PostgreSQL es un **DBMS** Objeto-Relacional (**ORDBMS**) que ha sido desarrollado de varias formas desde 1977. Comenzó como un proyecto denominado Ingres en la Universidad Berkeley de California. Ingres fue más tarde desarrollado comercialmente por la Relational Technologies/Ingres Corporation.

En 1986 otro equipo dirigido por Michael Stonebraker de Berkeley continuó el desarrollo del código de Ingres para crear un sistema de bases de datos objeto-relacionales llamado Postgres.

En 1996, debido a un nuevo esfuerzo de código abierto y a la incrementada funcionalidad del software, Postgres fue renombrado a PostgreSQL, tras un breve periodo como Postgres95.

El proyecto PostgreSQL sigue actualmente un activo proceso de desarrollo a nivel mundial gracias a un equipo de desarrolladores y contribuidores de código abierto.

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle.

A continuación se observa una breve lista de algunas de esas características, a partir de PostgreSQL 7.1.x.

DBMS Objeto-Relacional

PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas, herencia y arrays.

Altamente Extensible

PostgreSQL soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario.

Soporte SQL Comprensivo

PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial

PostgreSQL soporta integridad referencial, que es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible

La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++ y Pike.

Lenguajes Procedurales

PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

MVCC

MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Cuando se usa algún DBMS con capacidades SQL, tal como MySQL o Access, hay ocasiones en que una lectura tiene que esperar para acceder a la información de la base de datos.

La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros.

Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor.

En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

Cliente/Servidor

PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Ésta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

Write Ahead Logging (WAL)

La característica de PostgreSQL conocida como *Write Ahead Logging* incrementa la dependencia de la base de datos al registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del que podremos restaurar la base de datos.

Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos.

2.9.3 Oracle

Oracle es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de páginas web pasa lo mismo: Como es un sistema muy caro no está tan extendido como otras bases de datos, por ejemplo, Access, MySQL, SQL Server, etc.

Oracle como antes se mencionó se basa en la tecnología cliente/servidor, pues bien, para su utilización primero sería necesario la instalación de la herramienta servidor (Oracle 8i) y posteriormente podríamos atacar a la base de datos desde otros equipos con herramientas de desarrollo como Oracle Designer y Oracle Developer, que son las herramientas básicas de programación.

Para desarrollar en Oracle utilizamos PL/SQL un lenguaje de 5ª generación, bastante potente para tratar y gestionar la base de datos, también por norma general se suele utilizar SQL al crear un formulario.

2.9.4 Access

Microsoft Access es un sistema interactivo de administración de bases de datos para Windows.

Access tiene la capacidad de organizar, buscar y presentar la información resultante del manejo de sus bases de datos. Entre sus principales características se encuentran:

Access es gráfico, por lo que aprovecha al máximo la potencia gráfica de Windows, ofreciendo métodos usuales de acceso a los datos y proporcionando métodos simples y directos de trabajar con la información.

Access facilita la administración de datos, ya que sus posibilidades de consulta y conexión le ayudan a encontrar rápidamente la información deseada, cualquiera que sea su formato o lugar de almacenamiento.

Con Access es posible producir formularios e informes sofisticados y efectivos, así como gráficos y combinaciones de informes en un solo documento. Access permite lograr un considerable aumento en la productividad mediante el uso de los asistentes y las macros.

Estos permiten automatizar fácilmente muchas tareas sin necesidad de programar.

Una base de datos de Access no es sólo una tabla de datos, sino que es un conjunto de objetos. Access permite crear formularios, informes y otros objetos que le ayudan a

presentar sus datos tal como lo desee, pero la información propiamente dicha, se almacena en tablas.

2.9.5 MS SQL Server

Continuando con la base sólida establecida por SQL Server 6.5. Como la mejor base de datos para Windows NT, SQL Server es el **DBMS** de elección para una amplia gama de clientes corporativos y Proveedores Independientes de Software que construyen aplicaciones de negocios.

Tiene características como administración multi-servidor y con una sola consola; ejecución y alerta de trabajos basadas en eventos; seguridad integrada; y scripting administrativo.

Esta versión también libera al administrador de base de datos para aspectos más sofisticados del trabajo al automatizar las tareas de rutina. Al combinar estos poderosos servicios de administración con las nuevas características de configuración automática.

2.9.6 DB2

Manejador de base de datos de **IBM** que está diseñado para atender las necesidades de servidor de la base de datos de empresas de medianas a grandes.

Puede ser desplegado en ambientes Linux, UNIX, y Windows en servidores de cualquier tamaño.

Proporciona capacidades significativas de automatización que incluyen auto-configuración, auto-optimización y auto-administración, se han hecho perfeccionamientos al DB2 para trabajar con XML que hacen más fácil, a los programadores, la integración del DB2 y la información XML.

Tiene un procesamiento de transacciones o soluciones basadas en Web.

2.9.7 MySQL

Es un manejador de Base de Datos **SQL** (Structured Query Language). Es una implementación Cliente-Servidor que consta de un servidor y diferentes clientes (programas/librerías).Podemos agregar, acceder, y procesar datos grabados en una base de datos. Actualmente, el manejador de base de datos juega un rol central en la informática, como única utilidad, o como parte de otra aplicación.

MySQL es un software de código abierto esto quiere decir que es accesible para cualquiera, para usarlo o modificarlo. Podemos descargar MySQL desde Internet y usarlo sin pagar nada, de esta manera cualquiera puede inclinarse a estudiar el código fuente y cambiarlo para adecuarlo a sus necesidades. MySQL usa el GPL (GNU Licencia Publica General) para definir que podemos y no podemos hacer con el software en diferentes situaciones. Entre otras cuestiones esta licencia aclara que no cuesta dinero a menos que lo incluyamos en un software comercial y tenemos el código fuente.

MySQL es muy rápido, confiable, robusto y fácil de usar tanto para volúmenes de datos grandes como pequeños. La conectividad, velocidad y seguridad hace de MySQL altamente conveniente para acceder a bases de datos en Internet.

Principales Características

- El principal objetivo de MySQL es velocidad y robustez.
- Escrito en C y C++,
- Clientes C, C++, JAVA, Perl, TCL, php.
- Multiproceso, es decir puede usar varios CPU si éstas están disponibles.
- Puede trabajar en distintas plataformas y S.O. distintos.
- Sistema de contraseñas y privilegios muy flexibles y seguros.
- Todas las palabras de paso viajan encriptadas en la red.
- Registros de longitud fija y variable.
- 16 índices por tabla, cada índice puede estar compuesto de 1 a 15 columnas o partes de ellas con una longitud máxima de 127 bytes.
- Todas las columnas pueden tener valores por defecto.
- Utilidad (Isamchk) para chequear, optimizar y reparar tablas.
- Todos los datos están grabados en formato ISO8859_1.
- Los clientes usan TCP o UNIX Socket para conectarse al servidor.
- El servidor soporta mensajes de error en distintas lenguas.
- Todos los comandos tienen -help o -? Para las ayudas.
- Diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma flotante, doble precisión, carácter, fechas, enumerados, entre otros.
- ODBC para Windows 95 (con fuentes), se puede utilizar ACCESS para conectar con el servidor.

CAPÍTULO 3

Herramientas y Lenguajes de Programación

3.1 Servidor Apache

El **SERVIDOR APACHE** es el servidor más popular actualmente, su nombre Apache es **"A PAtCHy Server",** desde su origen a evolucionado hasta convertirse en uno de los mejores servidores en términos de eficiencia, funcionalidad y velocidad. Apache ha demostrado ser más rápido que muchos otros servidores libres, considerándole un servidor de código fuente abierto que ha llegado a competir de cerca con los mejores servidores comerciales como son: Microsoft y Netscape, como plataforma de servidores Web.

La primera versión (0.6.2) de Apache que fue distribuida al público se estrenó en abril de 1995. La versión 1.0 se estrenó el 1 de diciembre de 1995. El grupo Apache se amplió y se convirtió en un grupo sin ánimo de lucro. E1 grupo trabaja exclusivamente vía Internet. Sin embargo, el desarrollo del **Servidor Apache** no está limitado en ningún sentido por el grupo.

Los servidores más utilizados son los mostrados en la tabla 3.1.

Las características más importantes son:

- Apache trabaja con gran cantidad de Perl, **PHP** y otros lenguajes de script.
- Perl destaca en el mundo del script y Apache hace lo mismo con Perl, tanto con soporte CGI como con soporte mod perl.
- Funciona sobre muchas plataformas (como son: Unix, Linux, Vms, Win32,OS2)
- Módulos cargados dinámicamente.
- CGI, Perl (ejemplo: formularios, diccionarios en línea, entre otros)
- **PHP**3 + Bases de datos
- SSL: transacciones seguras
- Soporte para host virtuales
- Alto desempeño

Servidor	Nov 2001	Porcentaje	Dic 2001	Porcentaje
Apache	7750275	61.88	8588323	63.34
Micros/t IIS	3307207	26.40	3609428	26.62
(Planet	431935	3.45	383078	2.83
Zeus	174052	3.45	172352	1.27

Tabla 3.1 Estadística de Netcraft que muestra los servidores más utilizados

3.1.1 Arquitectura de Apache 2.0

Apache Server 2.0 es una solución Web flexible, transportable y escalable. La nueva versión 2.0 tiene las siguientes mejoras.:

- **Módulos multiproceso**

El primer cambio principal en Apache 2.0 es la introducción de los módulos multiproceso (MPM). La versión 1.3 de Apache y las anteriores utilizaban una arquitectura sin bifurcaciones. En este tipo de arquitecturas, un proceso padre de Apache se bifurca como un conjunto de procesos hijos, los cuales son los que en realidad sirven las solicitudes. El proceso padre simplemente monitoriza los hijos y produce o elimina procesos hijos basándose en la cantidad do solicitudes recibidas.

Este modelo no trabajaba bien, bajo plataformas que no estaban centradas en proceso como en el caso de Windows. De modo que, el grupo Apache propuso una solución basada en los MPM. Cada MPM es responsable de iniciar los procesos del servidor y de servir las solicitudes vía procesos hijos o hilos dependiendo de la implementación MPM.

Hay disponibles muchos MPM, algunos de ellos son:

- **El MPM prefork**

Este MPM sin bifurcaciones mimetiza al Apache 1.3 o a otras arquitecturas anteriores, creando un grupo de procesos hijos para servir las solicitudes. Cada proceso hijo tiene un solo hilo. Por ejemplo, si Apache inicia 30 procesos hijo, puede servir 30 solicitudes simultáneamente. Si algo va mal y el proceso hijo muere, únicamente se pierde una solicitud. El número de procesos hijo está controlado fijando un mínimo y un máximo.
Cuando aumenta el número de solicitudes, un nuevo proceso hijo se añade hasta que se alcanza el máximo. Del mismo modo, cuando falla una solicitud, cualquier proceso hijo extra es eliminado.

- **El MPM threaded**

Este MPM permite soporte de hilos en Apache 2.0. Es como el MPM prefork, pero en vez de que cada proceso hijo tenga un solo hilo, cada proceso hijo puede tener un número determinado de hilos. Cada hilo dentro de un proceso hijo puede servir una solicitud distinta.

Si Apache inicia 30 procesos hijo en los que cada hijo puede tener un máximo de 10 hilos, entonces Apache podrá servir 30 x 10 = 300 solicitudes simultáneamente. Si algo va mal con un hilo, por ejemplo, un módulo experimental hace que el hilo muera, entonces el proceso entero muere.

Esto significa que se perderán todas las solicitudes que han servido los hilos dentro del proceso hijo. Sin embargo, como las solicitudes están distribuidas en procesos hijo separados, la muerte de un proceso hijo afecta como máximo a **1/n** del total de las conexiones, donde **n** representa el **número de conexiones simultáneas.**

Los procesos se añaden o se eliminan monitorizando su conteo de hilos de repuesto. Por ejemplo, si un proceso tiene menos de un número mínimo de hilos de repuesto, se añade un nuevo proceso. Del mismo modo, cuando un proceso tiene un máximo de número de hilos parados, se elimina.

Todos los procesos funcionan con el mismo usuario e ID de grupo asignados por el **Servidor Apache**. Dado que los hilos son más eficaces en cuanto a recursos que los procesos, este MPM es muy escalable.

- **El MPM perchild**

Esto también es nuevo en Apache 2.0. En este modelo MPM, un número de procesos hijo se inicia con un número determinado de hilos. Según aumenta la carga de solicitudes, el proceso va añadiendo nuevos hilos a medida que los necesita. Cuando el conteo de solicitudes se reduce, los procesos disminuyen sus conteos de hilos utilizando un conteo mínimo y máximo fijo.

La diferencia clave entre este módulo y el MPM threaded es que el proceso de conteo es estático, además cada proceso puede ejecutarse utilizando un usuario y un ID de grupo distintos. Esto facilita le ejecución de distintos sitios Web virtuales bajo distintos usuarios e ID de grupos.

- **El MPM winnt**

Este es el MPM para la plataforma Windows, incluido Windows 2000, Windows NT y Windows 9x. Se trata de un módulo multihilo. Utilizando este módulo, Apache creara un proceso padre y un proceso hijo. El proceso hijo crea todos los hilos que sirve la solicitud. Además, este módulo saca partido de algunas llamadas a funciones de Windows, que le permiten funcionar mejor en la plataforma Windows que las versiones anteriores del **Servidor Apache**.

- **Filtrado I/O**

Ahora Apache 2.0 proporciona arquitectura para I/O jerarquizada. Esto significa que un output de un módulo puede convertirse en un input de otro módulo. El efecto de este filtrado es muy interesante.

Por ejemplo, el output producido por scripts de CGI, que es procesado por el módulo mod _ cg, puede ahora pasarse al módulo mod _ include responsable de las SSI.

En otras palabras, los scripts de CGI pueden producir un output en forma de etiquetas SSI, que se pueden procesar antes de que el output final se enviara al navegador Web.

3.1.2 Portabilidad en tiempo de ejecución

Apache tenía portabilidad interna, lo que hacía el código base menos manejable. De modo que el grupo Apache introdujo el Apache Portable Runtime (ARP). El propósito de APR es proporcionar una sencilla interfaz de C a funciones específicas de plataforma para que se pueda escribir el código una sola vez.

Esto permite que Apache funcione mejor en plataformas como Apache Windows, BeOS, Amiga y OS/2. Ya que gracias a ARP muchos programas como ApacheBench, pueden ejecutarse en estas plataformas.

3.1.3 Requisitos del sistema

Apache se ejecuta en prácticamente cualquiera de las plataformas que se utilizan hoy en día, incluido; FreeBSD; OpenBSD; NetBSD; BSDI; Amiga OS 3.x;

3.2 Manejadores de Base de Datos de Libre Distribución (MySql) vs. Licenciados

Se muestran las características de **MySql** y **Oracle**, con la finalidad de dar a conocer las razones para decidir Manejador de **Base de datos** se va a trabajar de acuerdo con la aplicación a desarrollar.

3.2.1 Oracle

Es una potente herramienta **cliente/servidor** para la gestión de **Bases de Datos.**
Para su utilización primero será necesario la instalación de la herramienta servidor (**Oracle 8i**) y posteriormente se podrá acceder a la **Base de datos** desde otros equipos con herramientas de desarrollo como **Oracle** Designer y **Oracle** Developer, que son las herramientas básicas de programación sobre **Oracle**.

Para desarrollar en **Oracle** utilizamos PL/SQL un lenguaje de 5ª generación, bastante potente para tratar y gestionar la **Base de datos**, por norma general se suele utilizar SQL al crear un formulario.

Es posible lógicamente ingresar a la **Base de datos** a través del SQL plus incorporado en el paquete de programas **Oracle** para realizar consultas, utilizando el lenguaje SQL.

- **Oracle Developer**

Es una herramienta que permite crear formularios en local, es decir, mediante esta herramienta se puede crear formularios, compilarlos y ejecutarlos, pero si lo que se desea es que los otros trabajen sobre este formulario se deberá copiar regularmente en una carpeta compartida para todos, de modo que, cuando se realice un cambio, se deberá copiar de dicha carpeta y luego a subir a la carpeta.

Este sistema como se puede observar es bastante engorroso y poco fiable, es normal que las versiones se pierdan y se cambien con frecuencia. La principal ventaja de esta herramienta es que es intuitiva y dispone de un modo que nos permite componer el formulario, tal y como lo haríamos en Visual Basic o en Visual C.

- **Oracle Designer**

Es una herramienta que se conecta a la **Base de datos** y por tanto se crean los formularios en ella, de esta manera todo el mundo se conecta mediante Designer a la aplicación que contiene todos los formularios y no hay problemas de diferentes versiones, esto es muy útil y perfecto para evitar cambiar el trabajo de otros.

Pero el principal problema es la falta de un entorno visual para diseñar el formulario, es decir, aparece una estructura como de árbol en el que se inserta un formulario, a la vez dentro de éste se insertan bloques o módulos, que son las estructuras que contendrán los elementos del formularios, que pueden estar basados en tablas o no.

Oracle es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de páginas Web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras bases de datos, por ejemplo:

- Access
- **MySQL**
- SQL Server

- **FUNCIONES ORACLE**

Una función es un conjunto de instrucciones en PL/SQL, que pueden ser llamados usando el nombre con que se le haya creado. Se diferencian de los procedimientos, en que las funciones retornan un valor al ambiente desde donde fueron llamadas.

3.2.2 MYSQL

MySql es una **Base de datos** muy popular, las principales metas de su diseño son:

- Velocidad
- Robustez y
- Fácil manejo

MySql es un servidor multihilos de **bases de datos** de código abierto, confiable, rápido, compacto, poderoso y multiplataforma se puede hacer las bases de datos a código abierto.

Esta **Base de datos** la desarrollo la empresa **MySql AB**, una gran ventaja es que se puede utilizar gratis y su código fuente siempre esta disponible, nos podemos guiar por medio de manuales.

Ésta es una de las bases de datos más fuertes en nuestro medio.

Sus principales características son:

- El principal objetivo de **MySql** es velocidad y robustez
- Escrito en C y C++, testado con GCC 2.7.2.1. Usa GNU autoconf para potabilidad.
- Clientes C, C++, Java, Perl, TCL, etc.
- Multiproceso, es decir puede usar varias CPU si éstas están disponibles
- Puede trabajar en distintas plataformas y S.O. distintos.
- Sistema de contraseñas y privilegios muy flexibles y seguros
- .Registros de longitud fija y variable.
- Índices por tabla, cada índice puede estar compuesto de 1 a 15 columnas o partes de ellas con una longitud máxima de 127 bytes.
- Todas las columnas pueden tener valores por defecto.

- Los clientes usan TCP o UNIX Socket para conectarse al servidor.
- El servidor soporta mensajes de error en distintas lenguas.
- Todos los comandos tienen -help o -? Para las ayudas.
- Diversos tipos de columnas como enteros de 1, 2, 3, 4, y 8 bytes, coma flotante, doble precisión, carácter, fechas, enumerados, etc.
- Todos los datos están grabados en formato ISO8859_1.
- Lenguajes que se comunican con **MySql**: C, C++, Eiffel, Java, Perl, **PHP**, Python y TCL

En la figura 2.7 se muestra la relación entre el servidor y los clientes en una Base de datos mediante la Web

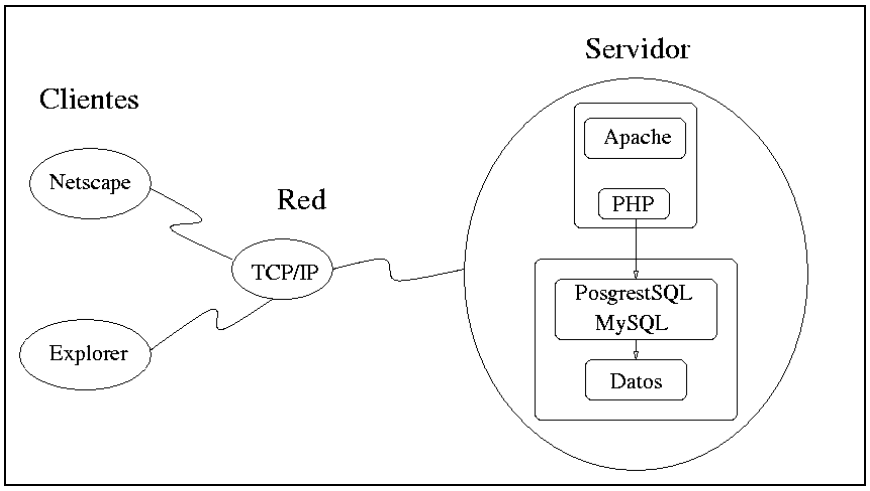


Figura 2.7 Base de datos mediante la Web.

3.2.3 Lenguajes de Programación de Libre Distribución (PHP) vs. Licenciados

Estas son las razones importantes en el momento de hacer la elección entre un Lenguaje de Programación de Libre Distribución vs. Un Lenguaje de Distribución con Licencia.

Es fácil de aprender

PHP es un lenguaje script muy sencillo, ya que se toma en cuenta que sí se tienen conocimientos en lenguaje C

Hoy en día están de sobra los sitios en Internet adonde se puede aprender a programar en **PHP**, se puede aprender a través de Internet y algunos libros. En la mayoría de programadores de **PHP**, lo han aprendido de la misma forma.

Es multiplataforma

Una de las grandes ventajas de **PHP** es, utilizar el mismo código en casi cualquier Sistema Operativo del mercado. Esto es de gran importancia puesto que se puede usar sobre Software gratuito (Linux+Apache), que además de todo, es el más seguro del mercado, en lo que ha servidores respecta. Si se desea puede desarrollar la aplicación en Windows o MAC, pero luego subir el mismo código al servidor LINUX, Solaris, etc.

La portabilidad de **PHP** es algo que lo hace más poderoso. **PHP** corre en más de 25 Plataformas diferentes, entre ellas LINUX, Windows, Solaris y MAC. Todo esto sin necesidad de un componente adicional que deba comprar.

Tiene funciones nativas para la mayoría de Bases de Datos

Ésta es una de las razones que más se decide por **PHP**, las funciones nativas que **PHP** tiene para **MYSQL**, MSSQL y **ORACLE**. Además de las funciones para estas Bases de Datos, también tiene funciones nativas para muchas más, como Postgress, SYBASE, SQLite, Informix, DBase, Adabas y muchas otras.

Sin duda la velocidad de respuesta es mejor con funciones nativas que con algún mediador. Por otro lado cuando se utilizan Bases de Datos potentes como **Oracle** y se utiliza ODBC, OLE, ADO, etc., se pierde gran parte del poder, pues hay funciones propias de la **Base de Datos** que no se pueden utilizar con un mediador genérico como éstos. Por lo tanto, si se quiere utilizar todo el poder de la **Base de datos**, solamente se puede con funciones nativas para ésta.

Es código abierto

Una ventaja del Código Abierto es que no está ligado de ninguna manera ha alguna empresa que puede llegar a tomar decisiones que afecten a los usuarios. Si bien **PHP** hoy en día utiliza el Zend Engine, si ellos tomaran alguna decisión arrebatada sobre **PHP**, algún grupo de desarrolladores podría tomar el código fuente y darle seguimiento por otras vías.

Así como lo sucedido con Netscape que desapareció, pero con sus fuentes liberados se han desarrollado Mozilla, Firefox, etc.

Por ser código abierto hay muchas herramientas que se adaptan grandemente a este, pues hay desarrolladores que han creado funciones para que **PHP** se pueda utilizar con **Oracle**; con FLASH, el que se adapta grandemente a **PHP** e incluso se pueden desarrollar películas SWF desde **PHP**, utilizando la librería gratuita MING; con documentos PDF, los que se pueden generar desde **PHP**, agregando alguna de las librerías para dicho fin; así como estas hay funciones propias para las más grandes bases de datos del mercado: MSSQL, **MYSQL**, SYBASE e Informix.

Por ser Código Abierto no se debe considerar que **PHP** sea menos seguro, pero si esto fuera así, algunos grandes sitios como Yahoo o Amazon no lo estarían utilizando. La seguridad que **PHP** provee en si mismo, es mejor, si se monta sobre Apache, el que tiene una menor vulnerabilidad a ataques que el Windows Server de Microsoft.

Velocidad

PHP es bastante veloz, sobre todo cuando se utiliza como módulo de Apache. Aun y cuando es un lenguaje interpretado, es bastante rápido e incluso mejora los tiempos de respuesta que tienen sus competidores que hoy en día son en su mayoría lenguajes compilados.

Es gratuito, la gran mayoría de componentes, librerías y otros, también son gratuitos

Si bien hoy en día algunas versiones de LINUX para servidor son pagadas, el precio de éstas son mucho menor que el de una copia de Windows, además que las políticas de uso son mucho más abiertas.

Por otro lado, para **PHP** hay librerías gratuitas casi que para hacer cualquier cosa, hay reposiciones de clases muy completos (como Pearl), sistemas de plantillas (como SMARTY), foros (como **PHP**BB), chats, BLOG (como PLOG, que es el mismo de este BLOB, el cual esta en **PHP** y es Open Source), portales (como **PHP**NUKE) y muchas herramientas más que son gratuitas, las que se pueden utilizar y adaptar.

A continuación se presenta en la Tabla 3.2, los factores más importantes para seleccionar una tecnología entre **PHP**4, **PHP**5 y **ASP.NET**.

	PHP 4	**PHP** 5	**ASP.NET**
Software price	Free	free	Free
Platform price	Free	free	$$
Speed	strong	strong	Weak
Efficiency	strong	strong	Weak
Security	strong	strong	Strong
Platform	strong	strong	Weak(IIS only)
Platform	any	any	Win32(IIS only)
Source available	yes	yes	yes
Exceptions	no	yes	no
OOP	weak	strong	strong

Tabla 3.2 Tabla Comparativa

La más fuerte y principal característica de **PHP** es el soporte para una gran variedad de Bases de Datos, que se muestran en la Tabla 3.3.

Adabas D	InterBase	Solid
dBase	mSQL	Sybase
Empress	**MySql**	Velocis
FilePro	**Oracle**	Unix dbm
Informix	PostgreSQL	

Tabla 3.3 Bases de Datos en donde se puede encontrar código de PHP

Las ventajas que se presentan al utilizar **PHP**, son:

- **PHP** es un potente lenguaje de script del lado del servidor, que se utiliza principalmente para generar páginas de forma dinámica.
- Libre.
- Abierto.
- Código fuente disponible.
- Diseñado para la Web.
- Multiplataforma HW.
- Multisistema Operativo.
- Soporte para varios servidores Web.
- Soporte nativo para prácticamente cualquier **Base de datos**.
- Buena documentación.
- Miles de ejemplos y código fuente disponible.
- Perfecta integración del **Apache-PHP-MySql**.
- Sintaxis clara y bien definida.
- Bastante sencillo de aprender y utilizar.
- Modulable.
- Seguro (evidentemente tiene errores pero se solucionan mucho antes que otros sistemas propietarios)
- Amplia base de usuarios (Ahora es el número 1, como lo es también **Apache**).
- No dependes de un único proveedor de servicios.

RESUMEN

Son varios los lenguajes de programación que también podrían utilizarse en vez de **PHP**, el caso más factible es **Java**, el cual ofrece características similares, como un código abierto, software libre, manejo para bases de datos, facilidad de aprendizaje, etc.

Se elige **PHP** porque tiene un carácter más delimitado y más específico, con esto nos referimos a que es más orientado a hacer pareja con **MySQL** (pero no por esto deja de ser utilizado en grandes proyectos), lo anterior lo concluimos por su manejo de variables, cadenas en los códigos, funciones de manejo de bases de datos etc. y por lo tanto, tiene algunas ventajas en comparación con la programación en Java.

El acoplamiento **Apache-PHP-MySQL** ha resultado muy confiable, por lo que la mayoría de los sistemas tienen esta configuración. La gran adaptabilidad, la integridad, la documentación, el rendimiento, entre otras cosas, han hecho que el sistema tenga un soporte cada vez más fuerte, además que los códigos fuentes se tienen a la mano y son gratuitos, por lo que se pueden instalar en cualquier máquina sin tener ningún problema de licencias.

El Software Libre nace en la década de los 80's y comienza por la necesidad de tomar otra opción al llamado software **comercial** o **propietario**, este tipo de software permite al usuario tener los códigos fuentes que le permiten usar, modificar y redistribuir el sistema con las restricciones mínimas para garantizar esa misma libertad a otras personas, es decir, prohíbe que un usuario le restrinja la libertad a otro.

Al tener cualquier persona acceso al código, se evita el problema de copyright y de legalidad del uso del software, ya que no existe un dueño, ni monopolios que puedan pelear ventajas o recursos de lo que se realiza con el sistema.

Uno de los principales miembros del software libre es Linux, creado inicialmente por Linus Torvalds, pero que actualmente tiene soporte de millones de personas repartidas en todo el mundo.

Linux destaca por su estabilidad (sistema que reporta menos fallas), lo que conlleva a tener una mejor seguridad, un mejor manejo de datos, menor tiempo y por lo tanto, una reducción considerable en los costos del sistema.

Actualmente muchos usuarios a nivel mundial están cambiando el sistema con el que trabajan por software libre, por ejemplo, universidades, redes gubernamentales y militares, empresas públicas y privadas, ya que este sistema es capaz de cumplir desde tareas muy complejas y de gran escala, hasta tareas de escritorio.

Tiene también grandes ventajas para el manejo de internet y en redes locales tiene muy buenos sistemas de seguridad. Además que se pueden poner estaciones Linux conectadas con estaciones de otros sistemas operativos (UNIX, Windows NT, etc.).

La programación también es una de las tareas en las que se destaca Linux. La popularidad que tiene entre los programadores e ingenieros se debe a las características avanzadas de este sistema, el cual permite desarrollar un software más complejo.

Por todo lo anterior, delimitamos que la plataforma con la que se trabaja el **SIATT** es **Linux**, se escoge el servidor **Apache**, el lenguaje de programación **PHP** y para el gestor de la base de datos **MySQL**.

3.3 Migración a Linux

En la realización del sistema, se pensó en hacer uso de software libre en todos los niveles de la aplicación, tanto en el sistema operativo, como en el servidor, el manejador de bases de datos y el lenguaje de programación de la interfaz, debido a las ventajas ya descritas en los pasados subtemas.

Este deseo de hacer todo el sistema con software libre, no se pudo llevar a cabo, debido a que en el departamento de Asesoría para todos de **COPADI**, donde se va a implantar el sistema, ya se usa el sistema operativo Windows XP y seria innecesario implantar otro sistema operativo (Linux), debido a que es difícil que las personas que usaran el sistema puedan estudiar un poco de Linux, a pesar de ser sencillo, ya que también cuenta con ambiente gráfico, si requiere de algunos comandos básicos de UNIX para su mejor explotación.

El obstáculo que se observó fue que el usuario final, ya tiene sus aplicaciones y programas de oficina, ejecutándose en Windows y funcionando en red, por lo que al cambiar a Linux, muchos de las Aplicaciones y paquetería no aparecerán, por ejemplo Office.

Otro problema sería el entorno de red, que requiere de más conocimiento sobre Linux cuando se de cualquier falla o caída de la red.

Debido a que no se aceptó el cambio de sistema operativo por parte del departamento de Asesoría para todos de **COPADI**, se tendrá que hacer todo el sistema para Windows, pero se seguirá usando software libre en todo lo demás **(servidor, manejador de bases de datos y el lenguaje de programación de la interfaz).**

La posible migración a Linux podría darse en años posteriores ya que existe la probabilidad de usarse este sistema operativo en muchas dependencias de la UNAM, así como ya se usa en muchas dependencias del gobierno y de la propia universidad.

El sistema está diseñado para trabajar exactamente igual, ya sea que resida en un servidor bajo el sistema operativo Linux o Windows y será fácil de migrar de uno al otro, lo que realmente nos interesa es mostrar la forma de instalar las herramientas **(servidor, manejador de bases de datos y el lenguaje de programación de la interfaz)**, en Windows y a continuación explicaremos como podrían instalarse las mismas herramientas en el sistema operativo Linux, para una futura migración del sistema.

En el desarrollo del sistema se usaran las siguientes herramientas:

- **Servidor Apache**
- **PHP** como lenguaje de programación y
- **MySql** como el manejador de la **Base de datos**

Debido a sus ventajas ya descritas anteriormente y principalmente por ser gratuitos, (software libre).

3.4 Instalación de Apache, PHP y MySql en Windows

3.4.1 Instalación del Servidor Apache.

Atráves de Internet se descargo el archivo de instalación para Windows, dicho archivo lo encontramos en la dirección http://www.apache.org/dist/httpd/binaries/win32/ una vez que lo tengamos lo copiaremos en nuestro escritorio y lo ejecutamos haciendo doble clic sobre él.

Entraremos en un programa de instalación estándar de aplicaciones win32, así que nos limitaremos a elegir las opciones que nos muestre el programa de instalación, es muy sencillo, ya que sólo pide rutas de instalación de los archivos y se pueden dejar las que vienen por defecto.

Si no elegimos un directorio distinto Apache se instalará en:

C:\Archivos de programa\Apache Group\Apache

Una vez terminada la instalación vamos al directorio donde tenemos instalado el servidor, en el directorio "\conf\" es donde se alojan los archivos de configuración del servidor.

Dentro de este directorio se encuentra el archivo "httpd.conf" éste es el archivo de configuración que Apache utiliza al ejecutarse.

Para que Apache funcione correctamente deberemos modificar el archivo anterior, para realizar esto, deberemos abrirlo con un editor de texto (por ejemplo WordPad), y guardarlo como archivo de texto plano. Una vez abierto el archivo deberemos buscar la línea que ponga "ServerName". Lo más probable es que encontremos la siguiente línea:

#ServerName "nombre de servidor por defecto"

La debemos sustituir por la siguiente:

ServerName http://

Para poner en marcha el servidor deberemos ir al escritorio de Windows, el servidor se encontrará en la siguiente ubicación del menú de inicio "Inicio|Apache Web Server|star Apache", o por el contrario en la ubicación que hayamos indicado durante la instalación. Al ejecutarlo se nos quedará abierta una ventana MS-DOS en la que estará corriendo el servidor.

Con el servidor en funcionamiento, comprobaremos si funciona correctamente, para ello abriremos un explorador como puede ser Netscape Communicator o Internet Explorer y en la barra de direcciones teclearemos:

http://127.0.0.1 ó http://localhost ó http://"nombre de nuestro PC"

Si tenemos instalado correctamente Apache, veremos la página de bienvenida Apache. Ésta es la página de chequeo que por defecto incluye Apache. En caso de no verla es que no hemos realizado correctamente algún paso anterior.

Por defecto las páginas estarán alojadas en la siguiente dirección de nuestra máquina:

C:\Directorio de instalación de Apache\htdocs\

Por defecto tenemos configurado Apache para que abra el archivo index.html que se encuentre en el directorio que indiquemos. Por lo tanto, cuando en el navegador pongamos la dirección "http://127.0.0.1" el explorador cargará la página index.html del directorio C:\Directorio de instalación de Apache\htdocs\.

Si tenemos alguna duda o queremos saber algo más sobre Apache, en el directorio htdocs\manual existe información mucho más extensa sobre el funcionamiento, configuración e instalación de Apache.

3.4.2 Instalación del lenguaje de programación PHP

Descargamos y descomprimimos el archivo de http://www.php.net/downloads.php. Podemos descomprimirlo por ejemplo en el directorio "C:\PHP4\". Una vez descomprimido, vamos a ese directorio y editamos el archivo **PHP**.ini-dist.

Dentro de él buscamos la sección Rutas y Directorios. Comprobamos que el apartado "extension_dir" tenga el valor "./" esto indica el directorio actual de **PHP** y donde se encontrarán las extensiones de **PHP**.

En la siguiente sección tenemos las extensiones de **PHP**, que nos proporcionan funciones extra a nuestro interprete de **PHP** (muchas de estas extensiones no funcionarán correctamente bajo Windows98).

Seleccionamos las que necesitemos, esto lo hacemos quitando el ";" al principio de la línea. Debemos comprobar que el archivo al que se hace referencia está en nuestro directorio de **PHP** ("C:\PHP4\").

En este archivo podemos configurar muchas más opciones, como por ejemplo opciones para el soporte de distintos sistemas manejadores de bases de datos, pero por el momento no modificaremos nada, solamente en caso de necesitarlo. Una vez que tengamos modificado lo anterior, guardamos el archivo con el siguiente nombre "**PHP**.ini".

A continuación vamos al directorio donde teníamos instalado Apache y en el directorio **"/config/"** editamos el archivo **"httpd.conf"** donde deberemos introducir las siguientes directivas de configuración.

Vamos a la sección "**ScriptAlias**". En esta sección indicamos el alias que tendrá la ruta donde se encuentra nuestro intérprete de **PHP**. Debemos tener la siguiente línea.

```
ScriptAlias /cgi-bin/ "C:/Archivos de programa/Apache Group/Apache/cgi-bin/"
```

La línea anterior es un directorio que Apache tiene por defecto para ejecutar scripts. Debajo de esta línea debemos introducir:

```
ScriptAlias /PHP4/ "C:\PHP4\"
```

En la sección **Addtype**, se encuentra la configuración que indica al intérprete que archivos debe procesar.

Añadimos:

```
Addtype application/x-httpd-PHP .PHP4
Addtype application/x-httpd-PHP .phtml
```

En la sección **Action**, indicamos que es el archivo que sirve de intérprete de **PHP**. Añadiremos:

```
Action application/x-httpd-PHP "/PHP4/PHP.exe"
```

Una vez hecho esto guardamos el archivo y creamos una página de prueba. Guardaremos el archivo en formato de texto plano. Abrimos un archivo nuevo y escribimos el código inferior:

```
<?PHP
echo "Hola, mi primera página PHP";
PHPinfo();
?>
```

Una vez escrito el código anterior lo guardamos en el directorio "/htdocs/" del directorio Apache con el nombre "info.**PHP**". Ejecutamos de nuevo un explorador Web e introducimos la siguiente dirección: "http://127.0.0.1/info.**PHP**". El resultado es una pantalla con información sobre la versión de **PHP** que utilizamos, además de información sobre el valor de las variables que **PHP** maneja. Si no vemos esa pantalla deberemos comprobar los pasos anteriores y comprobar que hemos realizado todo correctamente.

3.4.3 Instalación del manejador de bases de datos MySql.

Una vez que tengamos el archivo de instalación **MySql-shareware_3_22_34-win.zip** el que descargamos de **http://www.MySql.com/Downloads/Win32/MySql-shareware-3.22.34-win.zip** debemos descomprimirlo, por ejemplo, podemos ubicarlo en el directorio "C:\MySqlinst\".

Una vez descomprimido vamos a ese directorio y ejecutamos el programa de instalación

"Setup.exe"

Donde indicaremos las rutas que queremos para su instalación.

Una vez terminada la instalación los archivos ejecutables se encuentran en el directorio "\bin\". **MySql** se comporta como un servidor, por tanto para tener acceso al **DBMS** (**S**istema **M**anejador de **B**ases de **D**atos) deberemos tenerlo en funcionamiento, para hacer esto deberemos ejecutar el archivo

"\bin\MySqld-shareware.exe".

Para acceder al sistema debemos hacerlo con el programa cliente, que es:

"/bin/MySql.exe".

Para comprobar que funciona correctamente, abrimos una ventana MS-DOS y vamos al directorio donde esté instalado **MySql** (por defecto "C:\MySql\" y dentro del directorio:

"\bin\" ejecutamos "MySql.exe".

Si todo va bien habremos entrado en el DBMS. Nos aparecerá el siguiente mensaje:

Welcome to the MySql monitor. Commands en with; or \g.
Your MySql connection id is 2 to server version: 3.22.34-shareware-debug
Type 'help for help'
MySql>

Podemos ver las bases de datos que tenemos en el sistema escribiendo lo siguiente:

show databases;

Para salir de la sesión deberemos escribir

"exit;".

3.4.4 Instalación de Apache, PHP y MySql en Linux

Lo primero que debemos hacer es conseguirnos los paquetes necesarios, y que mejor para ello que dirigirnos a las páginas Web (o cualquiera de sus espejos) de los programas en cuestión:

- **Apache**: www.apache.org
  - apache-1.3.x.tar.gz
- **MySql**: www.MySql.com
  - MySql-3_22_22_tar.gz
- **PHP**: www.**PHP**.net
  - PHP-3.0.x.tar-gz

Para realizar todo el proceso de instalación debemos tener acceso como **root** a la máquina Linux.

Lo primero que debemos hacer un directorio de instalación, aunque lo normal sería que lo hiciéramos en **/usr/local**, **/urs/src**, o bien en **/opt**. Como hay que escoger uno, escogemos el primero, **/usr/local**, aunque el proceso sería el mismo sí nos declináramos por cualquier otro.

Supongamos que ya nos hemos conseguido los paquetes y los tenemos en el directorio **/root/install**, lo primero que hacemos es descomprimirlos:

```
cd /usr/local
tar zxvf /root/install/apache-1.3.x.tar.gz
tar zxvf /root/instal/MySql-3.22.x.tar.gz
toot/instal/MySql-3.22.x.tar.gz
tar zxvf /root/instal/PHP-3.0.x.tar-gz
```

Creamos enlaces sencillos (blandos) a código fuente:

```
ln -s /usr/local/apache-1.3.x  /usr/local/apache
ln -s /usr/local/MySql-3.22.x /usr/local/MySql
ln -s /usr/local/PHP-3.0.x /usr/local/PHP
```

Preparamos la fuente para la compilación de Apache:

```
cd /usr/loca/apache
./configure --prefix=/usr/local/apache
```

Compilamos e instalamos MySql**:**

```
cd /usr/local/MySql
./configure --witout-debug --prefix=/usr/local/MySql
make
make install
cp /usr/local/support-files/MySql.server /etc/rc.d/init.d/MySql
chmod 755 /etc/rc.d/init.d/MySql
```

Creamos la **Base de datos** del sistema **MySql:**

```
/usr/local/MySql/bin/MySql_install_db
```

Arrancamos el servidor **MySql:**

```
/etc/rc.d/init.d/MySql start/etc/rc.d/init.d/MySql start
```

Asignamos la password del administrador (root) de **MySql:**

```
/usr/local/MySql/bin/MySqladmin -u root password "clave"
```

Ya hemos terminado con **MySql**, ahora compilaremos **PHP** como módulo de Apache:

```
cd /usr/local/PHP
./configure --with-MySql=/usr/local/MySql \
            --with-apache=/usr/local/apache \
            --enable-track-vars
make
make install
#cp PHP3.ini-dist /usr/local/lib/PHP3.ini
```

Compilamos Apache:

```
cd /usr/local/apache
./configure --prefix=/usr/local/apache \
 --activate-module=src/modules/PHP3/libPHP3.a
# si hemos compilado PHP4 utilizaremos
#--activate-module=src/modules/PHP4/libPHP4.a
# quitar los comentarios para habilitar el módulo de proxy
#--activate-module=src/modules/proxy/libproxy.a
make
make install
```

Para definir las extensiones de los scripts **PHP**, hay que añadir las siguientes líneas en el fichero de configuración de apache (httpd.conf):

```
AddType application/x-httpd-PHP3 .PHP3
AddType application/x-httpd-PHP3 .PHP
AddType application/x-httpd-PHP3 .phtml
```

Ahora ya sólo nos queda arrancar el servidor, pero primero copiamos el script de arranque en:

```
/etc/rc.d/init.d

cp /usr/local/apache/bin/apachecte /etc/rc.d/init.d/apache
/etc/rc.d/init.d/apache start
```

Para comprobar nuestra instalación creamos un archivo llamado info.**PHP** con la siguiente línea:

```
<?PHP PHPinfo() ?>
```

Lo colocamos en el directorio de documentos de Apache y lo llamamos desde cualquier navegador. Si lo hemos hecho todo bien nos saldrá una página con todas las variables de **PHP**.

CAPÍTULO 4

Propuesta de Solución

4.1 Metodología utilizada

Una vez analizado el problema y habiendo estudiado las herramientas disponibles, se comienza a estructurar una posible solución, empezando por la metodología a utilizar, tiempos, manejo de datos, sistemas de información, además de otros.

La metodología que se utiliza para el **S**istema de **I**nformación para la **A**signación de tutores en el Área de "**T**utoría para **T**odos" (**SIATT**), es la **metodología en espiral**, por las siguientes razones.

La **metodología en espiral** tiene una naturaleza evolutiva compuesta por ciclos que se van dividiendo en tareas, lo que significa que cada vuelta tiende a mejorar a su antecesor y avanzar en el proyecto (pero puede tener sus excepciones si es que la metodología no se ha adaptado correctamente). Así, el ciclo más interno podría referirse a las formas más básicas, posiblemente a un prototipo en papel.

La segunda iteración podría ser la definición de los primeros objetivos y puntos que al cliente lo lleven a delimitar más su proyecto; las vueltas subsecuentes irán cumpliendo cada uno de los requerimientos del sistema.

Cada una de las vueltas que componen la metodología en espiral se dividen en sectores o tareas que dependen del sistema en particular, ya que varían conforme al tamaño del software, las características del sistema, el personal con que se cuenta, los clientes, los fondos económicos y otros puntos tanto externos, como internos en el sistema.

Las tareas de software deben proporcionar la estrategia para mantener un riguroso nivel de calidad en el sistema, pero sin aumentar con un trabajo innecesario.

En general, se pueden aconsejar las siguientes tareas, ya que la mayoría de los proyectos de software son afectados por los mismos factores,

a) **Conceptos de operación**, es la primera entrevista con el cliente, se definen los detalles más básicos.

b) **Requerimientos del software**, se detallan los objetivos del sistema, tipos de interfaces máquina–usuario, fechas de reuniones, requerimientos de bases de datos, etc.

c) **Diseño de producción de software,** los ingenieros del software estipulan los detalles mismos del proyecto como el sistema operativo, bases de datos, etc., se definen los grupos también se establecen reuniones, asesorias y las formas de comunicación entre el equipo.

d) **Diseño detallado,** aquí se comienzan a generar los modelos de análisis del sistema, que se dividen en cuatro fases:

- **Diseño de datos**, transforma la información en estructuras entendibles para un diseñador, donde las principales fuentes de información son los diagramas entidad–relación y el diccionario de datos.

- **Diseño arquitectónico**, se definen las relaciones entre los módulos de control, esto es, combina la estructura del programa y la estructura de datos definiendo las interfaces que permiten el flujo de datos.

- **Diseño de interfaz**, describe la comunicación del mismo software con los sistemas que operan con él y con la gente que lo va a emplear. La información la obtienen principalmente de los diagramas de estado.

- **Diseño procedimental**, es la definición del programa en un lenguaje que pueda ser entendido por cualquier persona, como las reglas que se establecen para un aprendizaje factible. Las construcciones son: la secuencia, la condición y la repetición, por lo tanto, se basan en gran parte en los diagramas de flujo, porque son muy comunes para todo aquel que se desenvuelva en el desarrollo del software.

e) **Código,** en base al estudio anterior se comienza a estructurar la programación, las funciones, las interfaces, las bases de datos, etc., donde los bloques pueden ir realizándose en paralelo o en serie de acuerdo con el mapa de administración de tiempos realizado.

f) **Prueba unitaria,** a cada una de las tareas de cada subgrupo, se le asignan pruebas de acceso y rendimiento para tener un mayor control de la calidad del sistema.

g) **Integración y pruebas,** todas las particiones hechas del proyecto se juntan y se forma una unidad, se realizan pruebas de calidad y se configuran los detalles que puedan surgir en la unión de las particiones.

h) **Pruebas de aceptación. Implementación**, el proyecto es presentado al cliente, se presentan las pruebas ya hechas (pruebas de integridad, de unidad, etc.) y se espera la aceptación por parte de éste. Se instala en el lugar definitivo de operación, también pueden incluirse asesorías o cursos al personal que va a operar el sistema en la empresa o institución.

Para poder delimitar las regiones de tareas en un proyecto específico (en este caso el **SIATT**), se observan también otros puntos característicos, ya que como se mencionó no se definen para ningún sistema, por ejemplo, las tareas de un proyecto muy grande serían consideradas muy exageradas para un proyecto de menor magnitud y viceversa.

Un punto de los antes mencionados es el **tipo de proyecto** y otro el establecer el **grado de rigor** del proyecto sobre el que se está trabajando.

4.1.2 Determinando grado de rigor y tipo de proyecto

Es difícil determinar a qué **tipo de proyectos** pertenecen los sistemas de ingeniería, por su naturaleza, Pressman[1] menciona que en general, la mayoría de las organizaciones de software encuentran proyectos en las siguientes clasificaciones:

A) Proyectos de desarrollo de concepto. Proyectos que investigan un nuevo concepto, ya sean en el ámbito privado, como industrias farmacéuticas, químicas, militares o dentro de las universidades públicas o privadas.

B) Proyectos de desarrollo de una nueva aplicación. Aquí se encuentran los proyectos que son delimitados por un cliente específico, son los encargos particulares de alguna empresa.

C) Proyectos de mejoras de aplicaciones. Aquellos que se encargan para que sea actualizado o mejorado el sistema, podrían ser las interfaces, el funcionamiento, los rendimientos, etc.

D) Proyecto de mantenimiento de aplicaciones. Son los que corrigen, adaptan o amplían el software, muchas veces sin que los cambios sean vistos por el usuario a corto plazo.

E) Proyectos de reingeniería. Proyectos que hacen una reconstrucción a un sistema cuando ya está en funcionamiento.

El **S**istema de **I**nformación para la **A**signación de tutores en el Área de "**T**utoría para **T**odos" (**SIATT**), esta dentro del grupo **B) Proyectos de desarrollo de una nueva aplicación**, ya que se le considera como un sistema que el Área de "**T**utoría para **T**odos" necesita para tener una mejor Asignación de tutores para los alumnos de la Facultad de Ingeniería. El Área de "**T**utoría para **T**odos" en este caso, es el cliente.

Los **grados de rigor** se definen a partir de todas las especificaciones y estudios preliminares del sistema y es la forma en cómo se debe tratar el proyecto, ya sea con un mayor rigor y exactitud en los lineamientos o no.

Esto puede traer algunos puntos de discusión, ya que independientemente del proyecto, ***todos*** los proyectos en sí, tienen una gran importancia y se deben apegar a lo establecido en los objetivos iniciales, así como respetar los tiempos y la entrega del sistema con la más alta calidad de desarrollo; pero la importancia entre los sistemas puede diferir grandemente entre los que se estén trabajando.

Los grados de rigor se pueden clasificar en:

1) Casual, son proyectos pequeños en donde las actividades de pruebas se minimizan y son considerados pequeños desarrollos de software. Independientemente de su tamaño llevan todos los procesos de la ingeniería del software. La documentación es mínima.

2) Estructurado, es más detallado que el anterior, llevan un proceso de calidad más sofisticado, pruebas específicas y una documentación más amplia. El número de tareas aumentan considerablemente con respecto al anterior, pero no sobrepasa a proyectos que están catalogados en la siguiente fase.

3) Estricto, aquí se aplican todas las medidas de seguridad para la obtención de un sistema de calidad, todo el proceso completo con la más alta disciplina, grados de error, análisis de riesgos, planes de contingencia, etc.

Existe una cuarta clasificación, pero siguiendo una estructura lógica de las anteriores propuestas, no tiene mucho que ver directamente con éstas y en algún momento podría clasificarse con un grado de rigor casual; por lo tanto, se menciona la cuarta clasificación dando pie a lo citado.

4) De reacción rápida, se pueden catalogar como métodos de bomberos porque se realizan en tiempos muy cortos, la documentación es entregada después del proyecto y las pruebas muchas veces se realizan conforme ya trabaja el sistema.

Para definir el grado de rigor se toman en cuenta puntos ya delimitados, llamados criterios de adaptación y un valor que **Pressman** define como el valor Selector del Conjunto de Tareas (**SCT**).

Entre los criterios de adaptación se pueden definir once puntos:

1. Tamaño del proyecto
2. Número potencial de usuarios
3. Misión crítica
4. Longevidad de la aplicación
5. Estabilidad de los requisitos
6. Facilidad de comunicación con el cliente o el desarrollador
7. Madurez de la tecnología aplicable
8. Limitaciones de rendimiento
9. Características ya establecidas o no
10. Personal del proyecto
11. Factores de reingeniería

A cada uno de los puntos anteriores se le asigna un número, que es el grado que se aplica según se necesite de éstos, por ejemplo, el personal del proyecto, si es mínimo se les da el valor de **uno**, en caso contrario se le asigna un valor de **cinco**.

Las operaciones y pasos para obtener el valor **SCT** se muestran en la **Tabla 4a** y son los siguientes:

1. Después de asignar los valores a los criterios de adaptación (del 1 al 5), se multiplican por los factores de ponderación, que van de 0.8 a 1.2. Los **valores de ponderación** son una indicación de la relativa importancia de un criterio de adaptación a los tipos de software desarrollados dentro del entorno de la empresa, (los valores que aquí fueron tomados son un ejemplo de varios proyectos, pero basados principalmente en los valores que estipula Pressman[1]).

2. El resultado se vuelve a multiplicar por el *multiplicador de punto de entrada*, que depende del tipo de proyecto a realizar, como se definió que es un proyecto para el desarrollo de una nueva aplicación, entonces se multiplica por los valores delimitados en la estructura del **SCT**.

3. Se calcula la media de todas las entradas en la columna *Producto* y éste es el valor **SCT**.

4. De acuerdo con el **SCT** y la **Tabla 4b**, se delimita el grado de rigor del proyecto.

Criterio de Adaptación	**Grado**	**Peso**	**Multiplicador de punto de Entrada Desarrollo de una Nueva Aplicación**	**Producto**
Tamaño del Proyecto	4	1.2	1	4.8
Número de Usuarios	2	1.1	1	2.2
Importancia para el Negocio	4	1.1	1	4.4
Longevidad	4	0.9	1	3.6
Estabilidad de los Requisitos	5	1.2	1	6
Facilidad de Comunicación	4	0.9	1	3.6
Madurez de la Tecnología	3	0.9	1	2.7

Limitaciones de Rendimiento	4	0.8	1	3.2
Empotrado/No empotrado	4	1.2	1	4.8
Personal del Proyecto	2	1	1	2
Interoperatividad	4	1.1	1	4.4
Factores de Reingeniería	2	1.2	0	2.4
Selector de Conjunto de Tareas (**SCT**) 3.67				

Tabla 4a Cálculo del Selector de Conjunto de Tareas

Valor del selector del conjunto de tareas	Grado de Rigor
SCT < 1.2	Casual
1.0 < SCT < 3.0	Estructurado
SCT > 3.0	Estricto

Tabla 4b Delimitación de los grados de rigor.

El valor resultante es **3.67** y basándonos en la **Tabla 4b**, se deduce que el proyecto tiene un grado de rigor **Estricto**.

4.1.3 Regiones de tareas

Al momento de seleccionar las regiones de tareas de la metodología en espiral para nuestro proyecto, se toma en cuenta los grados de rigor, necesidades, tipo de proyectos y las propuestas descritas anteriormente. Por lo tanto, para el proyecto del **SIATT**, se definen las siguientes tareas:

1. La primera tarea es la **comunicación con el cliente**

   Se establece la comunicación con el jefe del proyecto, investigadores y gente de cómputo; se delimitan alcances, opciones, requisitos y todo lo referente para realizar el análisis preliminar del proyecto. Es el ámbito del concepto.

2. La segunda tarea es **planificación y análisis de riesgos**

   Se analizan los recursos que se establecieron en la etapa anterior y basándose en estos datos, se realiza el estudio del análisis de riesgo, estudio de tiempos y se comienzan los primeros pasos del diseño del software que son la estructura de datos.

3. La tercera tarea es **desarrollo-ingeniería o prueba del concepto**

   Se define la propuesta de solución a utilizar con base en los análisis y diagramas de la etapa anterior. Generando el modelado de datos, diagramas de procesos jerárquicos, flujo de datos, diagramas entidad–relación, diccionario de datos, etc., y se asignan tareas de acuerdo con las fechas ya establecidas.

4. La cuarta tarea es **construcción y adaptación**

   Es la realización tangible de lo que se ha hecho en papel, se incluyen las etapas de pruebas de calidad y rendimiento. Se comienza a realizar la programación de cada una de las estructuras y la unión de éstas si es que fueron diversas etapas. Se prueba e instala el prototipo.

5. La quinta etapa es **la evaluación con el cliente**

   Se busca la opinión final del jefe de proyecto, se le entregan las pruebas realizadas (pruebas de carga, pruebas de punto de foco, prueba beta, checar las condiciones de errores, de unidad y de función) y la documentación, dictamina si el proyecto termina o se establecen los lineamientos para una nueva iteración en la espira.

La figura 4.1 muestra el esquema del diagrama en espiral del **SIATT**.

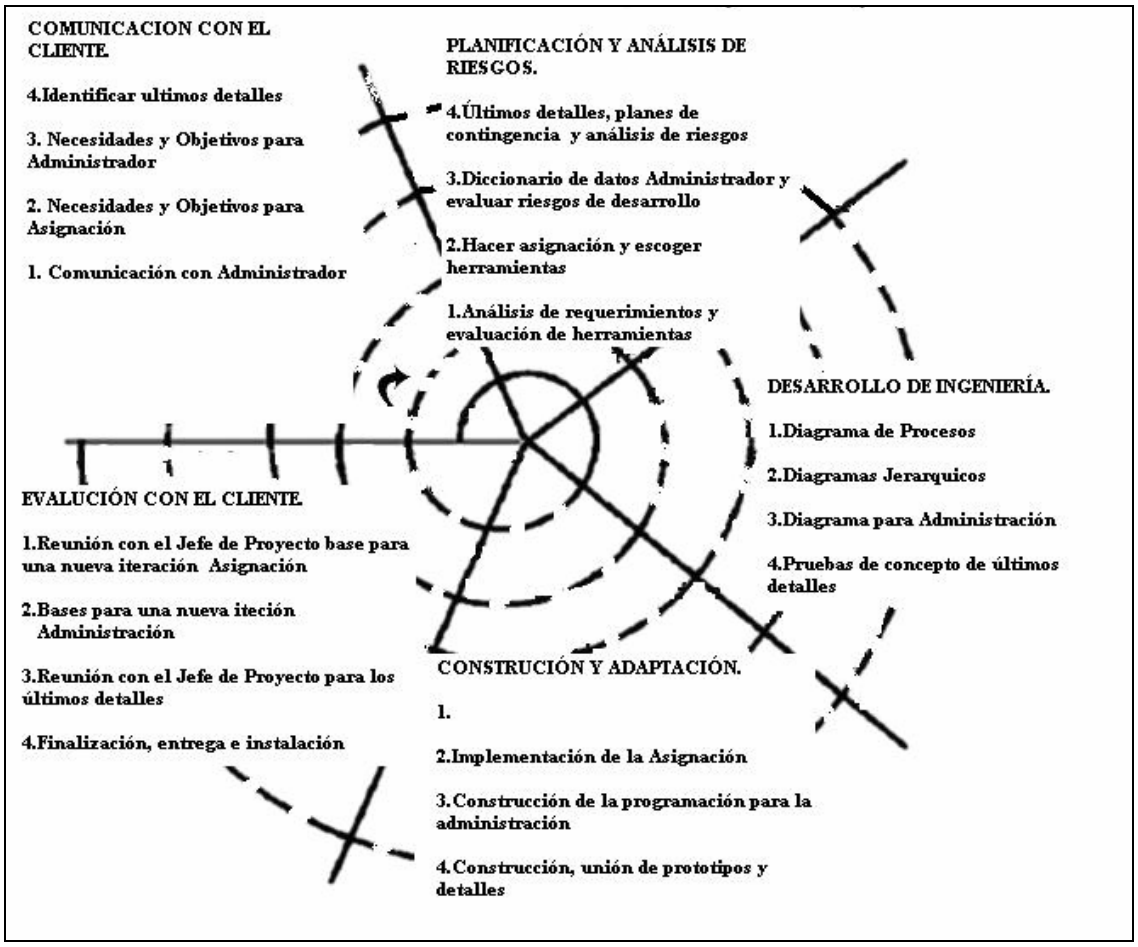


Figura 4.1 Modelo espiral del SIATT

4.2 Estudio de tiempos

Es de gran importancia el establecer los tiempos en cualquier sistema, lo que conlleva a que se maneje una calendarización del inicio y del término del proyecto. No es fácil delimitar el tiempo de realización de un sistema, por lo que el ingeniero que establece los **tiempos** debe basarse en los estudios de la ingeniería del software y ayudarse con la experiencia previa.

Los diagramas de **PERT** y **Gantt** proporcionan una manera de organizar los tiempos del proyecto, además de ayudar a establecer las rutas críticas; pero hay que recordar que las gráficas son una estimación del proyecto en general y por lo tanto, pueden tener variaciones que deben ser previstas por medio de tiempos de respaldo, en el caso del **SIATT**, por los desarrolladores del sistema.

El diagrama de **Gantt** se basa en el conjunto de tareas que son delimitadas por la metodología a seguir, en este caso, la metodología en espiral. La duración y la fecha de inicio son las entradas de cada tarea. Cada región de tarea de cada iteración que se realice debe ocupar un espacio en el esquema general.

En el diagrama de **Gantt** se observan fácilmente si las actividades pueden ser llevadas en una forma paralela y cuáles de ellas tienen que ser consecutivas.

Cuando el proyecto es realizado por varias personas, el paralelismo puede resultar muy práctico además de indispensable, ya que todo el personal no puede realizar la misma actividad a un mismo tiempo. Se puede trabajar en procesos paralelos, el objetivo final (disminución de tiempos) se debe lograr.

Los diagramas de **Gantt** utilizan barras con diferentes longitudes para representar la duración de cada una de sus actividades, además que muestran un colchón de tiempo para cada una de las etapas por cualquier inconveniente que pueda surgir en el transcurso de trabajo.

El objetivo de los sistemas tipo **PERT** consiste en ayudar en la planeación y el control, por lo que no implica mucha optimización directa. Algunas veces el objetivo primario es determinar la probabilidad de cumplir con fechas de entrega específicas.

También identifica aquellas actividades que son más probables que se conviertan en cuellos de botella y señala, por ende, en que puntos debe hacerse el mayor esfuerzo para no tener retrasos.

Un tercer objetivo es evaluar el efecto de los cambios del programa. Por ejemplo, se puede valorar el efecto de un posible cambio en la asignación de recursos de las actividades menos críticas a aquellas que se identificaron con cuellos de botella.

Otra aplicación importante es la evaluación del efecto de desviarse de lo programado. Todos los sistemas tipo **PERT** emplean una red de proyecto para visualizar gráficamente la interrelación entre sus elementos. Esta representación del plan de un proyecto muestra todas las relaciones de procedencia, respecto al orden en que se deben realizar las actividades.

En la figura 4.2 sé muestran estas características para la red de proyecto para la construcción del SIATT.

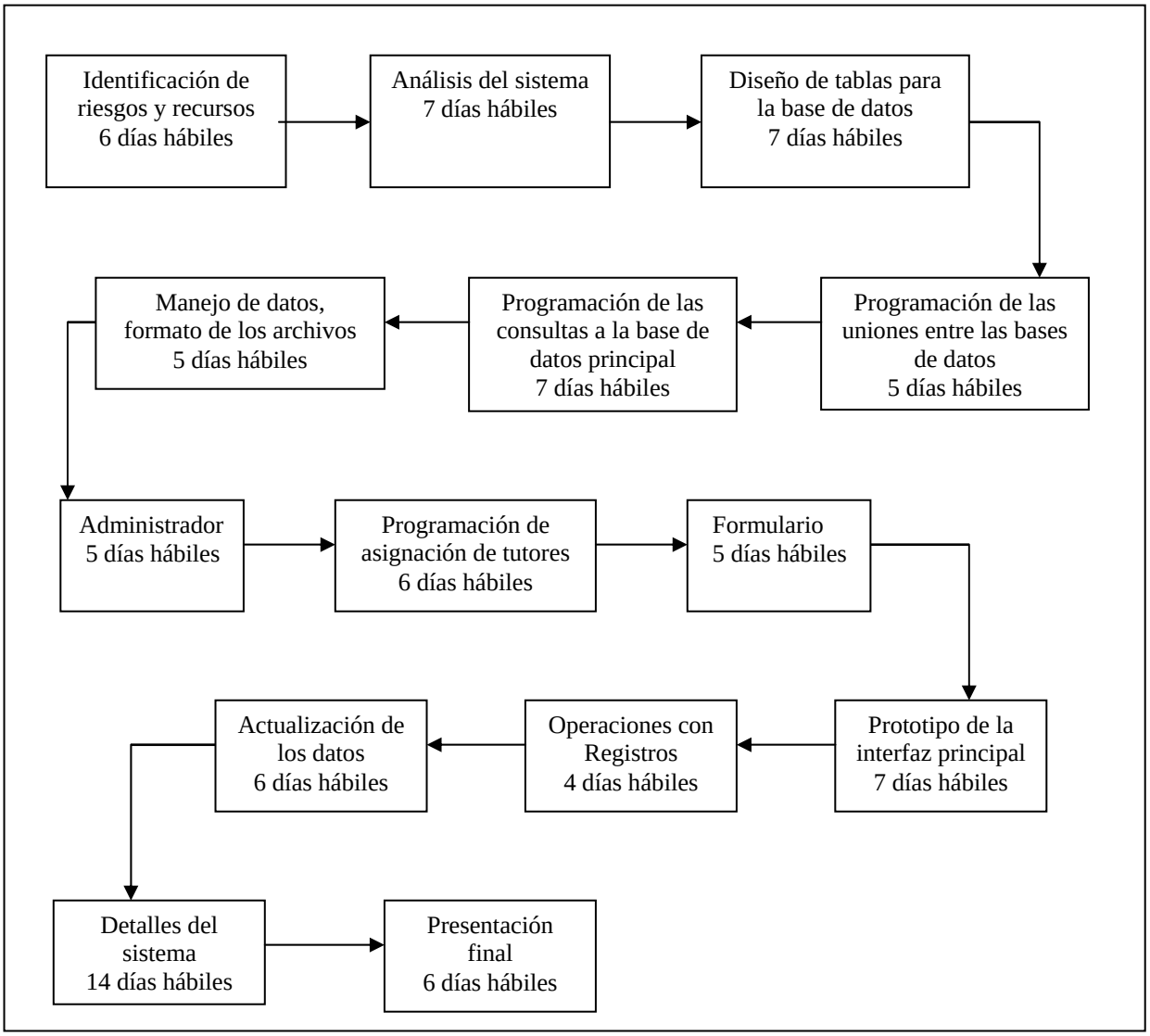


Figura 4.2 Diagrama PERT del SIATT

4.3 Modelado del sistema

En el análisis del sistema se desarrollan los modelos generales del proyecto, es decir, se desarrolla un modelo del funcionamiento del sistema que se expresa en términos de procesos, relaciones, en el control de flujos y en las transformaciones funcionales.

Para el análisis se realizan tres tipos de modelos, **el modelo de objetos, el modelo dinámico y el modelo funcional.**

4.3.1 Modelo de objetos

El primer modelo de objetos va a describir una estructura estática de los objetos del sistema y las relaciones que existen entre éstos, principalmente utiliza el diagrama de objetos, que se identifica por el estudio de clases, sus atributos y las operaciones que existen entre ellos.

Los elementos estructurales para el modelo de objetos son cuatro:

a) **Objeto**, es la representación de cualquier composición de información, la abstracción, un concepto bien delimitado que tiene un significado determinado para el sistema. Todos los objetos tienen una identidad y son distinguibles.

b) **Clases**, es un grupo de objetos, con atributos, características y operaciones propias.

c) **Operaciones**, las operaciones son acciones que pueden realizar las clases.

d) **Atributos**, definen las propiedades de los objetos de una clase.

También se encuentran las **asociaciones,** que establecen las relaciones entre las clases, pero éstas no están catalogadas como elementos estructurales.

La notación empleada para representar gráficamente las estructuras del modelo de objetos se muestra en la figura 4.3.

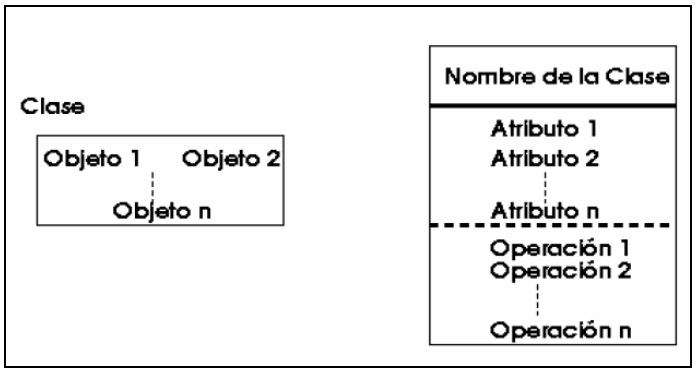


Figura 4.3 Objetos, Clases, Atributos y Operaciones

Para el diagrama de objetos del **SIATT**, se tienen que identificar las clases, las asociaciones entre las clases y los objetos que la componen, la figura. 4.4 muestra el modelo de objetos del **SIATT**.

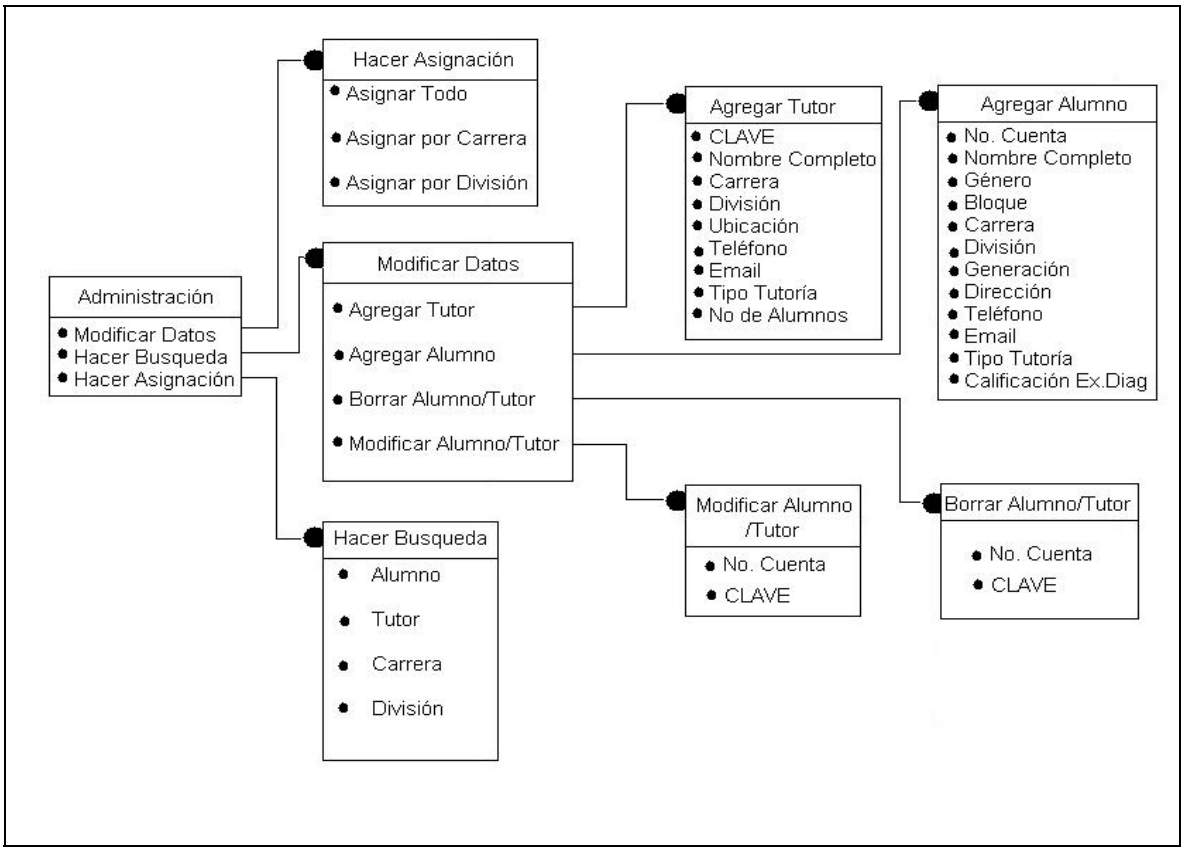


Figura 4.4 Diagrama de Objetos

4.3.2 Modelo dinámico

El segundo modelo para analizar es el **modelo dinámico**, donde supone que en cualquier momento el sistema puede encontrarse en un estado diferente y sólo cuando éste recibe una entrada el sistema se transfiere a otro, describe los aspectos conforme pasa el tiempo y específica e implanta los aspectos de control del sistema; su principal herramienta consiste en los diagramas de estado que se conforman por **estados, transiciones y eventos.**

En los diagramas de estado no se manejan flujos de datos ni variables de transferencia, sino que se manejan elementos para las transiciones entre las etapas; para realizarlo se tienen que identificar cada uno de los **estados** del proceso y los argumentos para moverse entre sí (**ligas o transiciones**), así como los **eventos** que ocasionan que se disparen estos argumentos.

En algunos proyectos si se pueden observar los estados iniciales y los estados finales (que se les denominan "de un solo ciclo de vida"), pero en la mayoría de los sistemas no se observan estos estados.

La figura 4.5 muestra el diagrama de estado para el **SIATT**, donde se observan los bloques que lo componen y la descripción de cada uno de ellos se encuentran en la **Tabla 4d.** Cabe mencionar que estos bloques adquieren una gran importancia cuando se genera el diccionario de datos, en nuestro caso, más adelante.

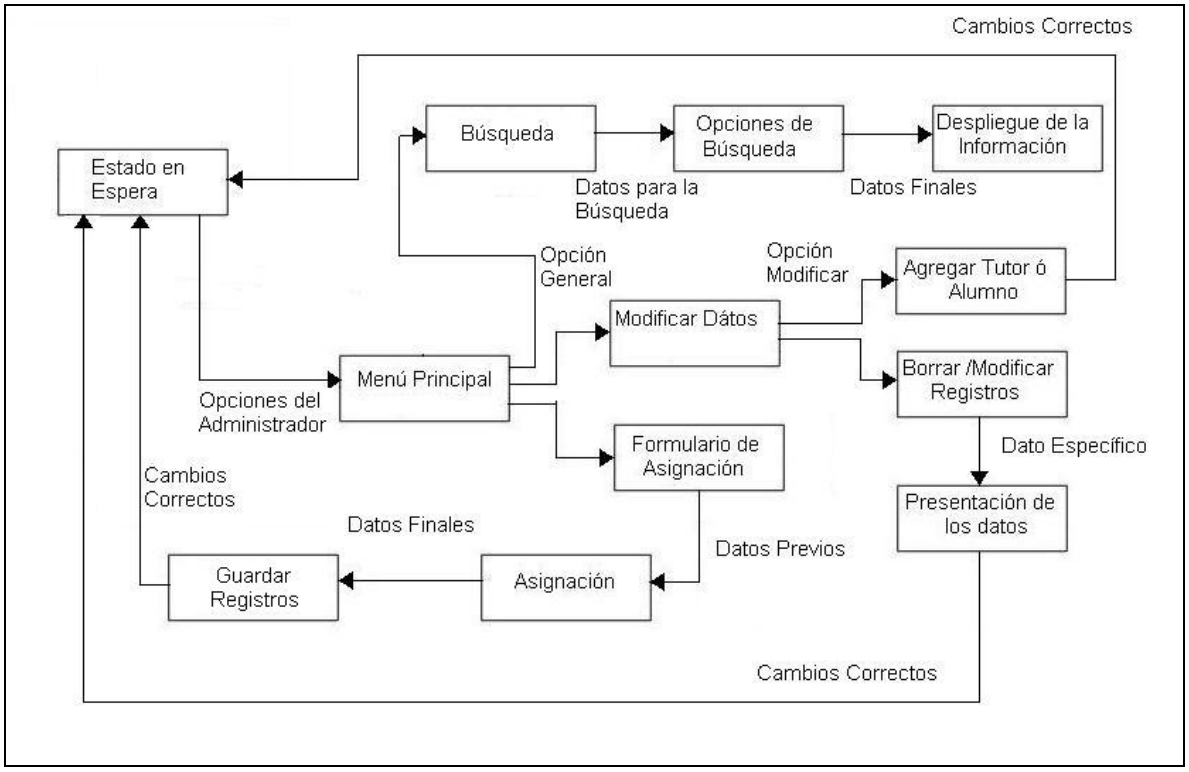


Figura 4.5 Diagrama de Estado

Estado	Descripción
En Espera	Se espera la entrada inicial que delimite el siguiente camino.
Menú Principal	El estado muestra todas las acciones que se pueden llevar a cabo en el sistema.
Formulario de Asignación	Se despliega una interfaz que ofrece criterios de Asignación.
Asignación	Se llevan a cabo las consultas de tablas, uniones y operaciones que en conjunto forman la asignación.
Guardar Registros	Se guarda la asignación (Tabla generada) en un archivo, en un directorio especificado.
Modificar Datos	Este estado despliega las opciones que tiene el administrador, para agregar, borrar o modificar datos de alumno o tutor.
Agregar tutor o alumno	Crea un registro de tutor o alumno en la base de datos.
Borrar/Modificar Registros	Se elimina o se edita un registro.
Presentación de los datos	Se despliega una consulta del registro anterior para cerciorarse de que los cambios son correctos.
Búsqueda	El estado muestra las alternativas para hacer una búsqueda de tutor o alumno.
Opciones de Búsqueda	Se ofrecen diferentes formas de hacer la búsqueda y se da la opción de buscar a una persona por su Clave general.
Despliegue de información	Se muestra la información sobre la persona o características particulares.

Tabla 4d Descripción de los Estados

4.3.3 Modelo funcional

El tercer y último paso es el modelo funcional, que describe las transformaciones de los valores dentro del sistema. Su principal herramienta son los diagramas de flujo de datos, con los que se pueden representar cálculos, entradas, flujos de datos y archivos para almacenar datos.

Los diagramas de flujos de datos pueden ser representados en diferentes niveles de abstracción que van tomando valores que comienzan del cero, que son los diagramas más generales, hasta los más específicos, donde pueden llegar hasta la descripción o acceso de una sola variable.

Los elementos que los construyen son los procesos, flujos de datos, almacén de datos y los terminadores, que describiremos más adelante dada la estructura del capítulo, además de los diagramas de flujos del sistema.

CAPÍTULO 5

Diseño de la aplicación

5.1 Diseño de la aplicación

En el capítulo anterior (propuesta de solución) ya se ha empezado a forjar el diseño de la aplicación, porque el análisis de los procesos y la estructura principal del sistema ya se han desarrollado; en este capítulo se comienzan a manejar los datos, el flujo de información y las estructuras jerárquicas.

El diseño de los datos es una parte importante del diseño de cualquier aplicación, el mantener una idea que pueda ser entendida por cualquier persona que accede al proyecto es un tanto difícil si no se siguen ciertas reglas ya estructuradas. Para estas reglas, algunos estudios delimitan que para realizar un diseño del sistema se recomiendan seguir los siguientes pasos:

a) Organizar el sistema en **subsistemas**.

b) Identificar **concurrencia** inherente al problema.

c) Asignar subsistemas a **procesadores y tareas**.

d) Escoger una **estrategia básica para implementar los almacenamientos** en términos de estructura de datos, archivos y bases de datos.

e) Determinar los mecanismos para controlar el **acceso a recursos globales**.

f) Escoger la implementación del **control del software**.

g) Considerar las condiciones de **frontera**.

h) Establecer **prioridades de decisión sobre características** del producto **de software**.

El diseño se puede documentar y estructurar como un conjunto de modelos gráficos, entre los que se cuentan el **modelo entidad-relación**, los **diagramas de flujo de datos**, **diagramas de construcción** y los **diagramas jerárquicos**.

Para llegar a realizar el **diagrama de flujo**, tendremos que hacer uso del diseño arquitectónico, este desarrolla un mayor nivel de abstracción en el sistema. Este diseño realiza un modelo por bloques, las conexiones entre éstos y todas las estructuras del sistema, Pressman lo define como el diseño arquitectónico... *"desarrolla una estructura de programa modular y representa las relaciones de control entre los módulos"*.

Además, el diseño arquitectónico combina la estructura del programa y la estructura de los datos, definiendo interfaces que permiten el flujo de datos a través del programa.

Para encausar la información se van delimitando las estructuras antes dichas, al principio son esquemas muy generales en los que se pueden observar los subsistemas o componentes del proyecto.

Después se van delimitando los flujos de datos entre las estructuras y el intercambio de información que requieren cada uno de éstos, también se delimitan las interfaces y los datos de entrada y de salida, donde la estructura mínima de la que se componen son de **tres pasos**:

- **Entrada**
- **Procesamiento**
- **Salida**

Por lo tanto, un esqueleto puede tener varios niveles muy detallados o superfluos de los datos en la estructura general y dada la claridad o complejidad de éstos, la calidad del sistema se mueve incondicionalmente.

Los principales objetivos para cualquier modelo se pueden resumir en **dos puntos**:

A. Proveer un modelo preciso de las necesidades funcionales de la organización que actuarán como marco referencial para el desarrollo o actualización del sistema.

B. Dar un modelo que sea independiente de cualquier mecanismo o método de procesamiento y que permita la toma de decisiones certeras a cerca de técnicas de implementación y acoplamiento con sistemas existentes.

Los elementos para generar un modelo principalmente **son cuatro**:

- **El proceso**
- **El flujo de datos**
- **El almacén**
- **Los actores**

a) Proceso

Un proceso representa el bloque donde se efectúa **la transformación del valor de los datos**, por lo tanto, se habla de una entrada, un proceso y una salida. La notación se muestra en la figura 5.1.

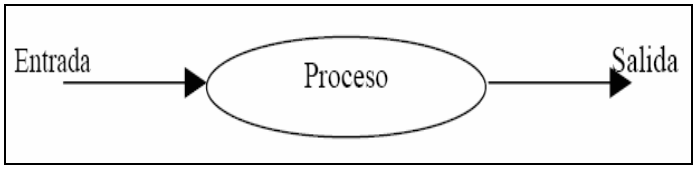


Figura 5.1 Esquema de un proceso

b) Flujo de datos

Un flujo de datos **conecta la salida de un proceso con la entrada de otro**, esto es, agrupa los valores de datos entre objetos dentro de un camino, es la tubería del sistema y la notación se representa en la figura 5.2.

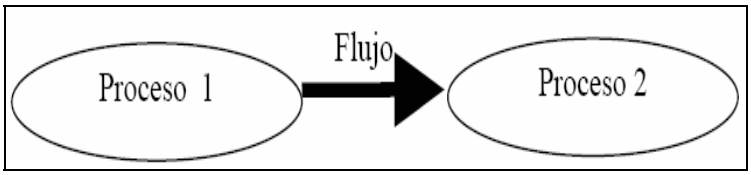


Figura 5.2 Esquema de Flujo de datos

c) Actores

El actor es un objeto activo que conduce el grafo de flujo de datos, ya sea consumiendo o produciendo valores. Los actores **están asociados con las entradas y salidas de flujo de datos en las fronteras de un diagrama**, por ello, algunas veces se llaman **terminadores**.

Por lo regular se le asocia con una persona o un grupo, pero también puede ser otro sistema, como algún otro hardware o software con el que se comunique.

La figura 5.3 muestra la representación de los actores empezando y terminando un flujo de datos.

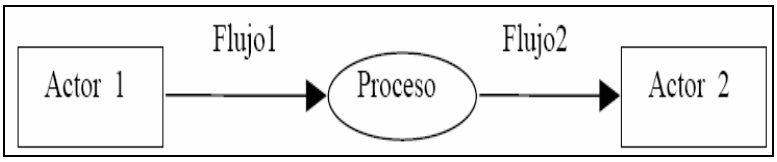


Figura 5.3 Esquema de actores

d) Almacenamiento de datos

El almacenamiento de datos es un objeto pasivo, ya que no modifica ningún dato en toda la estructura. Como su nombre lo indica **sirven para guardar registros**, ya sea como datos finales o como datos temporales, los primeros son aquellos que no van a tener modificación alguna al ser llamados por el sistema, no así los datos temporales que sí serán modificados.

La notación para la representación son dos líneas horizontales y se muestran en la figura. 5.4.



Figura 5.4 Almacenamiento de datos

5.2 Diagrama de flujo de datos

Un diagrama completo de flujo de datos representa un proceso de alto nivel, esto es, una estructura muy general y abstracta (denominado nivel 0), pero los procesos en el diagrama pueden ser detallados cada vez más, logrando procesos de alto nivel, un diagrama de *nivel n* (donde *n* es mayor que 2) puede llegar sólo a detallar la descripción de una variable.

El diseño a un nivel 0 da un esquema sobre cada uno de los procesos que recorre un dato como un objeto, cuál es su principio y su fin, en dónde son guardados y de dónde se extraen, así como algunas de las modificaciones que tengan en el proceso de transferencia o los nuevos flujos que vayan surgiendo.

Además de observar el camino de cada proceso y los cambios que les vayan ocurriendo.

La visión global del sistema, el no involucrar demasiados datos y ver una estructura alterna es muy útil para las personas que llegan a involucrarse por primera vez en el sistema.

En la figura 5.6 se observa el **diagrama de flujo** a nivel 0 del **SIATT** se observa un menor detalle en las variables y bloques con respecto a lo que sería un nivel 1 ó 2.

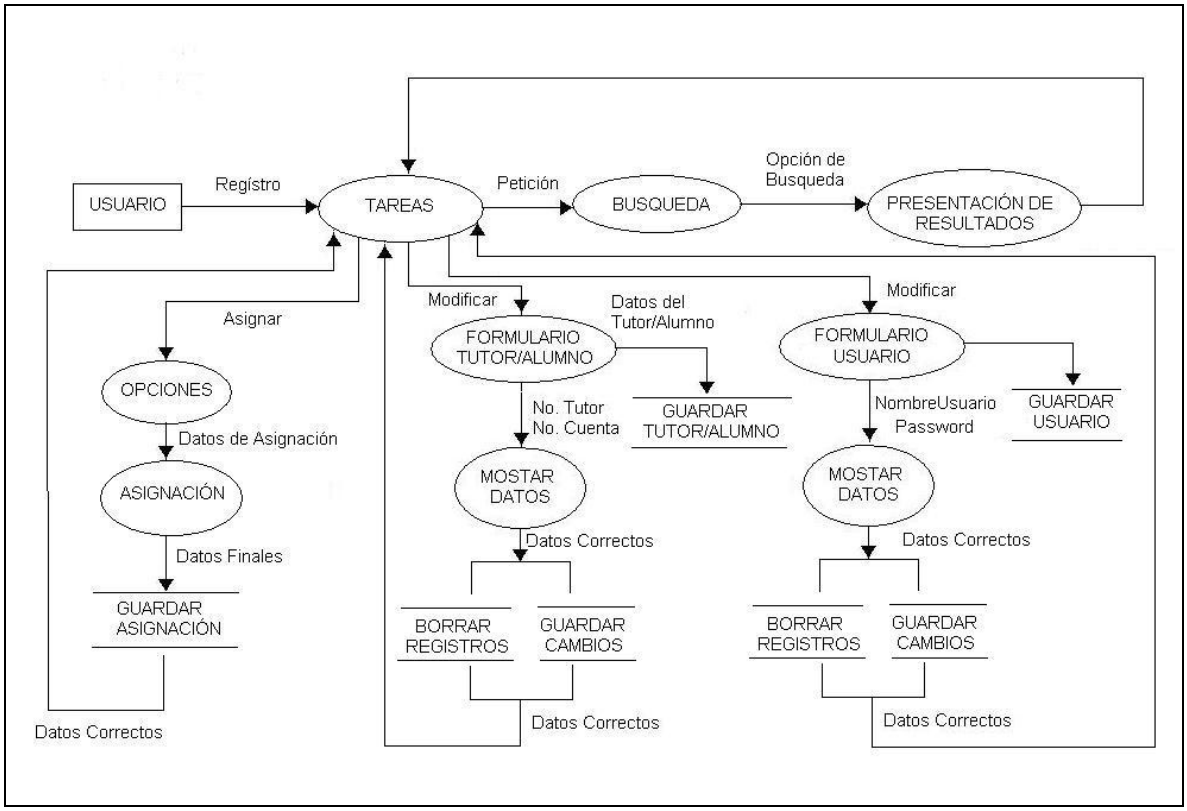


Figura 5.5 Diagrama de Flujo del SIATT, Nivel 0

5.3 Diseño de la interfaz

En el desarrollo de los sistemas de software, se ha tenido la necesidad de adaptar su uso para personas ajenas al ambiente de la programación. Así surgen los métodos de interacción máquina–usuario cada vez más fáciles, rápidos y prácticos, basándose principalmente en ventanas.

Pero el diseño de las interfaces se condiciona a la memoria de proceso y almacenamiento disponible, ancho de banda, la velocidad de descarga, etc., por lo que se inicia la construcción de las mismas a partir de elementos ligeros, de tal forma que el proceso de descarga y visualización de una interfaz gráfica no se convierta en algo estresante para el usuario.

Existen dos tipos de interfaces de datos: El **diseño de interfaz intermodular** y El **diseño de interfaz externo**. En el primero se trabajan los datos dentro de cada módulo y entre ellos, en el segundo los datos que necesita y que no puede generarlos por sí mismo, como pueden ser pulsos, emisión de luz, alguna conexión a un puerto paralelo o datos que el mismo usuario proporciona.

Para obtener la **interfaz intermodular** es muy útil el **análisis del sistema**, ya que se observa el flujo de datos y la transformación de los datos en la transferencia entre los módulos, pero el usuario final directamente no tiene acceso a esta sección, por lo que el diseño es orientado a los ingenieros del sistema, no así en la interfaz externa que principalmente nos enfocaremos a continuación.

Para la **interfaz externa** se necesita cumplir con ciertos requisitos para mantener una excelente calidad, ya que en sí, esto es lo que evalúa el cliente. El usuario no se fija si un dato X se movió de un módulo a otro en vez de tomar otro camino o si el dato se convirtió en flotante u otra cuestión que no necesite, pero sí toma en cuenta lo rápido, fácil, útil y hasta “bonito” que le resulte el sistema.

El estudio debe comenzar analizando el tipo de usuario que utiliza el sistema, cómo maneja la información, qué es lo que espera del sistema, etc., además de investigar sus perfiles, edad, sexo, capacidades físicas, estudios, historial cultural o étnico, motivación, metas y personalidad.

En el estudio anterior se comienza a estructurar un modelo de diseño que irá refinando el cliente, sin olvidar que este proceso se debe apegar a la estructura general del sistema, esto es, tiene que seguir la metodología tomada para todo el sistema (para el SIAT es el modelo en espiral).

Principalmente los aspectos que toma el diseño son el **tiempo de respuesta**, **la ayuda** y el **manejo de la información**.

El **tiempo de respuesta**, es aquel intervalo en el que el sistema responde después de que se suministra un dato de entrada, el cual debe tener un tiempo corto, ya que puede producir frustración y enojo en el usuario, aunque hay autores que afirman que el tiempo no debe ser tan corto porque si no, la interfaz podría apresurar al usuario.

La figura 5.6 muestra la principal interfaz del **SIATT** para el usuario que ha sido aceptado, después de introducir su contraseña.

Observar que el **menú principal**, presenta de forma clara y descriptiva las tareas más importantes del sistema.



Figura 5.6 Interfaz principal del SIATT

Para una mejor visualización, algunos puntos que se recomiendan seguir son:

a) **Mostrar la información que sea relevante en el contexto actual**. El sistema no debe mostrar gráficas, datos o interfaces que no estén directamente relacionadas con la información pedida.

b) **No abrumar al usuario con datos, utilizar un formato de visualización que le permita una rápida asimilación de la información**, al usuario no se le presentan grandes marcos y cuadros, sólo información puntual.

c) **Usar etiquetas consistentes, abreviaciones estándar y colores predecibles**, la mayoría de las opciones de trabajo tienen un formato estándar y se recomienda tener un uso similar.

d) **Estudiar la "geografía" disponible en la pantalla**. Cuando hay múltiples ventanas, mínimo se tiene que ver una porción de éstas conteniendo los datos requeridos.

La **entrada de datos** para el **SIATT** adquiere gran importancia en la sección de **Asignación** ya que ésta es la principal función del sistema.

La figura 5.7 muestra los cuatro pasos para realizar la asignación y por lo tanto, las entradas de información al sistema.

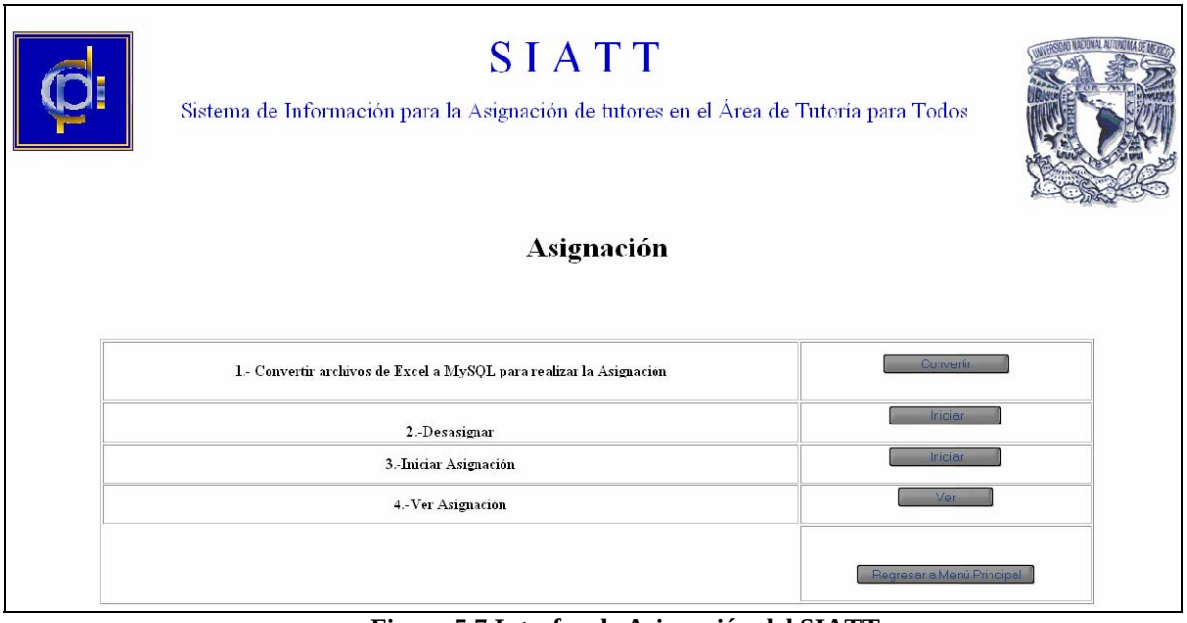


Figura 5.7 Interfaz de Asignación del SIATT

5.4 Diseño de la base de datos

El diseño de las bases de datos ha tomado una gran importancia a partir de las exigencias de los usuarios respecto a sus sistemas de información para que cada vez sean más flexibles, eficientes y accesibles.

En los últimos años gracias al avance tecnológico y a la gran difusión de los sistemas de gestión para las bases de datos cuyos productos ahora ya pueden ser soportados por grandes y pequeños (supercomputadoras y computadoras personales), el camino se ha hecho más fácil, pero aún así, todavía existe un largo camino.

Similar al diseño del proyecto, en las bases de datos también se siguen pasos ordenados y delimitados dentro del marco de una metodología y se manejan tres fases en este proceso:

- **Diseño de la base de datos conceptual**. Es aquí donde se obtiene una representación de lo que el cliente necesita, objetivos primarios y secundarios, restricciones, manejo de la información, en general todo lo que conlleva al análisis del sistema. No se hace un gran hincapié a los tipos de perfiles de los usuarios ni a la plataforma de trabajo, ya que son sólo las delimitaciones generales.

- **Diseño de la base de datos lógica**. En este bloque se transforma el esquema conceptual de la etapa anterior al modelo de datos que se apoya en el sistema gestor. Se puede adaptar la etapa (*diseño lógico*) a otros modelos de datos como el relacional o jerárquico.

- **Diseño de la base de datos física**. Aquí es donde se delimita el tipo de programación y se consigue la instrumentación necesaria.

5.5 Diseño de la base de datos conceptual

La importancia del modelo de datos en el diseño de bases, es la representación de un modelo conceptual que recoja la semántica del mundo real. Un modelo entidad–relacional sencillo pero completo que conlleve la participación tanto de los ingenieros del proyecto como los clientes es lo indispensable para el comienzo del proyecto.

Un **modelo de datos** es una serie de conceptos que pueden utilizarse para describir un conjunto de registros y las operaciones para manipularlos. Son la base conceptual para el diseño de aplicaciones que hacen un uso intensivo de los datos, así como la base para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información.

Estos modelos se dividen en **modelos conceptuales y modelos lógicos**. Los primeros manejan una mayor abstracción para presentar la realidad con respecto a los conceptuales que tienen una forma mucho más sencilla en la estructura física de las bases de datos.

Por lo tanto, en el diseño de bases de datos se utilizan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad y después, se transforma el esquema conceptual en un esquema lógico, esto ayuda a disminuir el grado de dificultad con el que se observa a grandes rasgos, todo el proyecto.

Los modelos de datos se clasifican en:

a) Modelos lógicos basados en objetos

b) Modelos lógicos basados en registros

c) Modelos físicos de datos

En los **modelos lógicos basados en objetos** se tiene una representación de datos muy flexible y muy fácil de entender, nos muestra los datos como los captamos del mundo real.

El más representativo es el **modelo entidad-relación**, que nos muestra los formatos por medio de entidades, es el más sencillo y más adelante se detalla.

Los **modelos lógicos basados en registros** son principalmente usados para describir los datos en los niveles conceptuales y físicos. Utilizan también registros, instancias y ligas, pero estos modelos basados en registros se usan para especificar una estructura lógica global de la base de datos y para proporcionar una descripción a un nivel más alto de la implementación.

Los tres principales modelos basados en registros de datos son el modelo **relacional**, el modelo de **red** y el modelo **jerárquico**.

Los **modelos físicos de datos**, son utilizados para la descripción de los datos más bajos y sólo se manejan el modelo unificador y la memoria de elementos.

5.6 Modelo entidad-relación

El modelo entidad-relación forma parte del modelo de datos y es la estructura que delimita el enfoque global del sistema; es el más importante para la determinación de la estructura de las bases de datos.

Este modelo es usado como un soporte unificado de los datos que sintetiza el mundo exterior con un lenguaje natural: en relaciones y en entidades. Las **entidades** son objetos existentes que poseen ciertas características que los definen entre los demás, como son los **atributos** *y* las relaciones e indicadores de tipo de dato. Las **relaciones** son las definiciones que enlazan a las entidades.

El diagrama entidad-relación, está compuesto por entidades, que son representados por un **rectángulo**; los atributos de las entidades se representan por una **elipse**; la etiqueta dentro de un **rombo** nos indica la relación que existe entre las entidades destacando con líneas sus uniones y la llave primaria de una entidad es aquel atributo que se encuentra subrayado.

Toda la teoría del modelo entidad relación ya se explicó en el Capítulo 2, ahora nos enfocamos a como se realizó el modelo Entidad Relación del **SIATT**. A continuación en las figuras 5.8 y 5.9 se describe gráficamente como quedaron constituidas las entidades ALUMNO y TUTOR respectivamente.

Las Entidades (Tablas) **ALUMNO** y **TUTOR**, vienen dadas de la transformación a **MySQL** de un archivo de hoja de cálculo (**Excel**) para cada Entidad, con un formato dado por la Facultad de Ingeniería y por el administrador del sistema, entendiéndose por formato a las columnas (atributos) que contiene cada Entidad.

Debido a esto puede observarse que cada Entidad, tiene demasiados Atributos que la describen, y en algunos casos repetidos pero con diferente nombre.

Sabemos que esto no debería ser así por lo que respecta a la teoría de Base de Datos, pero se nos especifico que cuando la información de las tablas se desplegara en una consulta, lo hiciera exactamente como aparecen en el archivo original en **Excel**, con todo y columnas repetidas pero con diferente nombre.

Por ejemplo en la tabla **TUTOR** aparece nombre y nombre_ap, el primero es el nombre completo empezando por el nombre de pila y el segundo es el nombre completo pero empezando por apellido paterno.

Observar que tanto Entidades como Atributos, no pueden llevar acentos, debido a que más tarde al ser capturados en la base de datos no será posible mantenerlos.

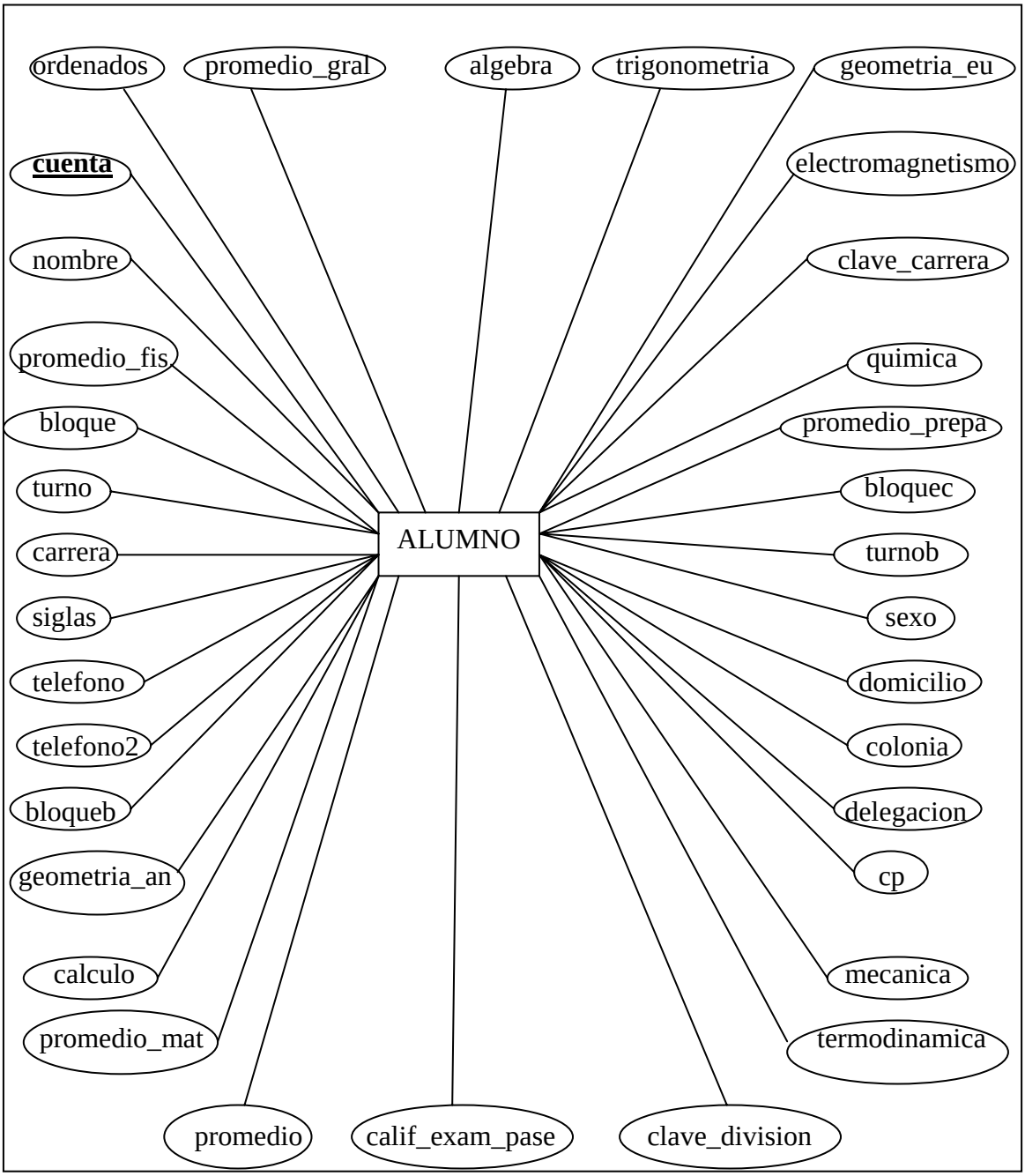


Figura 5.8 Diagrama Entidad-Relación de la Entidad ALUMNO

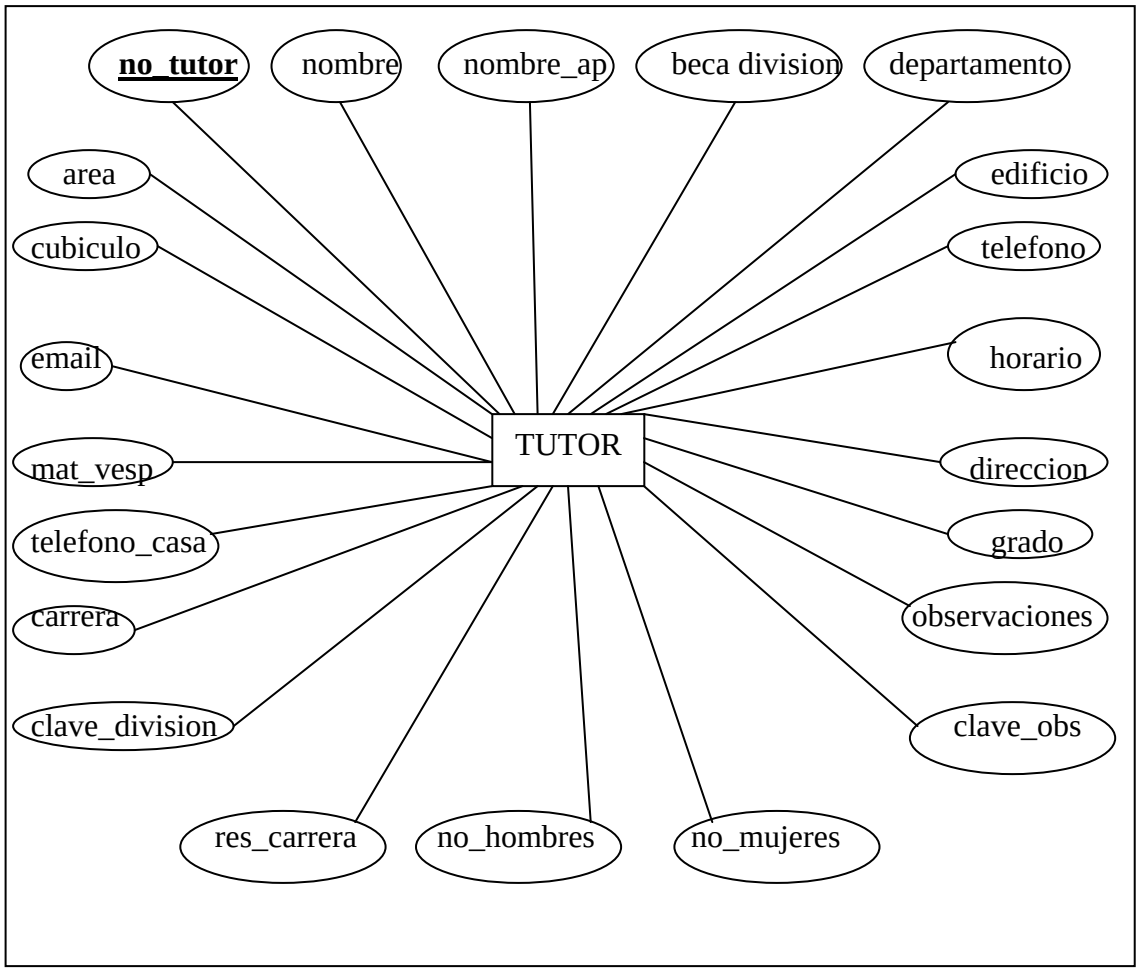


Figura 5.9 Diagrama Entidad Relación de la Entidad TUTOR

Estos archivos en Excel se convirtieron a **MySQL** mediante un programa hecho en lenguaje **PHP,** el que se describirá en el Capítulo 6. Programación del sistema.

Para la entidad **TUTOR**, se tomó como llave foránea, al número de tutor (**no_tutor**), ya que este número es un identificador único que el administrador del sistema le asignó a cada Tutor.

En la Entidad **ALUMNO**, la llave primaria es el número de cuenta (**cuenta**), ya que es el identificador en cualquier base de datos de alumnos en nuestra Universidad, además de ser el único atributo que es unívoco en cada alumno.

El diagrama Entidad Relación del **SIATT** se estructuró, partiendo de que se tienen dos Entidades, **ALUMNO** y **TUTOR**, a cada alumno se le asigna un tutor, pero un tutor puede tener muchos alumnos, por lo tanto la relación sería de uno a muchos.

La teoría de base de datos sugiere hacer una relación llamada **asigna** de la tabla **TUTOR** con **ALUMNO,** en nuestro caso, esto no resuelve el problema principal que es asignarle un tutor a un alumno siguiendo reglas que cambian conforme avanza la asignación.

El programa que hace la asignación es una serie de asignaciones o consultas sucesivas, programadas para que éstas vayan cambiando a partir del resultado de consultas anteriores, cada que se hace una consulta el resultado de ésta se debe ir guardándose en una tabla llamada **ASIGNADO** que es la tabla de alumnos asignados.

El sistema requiere que la información de las asignaciones sucesivas se vaya guardando, debido a que la asignación se va haciendo con respecto a reglas y restricciones dadas por el administrador del sistema.

Estas reglas y restricciones van cambiando conforme avanza el programa de asignación, que es el corazón de nuestro sistema.

Por lo tanto, debido a que una consulta en SQL como podría ser un JOIN o unión entre las tablas ALUMNO y TUTOR no resuelve nuestro problema de asignarle un tutor a cada alumno siguiendo reglas y restricciones que cambian dependiendo del resultado de una consulta anterior, se determino hacer un programa llamado asignación que se describe más a fondo en el Capítulo 6. Programación del sistema.

En la figura 5.10 observamos que un tutor puede tener varios alumnos asignados, que un alumno asignado sólo puede tener un tutor, que un alumno se transforma en un alumno asignado y que un alumno asignado puede transformarse en un alumno.

Observamos que no existe ninguna relación directa entre las entidades ALUMNO y TUTOR también se observa que la entidad ASIGNADO ayuda a formar esta relación faltante y al mismo tiempo almacenar los datos de la asignación.

En la entidad **ASIGNADO**, se tomaron como atributos, los atributos más significativos de las otras dos entidades (llaves primaria y foránea), además de incluir atributos que nos den un vistazo rápido de como quedó la asignación con solo consultar esta entidad.

En la figura 5.11 se describe como quedó el diagrama **Entidad-Relación** después de tomar dichas consideraciones.

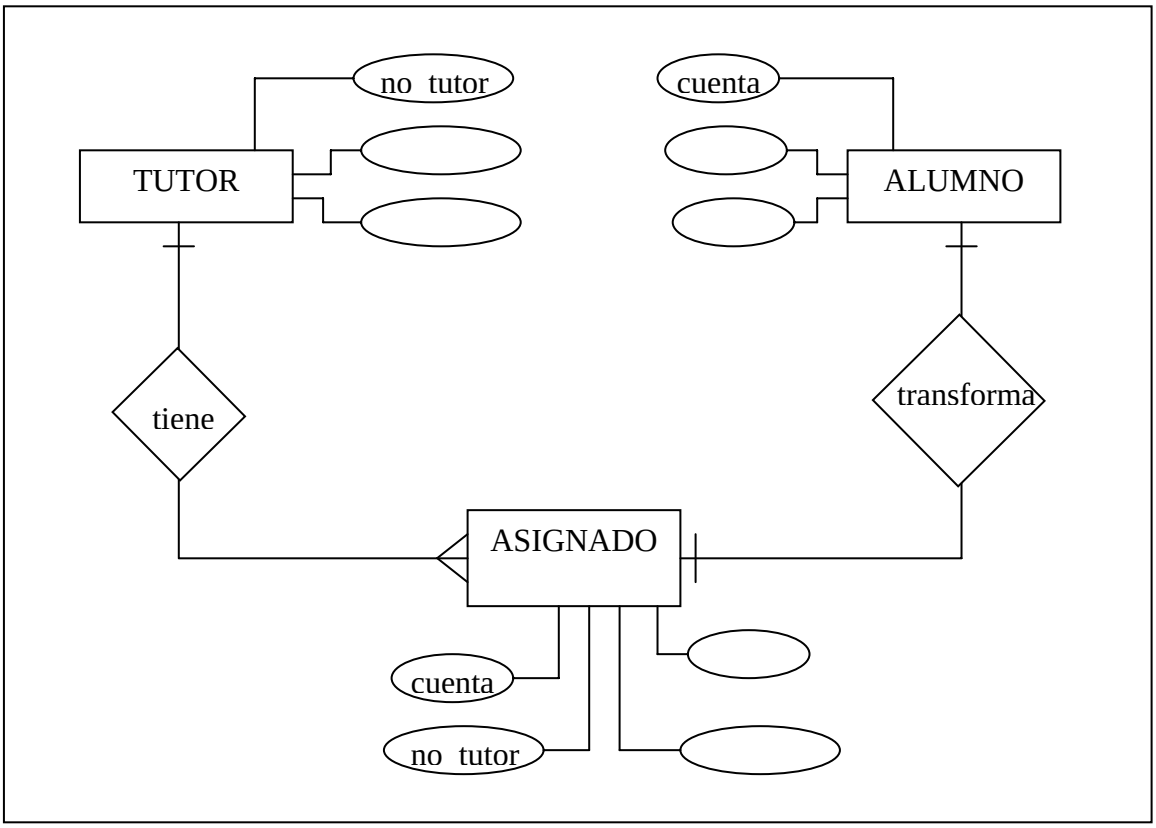


Figura 5.10 Diagrama Entidad-Relación del SIATT

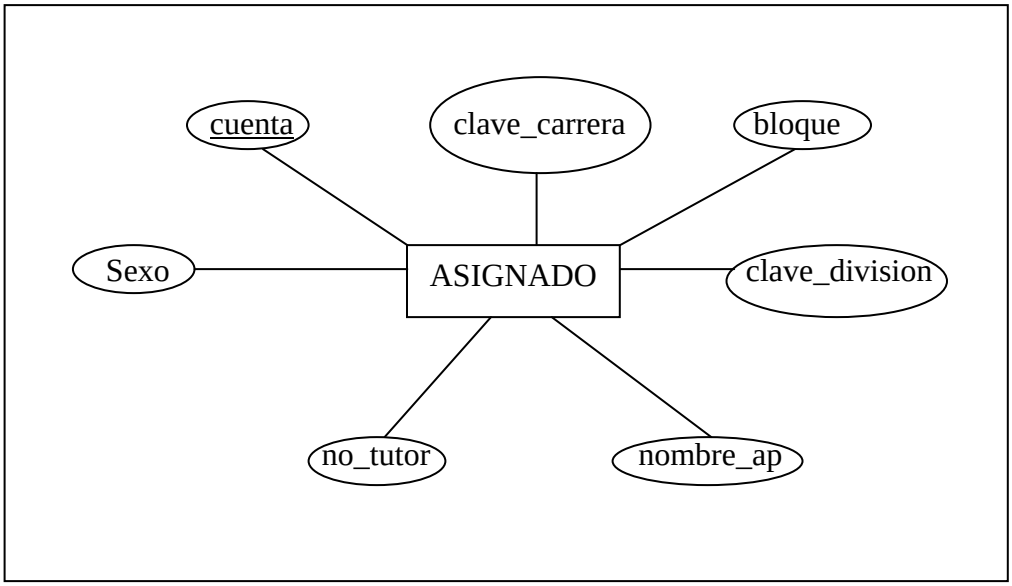


Figura 5.11 Diagrama Entidad-elación de la Entidad ASIGNADO

CAPÍTULO 6

Programación del sistema

6.1 Programación del sistema

Ya se ha realizado un estudio detallado en definir el problema, seleccionar que herramientas usar, elegir que método de diseño de software nos convenía más y llevarlo a cabo haciendo el diseño, viene ahora la programación del sistema, que está dividida en prototipos que se fueron mejorando hasta llegar a versiones finales uniéndolos para formar lo que es el **SIATT**.

En el Capítulo 3. Herramientas y Lenguajes de Programación se dieron las razones técnicas y teóricas por las que se escogieron como herramientas a, **Apache** como servidor Web, **PHP** como lenguaje de programación y **MySQL** como el manejador de base de datos.

La programación del sistema se divide en tres partes:

a) Programación de la interfaz

b) Programación de la base de datos y sus tablas

c) Programación de las consultas

A continuación, explicamos a detalle como se llevaron a cabo cada uno de éstos y la forma en que fueron unidos.

6.2 Programación de la interfaz

En la realización de las interfaces del **SIATT**, el intercalar **PHP** con **HTML** y viceversa de una manera muy sencilla, conlleva grandes ventajas en la realización de páginas dinámicas ya que facilita en gran medida la programación.

Es práctico principalmente porque todas las funciones explícitas de un lenguaje se pueden adaptar al otro y las variables que no se pueden manejar con éste, se pueden manejar con las características del otro lenguaje.

Al momento de diseñar la interfaz, existen diferentes características que debemos tomar en cuenta si queremos que el usuario tenga una buena relación con el sistema y viceversa.

1. Control

El usuario debe sentir siempre que está en control de la aplicación, esto implica que la aplicación es:

- Interactiva
- El usuario es el que indica la interactividad y no el sistema

2. Intuitivo

La interfaz gráfica debe de dar una manera directa e intuitiva para interaccionar con el sistema.

3. Consistencia

La interfaz debe ser consistente en todo el sistema. Las habilidades aprendidas en una sección deben repetirse en todo el sistema.

4. No penalizar

La aplicación no debe penalizar al usuario por errores cometidos. El usuario debe ser informado de operaciones peligrosas.

5. Claridad

La interfaz debe ser conceptual y visiblemente clara.

6. Retroalimentación

El usuario debe recibir de manera directa e inmediata respuesta a sus acciones.

7. Estética

Los efectos visuales no deben de distraer al usuario.

8. Simplicidad

Evitar demasiados mensajes que confundan al usuario. La información se muestra de manera paulatina y sólo cuando es necesario.

9. Estabilidad

La aplicación debe consistir en un conjunto claro y finito de objetos y acciones.

10. Ver y seleccionar

Los usuarios deben centrar más su atención en reconocer que en recordar.

La primera interfaz que se encuentra el usuario cuando entra al sistema es la pantalla de registro y validación de nombre de usuario y contraseña, ver figura 6.1.



Figura 6.1 Interfaz de registro y validación

Esta interfaz captura los datos introducidos por el usuario y los compara con los residentes en la base de datos, si éstos coinciden da acceso al sistema.

En el diseño se usa **HTML** y se hace uso de un formulario para pasar los datos como variables mediante el método post, a un programa en **PHP** llamado Acceso.php, que los procesara, a continuación se presenta el código de dicho formulario.

```
<FORM action="Acceso.php" METHOD="POST">
<center>
   <p>Tipo de usuario:
```

```
    <select name="tusuario" size="1" id="tusuario">
      <option value="NULL" selected></option>
      <option value="Ad">Administrador</option>
      <option value="Ay">Ayudante</option>
        <option value="Ot">Otro</option>
                </select>
    Usuario:
    <input name="usuario" type="text" id="usuario">
    Contrase&ntilde;a:
    <INPUT NAME="clave" TYPE="password" id="clave" size="19" maxlength="16">
    <INPUT name="Aceptar" TYPE="submit" id="Aceptar" value="Aceptar">
  </p>
</center>
</FORM>
```

Para intercalar PHP y HTML sólo hace falta delimitar entre los símbolos **<?php** cuando inicia este lenguaje y **?>** cuando se termina.

Los datos del formulario son recibidos por el programa Acceso.php, el cual se conecta a la base de datos para comparar el tipo de usuario, usuario y contraseña, si los datos coinciden con los de la base de datos permite el acceso pero dependerá del tipo de usuario a que parte del sistema nos conduzca y con que privilegios.

En las partes iniciales de cualquier script se deben realizar los llamados a páginas que conlleven funciones o subrutinas ocupadas en la ejecución del programa, en nuestro caso **SIAT.php** tiene estas subrutinas de conexión a la base de datos.

```
<?php require_once('MrBungle/SIAT.php'); ?>   //SIAT.php  se encuentra en el direcorio
//MrBungle, por razones de seguridad.

//función para iniciar la conección a la base de datos, más adelante se explica a detalle.

<?php
$tusuario = $_POST["tusuario"];   //recibe las tres variables por el método post
$usuario =$_POST["usuario"];
$clave =$_POST["clave"];

//hace una consulta  a la base de datos  para saber si existe ese usario y capta los datos de
//ser asi
$query_BuscaUsuario = "select usuario.tusuario, usuario.usuario, usuario.clave from
usuario  where usuario = '$usuario' and tusuario = '$tusuario'";
$BuscaUsuario = mysql_query($query_BuscaUsuario) or die(mysql_error());
$row_BuscaUsuario = mysql_fetch_assoc($BuscaUsuario);
$totalRows_BuscaUsuario = mysql_num_rows($BuscaUsuario);

?>
```

```
<?php

//si los tres campos contienen algún dato empieza a comparar

   if ( ($tusuario != "") and ($usuario != "") and ($clave != "") ) {

   //si el tipo de usuario, usuario y clave mandado por el formulario es igual al tipo de
//usuario, usario y clave del encontrado en la base de datos el acceso es concedido.

        if (($tusuario == $row_BuscaUsuario['tusuario']) and ($usuario ==
$row_BuscaUsuario['usuario']) and ($clave == $row_BuscaUsuario['clave']))
        { echo "<br><label>Acceso Concedido...</label><br>";


//si el tipo de usuario es Ad (Administrador), entonces es redireccionado a la sección
//Principal  contados los privilegios, como son: editar, borrar o agregar usuarios.
//Este  Menú Principal es un programa en PHP llamado AdminSist.php

if ($tusuario == "Ad")
                        {
                        $url = "AdminSist.php"; // target of the redirect
                        $delay = "3"; // 3 second delay
                        echo "<meta http-equiv='refresh' content=".$delay.";url=".$url.">";
                        }
//Si no es Ad entra al Menu de Ayudante donde no puede editar, borrar o agregar usuarios
                        else
                        {
                        $url = "Ayudante.php"; // target of the redirect
                        $delay = "3"; // 3 second delay
                        echo "<meta http-equiv='refresh' content=".$delay.";url=".$url.">";
                        }
                }

   //si el tipo de usuario, usuario y clave mandado por el formulario NO es igual al tipo de
//usuario, usuario y clave del encontrado en la base de datos el acceso es denegado y
redireccionado a la página de inicio para que vuelva a registrarse correctamente.

                else
                {
                echo "<br><label>Acceso Denegado...</label><br>";
                $url = "Inicio.php"; // target of the redirect
                $delay = "3"; // 3 second delay
                echo "<meta http-equiv='refresh' content=".$delay.";url=".$url.">";
                }
        }
        else
```

```
        {
//si alguno de los tres campos no contienen algún dato redirecciona a la página de inicio

        echo "Por favor ingrese nuevamente sus datos";
        $url = "Inicio.php"; // target of the redirect
        $delay = "3"; // 3 second delay
        echo "<meta http-equiv='refresh' content=".$delay.";url=".$url.">";
        }
?>
<?php
mysql_free_result($BuscaUsuario);
?>
```

Una vez validado el usuario, entra al **menú principal**, este menú cambia dependiendo el tipo de usuario, si es Administrador se despliega como en la figura 6.2. En caso de ser ayudante desplegaría un menú idéntico, pero sin la sección **Administrar usuarios del sistema.**



Figura 6.2 Interfaz del menú principal.

Este formulario o menú que se maneja como formato en HTML, por lo que se tiene que dar de alta el tipo de tabla, bordes, alineación, etc., además de manejar colores y tamaños de letras.

Para los formularios en HTML existen varios tipos de entradas que se observan en las variables del sistema, como **text area** que demarca una entrada de texto pero con un largo y un ancho.

Password (en la interfaz de registro) que indica que el campo a introducir será una palabra de acceso restringido por lo que muestra asteriscos en lugar de letras escritas; **Checkbox** donde el campo se elige marcando una entre varias opciones de casillas cuadradas y opciones como las siguientes:

- **type**, indica el tipo de variable a introducir.
- **text**, indica que el campo a introducir será un texto.
- **maxlenght** seguido de un valor, limita el número máximo de caracteres a introducir en ese campo.
- **size,** seguido de un valor, limita el número de caracteres a mostrar en pantalla.
- **value**, indica que no hay valor inicial del campo, entre comillas se indica el valor de la casilla.
- **checked,** la casilla aparece marcada por omisión.
- **radio**, el campo se elige marcando una casilla circular entre varias opciones.
- **image**, el campo contendrá el valor en coordenadas del punto de la imagen que haya delimitado. Pero debe llevar obligatoriamente el atributo: ***src***, y entre comillas el nombre del archivo de imagen.
- **hidden**, el visitante no puede modificar su valor ya que no está visible. Se manda siempre junto al atributo *value*, seguido de su valor entre comillas.

Las variables de entrada del sistema tienen varias características de las escritas anteriormente y se identifican cuando se definen.

El código de este menú se muestra a continuación:

```
<html>
<head>
<title>Administraci&oacute;n del Sistema SIATT...</title>
</head>
<body>
<center><br><br>
  <br>
```

```
 <img src="Imagenes/logocopadi.jpg" width="128" height="130" align="left"><font
color="#0000CC" size="7" face="Times New Roman, Times, serif"><img
src="Imagenes/escudoazul.GIF" width="176" height="197" align="right" ></font>
 <font color="#0000CC" size="7" face="Times New Roman, Times, serif">S I A T T
</font>
 <p><font color="#0000CC" size="5" face="Times New Roman, Times, serif">Sistema de
Informaci&oacute;n para la Asignaci&oacute;n de tutores en el &Aacute;rea de
Tutor&iacute;a para Todos</font></p>
 <br>
</center>
<br>
<center>
 <h1> <br>
  <strong>Bienvenido.</strong></h1>
 <table width="85%" border="1">
  <tr>
   <td><br><strong><div align="center">Iniciar
Asignaci&oacute;n</strong></div><br></td>
   <td><form name="form1" method="post" action="asignar.php">
      <div align="center">
       <input type="submit" name="Submit2" value="          Iniciar          ">
      </div>
     </form></td>
  </tr>
                                                    <tr>
   <td><strong><div align="center"><strong>Busqueda</strong></div></td>
   <td><form name="form2" method="post" action="BusquedaTutor.php">
      <div align="center">
       <input type="submit" name="Submit3" value="          Tutor          ">
      </div>
     </form>
                                                    <form name="form3"
method="post" action="BusquedaAlumno.php">
      <div align="center">
       <input type="submit" name="Submit3" value="          Alumno        ">
      </div>
     </form></td>
  </tr>
  <tr>
   <td><div align="center"><strong>Modificar Datos</strong></div></td>
   <td><form name="form4" method="post" action="ModificarTutor.php">
      <div align="center">
       <input type="submit" name="Submit" value="          Tutor          ">
      </div>
     </form>
                                                    <form name="form5"
method="post" action="ModificarAlumno.php">
```

```
        <div align="center">
          <input type="submit" name="Submit" value="          Alumno        ">
        </div>
       </form>
                                                              </td>
   </tr>
   <tr>
     <td><br><div align="center"><strong>Administrar Usuarios de
Sistema</strong></div><br></td>
     <td><form name="form4" method="post" action="AdminUsuarios.php">
        <div align="center">
          <input type="submit" name="Submit" value="Administrar Sistema">
        </div>
       </form></td>
   </tr>
  </table>
  <h1><br>
    <br>
    <br>
  </h1>
</center>
</body>
</html>
```

6.3 Programación de la base de datos y sus tablas

La programación de la base de datos ya se llevó a cabo en la etapa de **diseño**, por lo tanto ya se tiene bien definido como se llamara **la base de datos, las tablas y sus campos**.

Esta programación se lleva a cabo mediante MySQL que usa una variación del SQL estándar.

Primero se necesita crear la base de datos para continuar con las tablas y su programación se realiza de la siguiente manera:

Primero creamos la base de datos

% **create database asignacion;**
Query OK, 1 row affected (0.02 sec)

Al tener la base de datos creada, se comienza a estructurar y delimitar las tablas. En sí, todas las tablas fueron programadas de manera muy similar y se observará en el siguiente script.

La primera tabla mostrada es **'alumno',** que tiene 40 campos y es utilizada para identificar al alumno y guardar todos sus datos.

A continuación se da una breve descripción de los campos más importantes de esta tabla:

Ordenados
Es un número que no se usa pero se pone para futuras referencias (fue colocado por el administrador, en una asignación hecha a mano)

Cuenta
Número de identificación de los alumnos en la tabla y es la llave primaria.

Nombre
Nombre del alumno empezando por apellido paterno.

Clave carrera
Número de la carrera a la que pertenece el alumno.

Bloque
Número de bloque o grupo al que pertenece el alumno**.**

Clave_division
Número de la división a la que pertenece el alumno, cada división tiene un número que va del 1 al 6

Asignado
Nos indica si este alumno ya fue asignado, toma el valor de 1 si ya fue asignado, en caso contrario se queda con 0.

Esta es la forma en que se introdujo la estructura de la tabla **ALUMNO** con todos sus campos.

```
CREATE TABLE `ALUMNO` (
  `ordenados` int(11) NOT NULL default '0',
  `n1` text,
  `n2` text,
  `cuenta` bigint(20) NOT NULL default '0',
  `nombre` text NOT NULL,
  `clave_carrera` int(11) NOT NULL default '0',
```

```
  `bloque` int(11) NOT NULL default '0',
  `turno` text NOT NULL,
  `n3` text,
  `n4` text,
  `n5` text,
  `carrera` text NOT NULL,
  `siglas` text NOT NULL,
  `n6` text,
  `telefono` varchar(8) NOT NULL default '',
  `telefono2` varchar(8) NOT NULL default '',
  `bloqueb` int(11) NOT NULL default '0',
  `promedio_gral` double NOT NULL default '0',
  `algebra` int(11) NOT NULL default '0',
  `trigonometria` int(11) NOT NULL default '0',
  `geometria_eu` int(11) NOT NULL default '0',
  `geometria_an` int(11) NOT NULL default '0',
  `calculo` int(11) NOT NULL default '0',
  `promedio_mat` double NOT NULL default '0',
  `mecanica` int(11) NOT NULL default '0',
  `termodinamica` int(11) NOT NULL default '0',
  `electromagnetismo` int(11) NOT NULL default '0',
  `promedio_fis` double NOT NULL default '0',
  `quimica` int(11) NOT NULL default '0',
  `promedio_prepa` double NOT NULL default '0',
  `bloquec` int(11) NOT NULL default '0',
  `turnob` text NOT NULL,
  `sexo` int(11) NOT NULL default '0',
  `domicilio` text NOT NULL,
  `colonia` text NOT NULL,
  `delegacion` text NOT NULL,
  `cp` bigint(20) NOT NULL default '0',
  `promedio` int(11) NOT NULL default '0',
  `calif_exam_pase` int(11) NOT NULL default '0',
  `clave_division` int(11) NOT NULL default '0',
  `asignado` int(11) NOT NULL default '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

La siguiente tabla es **TUTOR** que tiene 25 campos y es utilizada para identificar al tutor y guardar todos sus datos personales.

Los campos que nos interesan a la hora de hacer la asignación son:

- **no_tutor,** es el número que se le asigna a cada tutor para identificarlo y es la llave primaria del sistema**.**

- **nombre_ap**, es el nombre completo del tutor, empezando por el apellido.
- **clave_division,** número de la división donde labora el tutor.
- **clave_obs,** tiene que ver con la participación del tutor, se clasifican en: aa=Participativo, a=normal, b=con restricción de que sus alumnos asignados sean de la misma carrera c=No convencidos de la tutoría o con poca disponibilidad.
- **res_carrera,** es la restricción de carrera, toma el valor de la clave de la carrera a la que tiene que pertenecer el alumno asignado.
- **no_hombres,** número de alumnos que desea tener el tutor.
- **no_mujeres,** número de alumnas que desea tener el tutor.
- **asignado,** indica si el tutor ya fue asignado con el número 1 o se queda con 0 si no ha sido asignado.
- **aleatorio,** se pone en 1 cuando los alumnos fueron asignados aleatoriamente.

La tabla **TUTOR** se introdujo asi:

```
CREATE TABLE `TUTOR` (
  `no_tutor` int(11) NOT NULL default '0',
  `nombre` text NOT NULL,
  `nombre_ap` text NOT NULL,
  `beca` text NOT NULL,
  `division` text NOT NULL,
  `departamento` text NOT NULL,
  `area` text NOT NULL,
  `edificio` text NOT NULL,
  `cubiculo` text NOT NULL,
  `telefono` text NOT NULL,
  `email` text NOT NULL,
  `horario` text NOT NULL,
  `mat_vesp` text NOT NULL,
  `direccion` text NOT NULL,
  `telefono_casa` text NOT NULL,
  `grado` text NOT NULL,
  `carrera` text NOT NULL,
  `observaciones` text NOT NULL,
  `clave_division` int(11) NOT NULL default '0',
  `clave_obs` text NOT NULL,
  `res_carrera` int(11) NOT NULL default '0',
```

```
  `no_hombres` int(11) NOT NULL default '0',
  `no_mujeres` int(11) NOT NULL default '0',
  `asignado` int(11) NOT NULL default '0',
  `aleatorio` int(11) NOT NULL default '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

La tabla siguiente es **ASIGNACIÓN** que es donde se van a guardar los datos más importantes del alumno y su tutor asignado.

- **cuenta**, número de identificación de los alumnos en la tabla y es la llave primaria.
- **clave_carrera**, número de la carrera a la que pertenece el alumno.
- **bloque,** número de bloque o grupo al que pertenece el alumno**.**
- **sexo,** sexo del alumno, 1=hombre o 2=mujer.
- **clave_division,** número de la división a la que pertenece el alumno, cada división tiene un número que va del 1 al 6.
- **no_tutor,** es el número que se le asigna a cada tutor para identificarlo y es la llave primaria del sistema**.**
- **nombre_ap,** es el nombre completo del tutor, empezando por el apellido.

A continuación se presenta como se introdujeron los comandos para crear la tabla **ASIGNADOS**:

```
CREATE TABLE `asignado` (
  `cuenta` bigint(20) NOT NULL default '0',
  `clave_carrera` int(11) NOT NULL default '0',
  `bloque` int(11) NOT NULL default '0',
  `sexo` int(11) NOT NULL default '0',
  `clave_division` int(11) NOT NULL default '0',
  `no_tutor` int(11) NOT NULL default '0',
  `nombre_ap` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

La tabla que se utiliza para la validación de usuario y contraseña es **USUARIO** y tiene los siguientes campos:

tusuario, tipo de usuario Ad=Administrador del sistema, Ay= ayudante, Ot=Otro

usuario, nombre de usuario

clave, contraseña para accesar al sistema.

Se construy de esta forma:

```
CREATE TABLE `usuario` (
  `tusuario` char(2) NOT NULL default '',
  `usuario` varchar(12) NOT NULL default '',
  `clave` varchar(12) NOT NULL default ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

La última tabla es **ALEATORIO,** la cual sirve para guardar las posibles combinaciones de hombres y mujeres que se le pueden asignar a un tutor.

Sólo tiene dos campos, a continuación se presentan las posibles combinaciones que podemos tener en cada tutor.

no_hombres	no_mujeres
10	2
9	3
10	3
9	2

Estos datos se asignaron después de sacar que porcentaje de mujeres existe en las generaciones anteriores, resultando de un 22%.

La tabla se introdujo al sistema de la siguiente manera:

```
CREATE TABLE `ALEATORIO` (
  `no_hombres` int(11) NOT NULL default '0',
  `no_mujeres` int(11) NOT NULL default '0'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Los datos se introducen de esta forma:

```
INSERT INTO ` ALEATORIO ` VALUES (10, 2);
INSERT INTO ` ALEATORIO ` VALUES (9, 3);
INSERT INTO ` ALEATORIO ` VALUES (10, 3);
INSERT INTO ` ALEATORIO ` VALUES (9, 2);
```

6.4 Programación de las consultas

Esta es la parte de la programación donde se hace la asignación de los alumnos, esta asignación debe tomar en cuenta varias restricciones puestas por el tutor y por el sistema, por lo tanto, tiene que estar conectándose a la base de datos a revisar estas restricciones, y a estas acciones les llamamos consultas, el sistema se compone de un programa principal, el cual manda a llamar a funciones (subrutinas) para ir haciendo la asignación por pasos, tal como la haría cualquier persona.

Primero describimos la rutina de conexión con la base de datos.

Siempre que encontremos esta sentencia, ésta estará haciendo referencia a la conexión a la base de datos.

```
<?php require_once('MrBungle/SIAT.php'); ?>
```

Nos pide ir al directorio MrBungle y ejecutar el programa SIAT.php este programa se muestra acontinuación:

```
<?php
//cadena de conexion
$conexion=@mysql_connect("localhost","root","fantomas");
 if (! $conexion){
echo "Error al intentar conectarse con el servidor MySQL";
exit();
}
if (! @mysql_select_db("asignacion",$conexion)){
echo "No se pudo conectar correctamente con la base de datos";
exit();
}
?>
```

En el lenguaje **PHP** existe una librería para la conexión a la base de datos llamada @mysql_connect, ésta debe contener los parámetros, dominio, usuario y contraseña del manejador MySQL al que estamos queriendo conectar.

Cuando el usuario quiere hacer la asignación, recibe dos documentos en Excel, uno con todos los datos de los tutores disponibles y otro con todos los datos de la generación de

alumnos, el sistema tiene que ser capaz de convertir estos archivos a MySQL para poder trabajar con toda la información.

Esta conversión de datos se realiza mediante un programa llamado convertir.php y convertirt.php, estos dos programas son muy similares y básicamente lo que hacen es abrir el documento en Excel dada una ruta donde se encuentra el archivo, ir iterando en un ciclo que dura el número de registros (alumnos o tutores) que se tengan y dentro de cada iteración hace otro ciclo con iteraciones que durarán tantas columnas tenga.

A continuación se presenta el código de convertir.php y convertirt.php respectivamente.

```
<?php require_once('MrBungle/SIAT.php'); ?>

<?php

 $excel = new COM("Excel.Application") or die("Excel could not be started");

$excel -> Visible = 0;

$excel -> DisplayAlerts = 0;

$data = $excel -> Workbooks -> Open("C:\Documents and Settings\Alvaro\Mis documentos\Tesis\Documentos\Gen 2006");

$Sheets = $data -> Worksheets(1);

for($i = 2; $i <= 2200; $i++){

for($x = 1; $x <= 40; $x++){

$Cell = $Sheets -> Cells($i, $x);

$Cell -> activate;

switch($x){

case 1; $ordenados = $Cell -> value; break;
case 2; $n1 = $Cell -> value; break;
case 3; $n2 = $Cell -> value; break;
case 4; $cuenta = $Cell -> value; break;
case 5; $nombre = $Cell -> value; break;
case 6; $clave_carrera = $Cell -> value; break;
case 7; $bloque = $Cell -> value; break;
case 8; $turno = $Cell -> value; break;
```

```
case 9; $n3 = $Cell -> value; break;
case 10; $n4 = $Cell -> value; break;
case 11; $n5 = $Cell -> value; break;
case 12; $carrera = $Cell -> value; break;
case 13; $siglas = $Cell -> value; break;
case 14; $n6 = $Cell -> value; break;
case 15; $telefono = $Cell -> value; break;
case 16; $telefono2 = $Cell -> value; break;
case 17; $bloqueb = $Cell -> value; break;
case 18; $promedio_gral = $Cell -> value; break;
case 19; $algebra = $Cell -> value; break;
case 20; $trigonometria = $Cell -> value; break;
case 21; $geometria_eu = $Cell -> value; break;
case 22; $geometria_an = $Cell -> value; break;
case 23; $calculo = $Cell -> value; break;
case 24; $promedio_mat = $Cell -> value; break;
case 25; $mecanica = $Cell -> value; break;
case 26; $termodinamica = $Cell -> value; break;
case 27; $electromagnetismo = $Cell -> value; break;
case 28; $promedio_fis = $Cell -> value; break;
case 29; $quimica = $Cell -> value; break;
case 30; $promedio_prepa = $Cell -> value; break;
case 31; $bloquec = $Cell -> value; break;
case 32; $turnob = $Cell -> value; break;
case 33; $sexo = $Cell -> value; break;
case 34; $domicilio = $Cell -> value; break;
case 35; $colonia = $Cell -> value; break;
case 36; $delegacion = $Cell -> value; break;
case 37; $cp = $Cell -> value; break;
case 38; $promedio = $Cell -> value; break;
case 39; $calif_exam_pase = $Cell -> value; break;
case 40; $clave_division = $Cell -> value; break;
}
}
$SQL = mysql_query("INSERT INTO ALUMNO VALUES('$ordenados', NULL, NULL,
'$cuenta', '$nombre', '$clave_carrera', '$bloque', '$turno', NULL, NULL, NULL, '$carrera',
'$siglas', NULL, '$telefono', '$telefono2', '$bloqueb', '$promedio_gral', '$algebra',
'$trigonometria', '$geometria_eu', '$geometria_an', '$calculo', '$promedio_mat', '$mecanica',
'$termodinamica', '$electromagnetismo', '$promedio_fis', '$quimica', '$promedio_prepa',
'$bloquec', '$turno', '$sexo', '$domicilio', '$colonia', '$delegacion', '$cp', '$promedio',
'$calif_exam_pase', '$clave_division')");

}


$excel -> Quit();
```

```
?>


<?php require_once('MrBungle/SIAT.php'); ?>

<?php

$excel = new COM("Excel.Application") or die("Excel could not be started");

ini_set('max_execution_time','120');

$excel -> Visible = 0;

$excel -> DisplayAlerts = 0;

$data = $excel -> Workbooks -> Open("C:\Documents and Settings\Alvaro\Mis documentos\Tesis\Documentos\Tutoresb");

$Sheets = $data -> Worksheets(1);

for($i = 5; $i <= 200; $i++){

for($x = 1; $x <= 23; $x++){

$Cell = $Sheets -> Cells($i, $x);

$Cell -> activate;

switch($x){
case 1; $no_tutor = $Cell -> value; break;
case 2; $nombre = $Cell -> value; break;
case 3; $nombre_ap = $Cell -> value; break;
case 4; $beca = $Cell -> value; break;
case 5; $division = $Cell -> value; break;
case 6; $departamento = $Cell -> value; break;
case 7; $area = $Cell -> value; break;
case 8; $edificio = $Cell -> value; break;
case 9; $cubiculo = $Cell -> value; break;
case 10; $telefono = $Cell -> value; break;
case 11; $email = $Cell -> value; break;
case 12; $horario = $Cell -> value; break;
case 13; $mat_vesp = $Cell -> value; break;
case 14; $direccion = $Cell -> value; break;
case 15; $telefono_casa = $Cell -> value; break;
case 16; $grado = $Cell -> value; break;
```

```
case 17; $carrera = $Cell -> value; break;
case 18; $observaciones = $Cell -> value; break;
case 19; $clave_division = $Cell -> value; break;
case 20; $clave_obs = $Cell -> value; break;
case 21; $res_carrera = $Cell -> value; break;
case 22; $no_hombres = $Cell -> value; break;
case 23; $no_mujeres = $Cell -> value; break;
}
}
$SQL = mysql_query("INSERT INTO TUTOR VALUES('$no_tutor', '$nombre', '$nombre_ap', '$beca', '$division', '$departamento', '$area', '$edificio', '$cubiculo', '$telefono', '$email', '$horario', '$mat_vesp', '$direccion', '$telefono_casa', '$grado', '$carrera', '$observaciones', '$clave_division', '$clave_obs', '$res_carrera', '$no_hombres', '$no_mujeres')");
}
$excel -> Quit();
?>
```

Lo siguiente será explicar cómo funciona el programa asignacion.php qué es el programa principal del sistema, ya que éste realiza la tarea principal para lo que fue hecha esta aplicación.

El programa está dividido en subrutinas llamadas, cada subrutina es un programa que en su interior puede tener programada una o más funciones que hace una serie de consultas y guarda datos en tablas.

Las funciones integradas en PHP son muy fáciles de utilizar. Tan sólo hemos de realizar la llamada de la forma apropiada y especificar los parámetros y/o variables necesarios para que la función realice su tarea.

```
<?php include ("../funcion.php");
funcion1();
funcion2();
      .
      .
funcionN();
?>
```

En nuestro código siempre que se manda a llamar una función, ésta solo está incluida dentro de un archivo que se llama igual que ella, pero con la extensión php.

A continuación se presenta el código del programa asignación.php :

```
<?php require_once('MrBungle/SIAT.php'); ?>

<?php

include("asignacion1.php");
asignacion1();

include("asignacion2a.php");
asignacion2a();

include("asignacion2b.php");
asignacion2b();

include("asignacion3a.php");
asignacion3a();

include("asignacion3b.php");
asignacion3b();

include("asignacion4a.php");
asignacion4a();

include("asignacion4b.php");
asignacion4b();

include("asignacion5.php");
asignacion5();

?>
```

Primero se conecta a la base de datos, enseguida manda a llamar a la función **asignacion1** que se encuentra dentro del archivo **asignacion1.php.**

Esta función lo que hace es asignar los profesores que hay de cada división, que tengan la **restricción** de que el alumno **sea de su misma carrera**, el código comentado se presenta a continuación.

```
<?php

Function asignacion1()

{
//Esto se repetira para cada división de la 2 a la 5
        for($i=2; $i<=5; $i=$i+1){

//cuento cúantos profesores hay de esta división que tengan la restricción de que el alumno sea de su misma carrera
$cuenta = mysql_query("SELECT no_tutor FROM TUTOR WHERE clave_division=$i AND res_carrera!=0 AND asignado=0");
$cuentotut = mysql_num_rows($cuenta);

//entro al ciclo que durará hasta el número de TUTORES que se hayan encontrado
for($x=1; $x<=$cuentotut; $x=$x+1){
$seltutor = "SELECT no_tutor, nombre_ap, no_hombres, no_mujeres, res_carrera FROM TUTOR
WHERE res_carrera!=0 AND asignado=0 ORDER BY RAND() LIMIT 0,1";
$result = mysql_query($seltutor);

//Si existe algún tutor sin asignar empieza la asignación
if ($result<>''){

//verificamos si la consulta tiene algún error
        if (! $result){
echo "La consulta SQL contiene errores.";
exit();
}


//si no tiene errores guardamos los datos en un arreglo
While($row=mysql_fetch_object($result))
{
  //Le asignamos una variable a cada elemento del arreglo, a cada columna
  $no_tutor=$row->no_tutor;
  $nombre_ap=$row->nombre_ap;
  $no_hombres=$row->no_hombres;
  $no_mujeres=$row->no_mujeres;
  $res_carrera=$row->res_carrera;
}
```

```
//Le asigno sus alumnas al Tutor
$asigmujeres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_carrera=".$res_carrera."
AND sexo=2 AND asignado!=1
ORDER BY bloque
LIMIT 0,".$no_mujeres."";
mysql_query($asigmujeres);


//le asigno sus alumnos al Tutor
$asighombres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_carrera=".$res_carrera."
AND sexo=1 AND asignado!=1
ORDER BY bloque
LIMIT 0,".$no_hombres."";
mysql_query($asighombres);

//les asigno a todos los alumnos asignados en esta iteración su número de tutor
$asigtutor = "UPDATE ASIGNADO SET no_tutor=$no_tutor WHERE no_tutor=0";
mysql_query($asigtutor);
$asigtutor2 = "UPDATE ASIGNADO SET nombre_ap=$nombre_ap WHERE
no_tutor=$no_tutor";
mysql_query($asigtutor2);

//le pongo marca asignado=1 para saber que este tutor ya fue asignado
$marcatutor = "UPDATE TUTOR SET asignado=1 WHERE no_tutor=$no_tutor";
mysql_query($marcatutor);

//le pongo marca asignado=1 para saber que este alumno ya fue asignado
$marcaalumno = "UPDATE ALUMNO, ASIGNADO SET ALUMNO.asignado=1
WHERE ASIGNADO.no_tutor=$no_tutor AND ASIGNADO.cuenta=ALUMNO.cuenta";
mysql_query($marcaalumno);
}
};
};
}
?>
```

La siguientes funciones **asignacion2a** y **asignacion2b** lo que hacen es asignar los tutores que hay de cada división (de la 2 a la 5), que No tengan **restricción** de que el alumno **sea de su misma carrera,** la única restricción es que los alumnos **sean de su misma división**.

En estas dos funciones es donde **se asignan a la mayoría de los alumnos**, ya que al término de su ejecución se han agotado los tutores y **faltan casi la mitad de los alumnos por asignar**.

El código comentado se presenta a continuación.

```
<?php

Function asignacion2a()

{

ini_set('max_execution_time','480');

//Dos ciclos uno para la división 2 y otro para la 3
for($i=2; $i<=3; $i=$i+1){

//Cuento cúantos profesores hay de esta división
$cuenta = mysql_query("SELECT no_tutor FROM TUTOR WHERE clave_division=$i");
$cuentotut = mysql_num_rows($cuenta);

//Entro al ciclo que durará hasta el número de tutores de la división
for($x=1; $x<=$cuentotut; $x=$x+1){
$seltutor2 = "SELECT no_tutor, nombre_ap, no_hombres, no_mujeres FROM TUTOR
WHERE clave_division=$i AND asignado=0 ORDER BY RAND() LIMIT 0,1";
$result = mysql_query($seltutor2);

//Verificamos si la consulta tiene algún error
        if (! $result){
echo "La consulta SQL contiene errores.";
exit();
}

//Si no tiene errores le asignamos a cada campo una variable
While($row=mysql_fetch_object($result))
{
  //le asignamos variables a las columnas
  $no_tutor=$row->no_tutor;
```

```
  $nombre_ap=$row->nombre_ap;
  $no_hombres=$row->no_hombres;
  $no_mujeres=$row->no_mujeres;

}
//Le asigno sus alumnas al tutor
$asigmujeres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=$i AND sexo=2
AND asignado!=1
ORDER BY bloque
LIMIT 0,".$no_mujeres."";
$mujeres = mysql_query($asigmujeres);

//Le asigno sus alumnos al tutor
$asighombres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=$i AND sexo=1
AND asignado!=1
ORDER BY bloque
LIMIT 0,".$no_hombres."";
$hombres = mysql_query($asighombres);

//Les asigno a todos los alumnos su número de tutor
$asigtutor = "UPDATE ASIGNADO SET no_tutor=$no_tutor WHERE no_tutor=0";
mysql_query($asigtutor);
$asigtutor2 = "UPDATE ASIGNADO SET nombre_ap=$nombre_ap WHERE
no_tutor=$no_tutor";
mysql_query($asigtutor2);

//Le pongo marca asignado=1 para saber que este tutor ya fue asignado
$marcatutor = "UPDATE TUTOR SET asignado=1 WHERE no_tutor=$no_tutor";
$tutor2 = mysql_query($marcatutor);

//Le pongo marca asignado=1 para saber que este alumno ya fue asignado
$marcaalumno = "UPDATE ALUMNO, ASIGNADO SET ALUMNO.asignado=1
WHERE ASIGNADO.no_tutor=$no_tutor AND ASIGNADO.cuenta=ALUMNO.cuenta";
$alumno = mysql_query($marcaalumno);

};
};
}?>
```

```
<?php

Function asignacion2b()

{

ini_set('max_execution_time','480');

//Dos ciclos uno para la división 4 y otro para la 5

for($i=4; $i<=5; $i=$i+1){

//Cuento cúantos profesores hay de esta división
$cuenta = mysql_query("SELECT no_tutor FROM TUTOR WHERE clave_division=$i");
$cuentotut = mysql_num_rows($cuenta);

//Entro al ciclo que durará hasta el número de tutores de la división
for($x=1; $x<=$cuentotut; $x=$x+1){
$seltutor2 = "SELECT no_tutor, nombre_ap, no_hombres, no_mujeres FROM TUTOR
WHERE clave_division=$i AND asignado=0 ORDER BY RAND() LIMIT 0,1";
$result = mysql_query($seltutor2);


//Verificamos si la consulta tiene algún error
        if (! $result){
echo "La consulta SQL contiene errores.";
exit();
}

//Si no tiene errores le asignamos a cada campo una variable

While($row=mysql_fetch_object($result))

{
  //le asignamos variables a las columnas
  $no_tutor=$row->no_tutor;
  $nombre_ap=$row->nombre_ap;
  $no_hombres=$row->no_hombres;
  $no_mujeres=$row->no_mujeres;

}
```

```
//Le asigno sus alumnas al tutor
$asigmujeres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=$i AND sexo=2
AND asignado!=1
ORDER BY bloque
LIMIT 0,".$no_mujeres."";
$mujeres = mysql_query($asigmujeres);

//Le asigno sus alumnos al tutor
$asighombres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=$i AND sexo=1
AND asignado!=1
ORDER BY bloque
LIMIT 0,".$no_hombres."";
$hombres = mysql_query($asighombres);

//Les asigno a todos los alumnos su número de tutor
$asigtutor = "UPDATE ASIGNADO SET no_tutor=$no_tutor WHERE no_tutor=0";
mysql_query($asigtutor);
$asigtutor2 = "UPDATE ASIGNADO SET nombre_ap=$nombre_ap WHERE
no_tutor=$no_tutor";
mysql_query($asigtutor2);

//Le pongo marca asignado=1 para saber que este tutor ya fue asignado
$marcatutor = "UPDATE TUTOR SET asignado=1 WHERE no_tutor=$no_tutor";
$tutor2 = mysql_query($marcatutor);

//Le pongo marca asignado=1 para saber que este alumno ya fue asignado
$marcaalumno = "UPDATE ALUMNO, ASIGNADO SET ALUMNO.asignado=1
WHERE ASIGNADO.no_tutor=$no_tutor AND ASIGNADO.cuenta=ALUMNO.cuenta";
$alumno = mysql_query($marcaalumno);
};
};
}?>
```

Llegando a este punto todos los alumnos de la división 2 (Ingeniería Civil, Topográfica y Geomatica) ya han sido asignados, pero faltan muchos alumnos de las otras divisiones (3,4

y 5) y solo quedan tutores de las divisones 1 y 6 que son División de Ciencias Básicas y División de Ciencias Sociales y Humanidades, repectivamente.

En la siguiente función **asignacion3a,** se asignaran los alumnos que faltan en la división de Ingeniería en Ciencias de la Tierra (división 3), para lograr esto el siguiente programa busca que alumnos de la división 3 quedan sin asignar y los asigna a los tutores de la división 1 (División de Ciencias Básicas).

```
<?php

Function asignacion3a()

{

//En esta sección se asignaran los alumnos que faltan en
//la División de Ingeniería en Ciencias de la Tierra (división 3)

//Cuento cúantos alumnos quedan sin asignar en la división 3
$cuenta = mysql_query("SELECT nombre FROM ALUMNO WHERE asignado!=1 AND clave_division=3 AND cuenta!=0");
$cuentoalum = mysql_num_rows($cuenta);

//Entro al ciclo que durará hasta el número de alumnos que quedan en esta división dividido
//entre el número de alumnos que le tocan a cada tutor que es alrededor de 10
$M=$cuentoalum/10;
for($x=1; $x<=$M; $x=$x+1){

//Selecciono al azar un tutor de la División de Ciencias Básicas que no haya sido asignado
$seltutor2 = "SELECT no_tutor, nombre_ap, no_hombres, no_mujeres FROM TUTOR
WHERE clave_division=1 AND asignado!=1 AND no_tutor!=0 ORDER BY RAND()
LIMIT 0,1";
$result = mysql_query($seltutor2);

//Verificamos si la consulta tiene algún error
        if (! $result){
echo "La consulta SQL contiene errores.";
exit();
}

//Si no tiene errores le asignamos una variable a cada campo del arreglo
While($row=mysql_fetch_object($result))
{

  $no_tutor=$row->no_tutor;
  $nombre_ap=$row->nombre_ap;
```

```
  $no_hombres=$row->no_hombres;
  $no_mujeres=$row->no_mujeres;
}


//Si la consulta ya no tiene valores salimos del ciclo
if (empty($no_tutor)){
break;}

//Le asigno sus alumnas al tutor
$asigmujeres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=3 AND sexo=2
AND asignado!=1 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$no_mujeres."";
$mujeres = mysql_query($asigmujeres);

//Le asigno sus alumnos al tutor
$asighombres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=3 AND sexo=1
AND asignado!=1 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$no_hombres."";
$hombres = mysql_query($asighombres);

//Les asigno a todos los alumnos asignados en esta iteración su número de tutor
$asigtutor = "UPDATE ASIGNADO SET no_tutor=$no_tutor WHERE no_tutor=0";
mysql_query($asigtutor);
$asigtutor2 = "UPDATE ASIGNADO SET nombre_ap=$nombre_ap WHERE
no_tutor=$no_tutor";
mysql_query($asigtutor2);

//Le pongo marca asignado=1 para saber que este tutor ya fue asignado
$marcatutor = "UPDATE TUTOR SET asignado=1 WHERE no_tutor=$no_tutor";
$tutor2 = mysql_query($marcatutor);

//Le pongo marca asignado=1 para saber que este alumno ya fue asignado
$marcaalumno = "UPDATE ALUMNO, ASIGNADO SET ALUMNO.asignado=1
WHERE ASIGNADO.no_tutor=$no_tutor AND ASIGNADO.cuenta=ALUMNO.cuenta";
$alumno = mysql_query($marcaalumno);
};
}
?>
```

En la función **asignacion3b** se van a asignar todos los tutores que faltan de asignar de la División de Ciencias Básicas (división 1) se les asignarán alumnos de la División de Ingeniería Eléctrica (4), debido que los de la división 3 ya se han acabado de asignar.

```
<?php

Function asignacion3b()

{

//En esta sección se va a asignar todos los  tutores que faltan de asignar de la División de Ciencias Basicas (división 1)

//Cuento cuántos profesores quedan sin asignar en la División de Ciencias Básicas
$cuenta = mysql_query("SELECT nombre FROM TUTOR WHERE asignado!=1 AND clave_division=1 AND no_tutor!=0");
$cuentotut = mysql_num_rows($cuenta);
/
/Entro al ciclo que durará hasta el número de tutores encontrados
for($x=1; $x<=$cuentotut; $x=$x+1){

//Selecciono al azar un tutor de la División de Ciencias Básicas que no haya sido asignado
$seltutor2 = "SELECT no_tutor, nombre_ap, no_hombres, no_mujeres FROM TUTOR
WHERE clave_division=1 AND asignado!=1 AND no_tutor!=0 ORDER BY RAND()
LIMIT 0,1";
$result = mysql_query($seltutor2);

//Verificamos si la consulta tiene algún error
        if (! $result){
echo "La consulta SQL contiene errores.";
exit();
}

//Si no tiene errores  se le asigna una variable a cada campo del arreglo
While($row=mysql_fetch_object($result))
{

  $no_tutor=$row->no_tutor;
  $nombre_ap=$row->nombre_ap;
  $no_hombres=$row->no_hombres;
  $no_mujeres=$row->no_mujeres;

}

//Si la consulta ya no tiene valores sale del ciclo
```

```
if (empty($no_tutor)){
break;}



//Le asigno sus alumnas al tutor
$asigmujeres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=4 AND sexo=2
AND asignado!=1 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$no_mujeres."";
$mujeres = mysql_query($asigmujeres);

//Le asigno sus alumnos al tutor
$asighombres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=4 AND sexo=1
AND asignado!=1 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$no_hombres."";
$hombres = mysql_query($asighombres);

//Les asigno a todos los alumnos asignados en esta iteración su número de tutor
$asigtutor = "UPDATE ASIGNADO SET no_tutor=$no_tutor WHERE no_tutor=0";
mysql_query($asigtutor);
$asigtutor2 = "UPDATE ASIGNADO SET nombre_ap=$nombre_ap WHERE
no_tutor=$no_tutor";
mysql_query($asigtutor2);

//Le pongo marca asignado=1 para saber que este tutor ya fue asignado
$marcatutor = "UPDATE TUTOR SET asignado=1 WHERE no_tutor=$no_tutor";
$tutor2 = mysql_query($marcatutor);

//Le pongo marca asignado=1 para saber que este alumno ya fue asignado
$marcaalumno = "UPDATE ALUMNO, ASIGNADO SET ALUMNO.asignado=1
WHERE ASIGNADO.no_tutor=$no_tutor AND ASIGNADO.cuenta=ALUMNO.cuenta";
$alumno = mysql_query($marcaalumno);

};

}
?>
```

La función **asignacion4a** asignara los alumnos que faltan de la División de Ingeniería Eléctrica (división 4), asignándoles tutores de la División 6 de Ciencias Sociales y Humanidades que son los únicos que quedan sin asignar.

```
<?php

Function asignacion4a()

{

//En esta sección se asignaran los alumnos que faltan en
//la división de Ingeniería Eléctrica (división 4)

//Cuento cúantos alumnos faltan de asignar en esta división
$cuenta = mysql_query("SELECT nombre FROM ALUMNO WHERE asignado!=1 AND
clave_division=4 AND cuenta!=0");
$cuentoalum = mysql_num_rows($cuenta);

//Entro al ciclo que durará hasta el número de alumnos que quedan en esta división dividido
//entre el número de alumnos que le tocan a cada tutor que a estas alturas es alrededor de 8
$M=$cuentoalum/8;
for($x=1; $x<=$M; $x=$x+1){

//Selecciono al azar un tutor de la División de Ciencias Sociales y Humanidades que no
//haya sido asignado
$seltutor2 = "SELECT no_tutor, nombre_ap, no_hombres, no_mujeres FROM TUTOR
WHERE clave_division=6 AND asignado!=1 AND no_tutor!=0 ORDER BY RAND()
LIMIT 0,1";
$result = mysql_query($seltutor2);

//Verificamos si la consulta tiene algún error
        if (! $result){
echo "La consulta SQL contiene errores.";
exit();
}

//Si no tiene errores le asignamos una variable a cada campo del arreglo
While($row=mysql_fetch_object($result))
{

  $no_tutor=$row->no_tutor;
  $nombre_ap=$row->nombre_ap;
  $no_hombres=$row->no_hombres;
```

```
  $no_mujeres=$row->no_mujeres;

}
//Si la consulta ya no tiene valores salimos del ciclo
if (empty($no_tutor)){
break;}

//Le asigno sus alumnas al tutor
$asigmujeres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=4 AND sexo=2
AND asignado!=1 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$no_mujeres."";
$mujeres = mysql_query($asigmujeres);

//Le asigno sus alumnos al tutor
$asighombres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=4 AND sexo=1
AND asignado!=1 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$no_hombres."";
$hombres = mysql_query($asighombres);

///Les asigno a todos los alumnos asignados en esta iteración su número de tutor
$asigtutor = "UPDATE ASIGNADO SET no_tutor=$no_tutor WHERE no_tutor=0";
mysql_query($asigtutor);
$asigtutor2 = "UPDATE ASIGNADO SET nombre_ap=$nombre_ap WHERE
no_tutor=$no_tutor";
mysql_query($asigtutor2);

//Le pongo marca asignado=1 para saber que este tutor ya fue asignado
$marcatutor = "UPDATE TUTOR SET asignado=1 WHERE no_tutor=$no_tutor";
$tutor2 = mysql_query($marcatutor);

//Le pongo marca asignado=1 para saber que este alumno ya fue asignado
$marcaalumno = "UPDATE ALUMNO, ASIGNADO SET ALUMNO.asignado=1
WHERE ASIGNADO.no_tutor=$no_tutor AND ASIGNADO.cuenta=ALUMNO.cuenta";
$alumno = mysql_query($marcaalumno);

};
}
?>
```

En la función **asignacion4b** se van a asignar todos los tutores que faltan de asignar de la División de Ciencias Sociales y Humanidades (división 6).

A estos tutores se les asignarán alumnos de la división 5 División de Ingeniería Mecánica e Industrial, ya que estos alumnos son los únicos que faltan por asignar.

```
<?php

Function asignacion4b()

{


//En esta sección se va a asignar todos los  Tutores que faltan de asignar de la División de Ciencias Sociales y Humanidades  (división 6)

//Cuento cuantos profesores quedan sin asignar en la División de Ciencias Sociales y Humanidades
$cuenta = mysql_query("SELECT nombre FROM TUTOR WHERE asignado!=1 AND clave_division=6 AND no_tutor!=0");
$cuentotut = mysql_num_rows($cuenta);

//Entro al ciclo que durará hasta el número de tutores encontrados
for($x=1; $x<=$cuentotut; $x=$x+1){

//Selecciono al azar un tutor de la División de Ciencias Sociales y Humanidades que no
//haya sido asignado
$seltutor2 = "SELECT no_tutor, nombre_ap, no_hombres, no_mujeres FROM TUTOR
WHERE clave_division=6 AND asignado!=1 AND no_tutor!=0 ORDER BY RAND()
LIMIT 0,1";
$result = mysql_query($seltutor2);

//Verificamos si la consulta tiene algún error
        if (! $result){
echo "La consulta SQL contiene errores.";
exit();
}

//Si no tiene errores le asignamos una variable a cada campo del arreglo
While($row=mysql_fetch_object($result))
{

  $no_tutor=$row->no_tutor;
  $nombre_ap=$row->nombre_ap;
  $no_hombres=$row->no_hombres;
```

```
  $no_mujeres=$row->no_mujeres;

}

//Si la consulta ya no tiene valores salimos del ciclo
if (empty($no_tutor)){
break;}

//Le asigno sus alumnas al tutor
$asigmujeres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=5 AND sexo=2
AND asignado!=1 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$no_mujeres."";
$mujeres = mysql_query($asigmujeres);

//Le asigno sus alumnos al tutor
$asighombres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=5 AND sexo=1
AND asignado!=1 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$no_hombres."";
$hombres = mysql_query($asighombres);

//Les asigno a todos los alumnos asignados en esta iteración su número de tutor
$asigtutor = "UPDATE ASIGNADO SET no_tutor=$no_tutor WHERE no_tutor=0";
mysql_query($asigtutor);
$asigtutor2 = "UPDATE ASIGNADO SET nombre_ap=$nombre_ap WHERE
no_tutor=$no_tutor";
mysql_query($asigtutor2);

//Le pongo marca asignado=1 para saber que este tutor ya fue asignado
$marcatutor = "UPDATE TUTOR SET asignado=1 WHERE no_tutor=$no_tutor";
$tutor2 = mysql_query($marcatutor);

//Le pongo marca asignado=1 para saber que este alumno ya fue asignado
$marcaalumno = "UPDATE ALUMNO, ASIGNADO SET ALUMNO.asignado=1
WHERE ASIGNADO.no_tutor=$no_tutor AND ASIGNADO.cuenta=ALUMNO.cuenta";
$alumno = mysql_query($marcaalumno);

};

}
?>
```

La última función es **asignacion5** y es donde se asignaran los alumnos que faltan, que son de la División de Ingeniería Mecánica e Industrial (división 5).

A estos alumnos se les tiene que asignar tutores que ya han sido asignados, debido a que no fue posible asignarlos de primera intención, estos alumnos se les asignan a los tutores participativos cuyo campo **clave_obs (clave de Observación)** se igual á **aa.**

```
<?php

Function asignacion5()

{

//En esta sección se asignaran los alumnos que faltan en
//la división de Ing. Mecánica Industrial (división 5)

//Cuento cúantos alumnos faltan de asignar en esta división
$cuenta = mysql_query("SELECT nombre FROM ALUMNO WHERE clave_division=5 AND asignado=0 AND cuenta!=0");
$cuentoalum = mysql_num_rows($cuenta);

//Entro al ciclo que durará hasta el número de alumnos que quedan en esta división dividido
//entre el número de

//Alumnos que le tocan a cada tutor que es alrededor de 3
$M=$cuentoalum/3;
for($x=1; $x<=$M; $x=$x+1){

//Selecciono al azar un tutor de la División de Ing. Eléctrica que sea Participativo (aa) y
//que no haya sido asignado
$seltutor2 = "SELECT no_tutor, nombre_ap, no_hombres, no_mujeres FROM TUTOR
WHERE clave_obs='aa' AND observaciones!=" AND clave_division=4 AND no_tutor!=0
AND asignado!=2 ORDER BY RAND() LIMIT 0,1";
$result = mysql_query($seltutor2);

//Verificamos si la consulta tiene algún error
        if (! $result){
echo "La consulta SQL contiene errores.";
exit();
}

//Si no tiene errores le asignamos una variable a cada campo del arreglo
While($row=mysql_fetch_object($result))
{

  $no_tutor=$row->no_tutor;
```

```
  $nombre_ap=$row->nombre_ap;
  $no_hombres=$row->no_hombres;
  $no_mujeres=$row->no_mujeres;

}


//si la consulta ya no tiene valores salimos del ciclo
if (empty($no_tutor)){
break;}


$M2=$no_mujeres/3;
$M3=$no_hombres/3;

//Le asigno sus alumnas al tutor
$asigmujeres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=5 AND sexo=2
AND asignado=0 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$M2."";
$mujeres = mysql_query($asigmujeres);

//Le asigno sus alumnos al tutor
$asighombres = "INSERT INTO ASIGNADO (cuenta, clave_carrera,
bloque, sexo, clave_division) SELECT cuenta, clave_carrera,
bloque, sexo, clave_division FROM ALUMNO WHERE clave_division=5 AND sexo=1
AND asignado=0 AND cuenta!=0
ORDER BY bloque
LIMIT 0,".$M3."";
$hombres = mysql_query($asighombres);

//Les asigno a todos los alumnos asignados en esta iteración su número de tutor
$asigtutor = "UPDATE ASIGNADO SET no_tutor=$no_tutor WHERE no_tutor=0";
mysql_query($asigtutor);
$asigtutor2 = "UPDATE ASIGNADO SET nombre_ap=$nombre_ap WHERE
no_tutor=$no_tutor";
mysql_query($asigtutor2);

//Le pongo marca asignado=1 para saber que este tutor ya fue asignado
$marcatutor = "UPDATE TUTOR SET asignado=2 WHERE no_tutor=$no_tutor";
$tutor2 = mysql_query($marcatutor);

//Le pongo marca asignado=1 para saber que este alumno ya fue asignado
$marcaalumno = "UPDATE ALUMNO, ASIGNADO SET ALUMNO.asignado=1
WHERE ASIGNADO.no_tutor=$no_tutor AND ASIGNADO.cuenta=ALUMNO.cuenta";
```

```
$alumno = mysql_query($marcaalumno);

};

}
?>
```

A continuación se presenta la programación para presentar los resultados de la asignación, utilizando una consulta a la tabla **ASIGNADOS** y mostrando el resultado, dando las opciones de mostrar por nombre de división o completa.

Ésto se realiza con una paginación de resultados y un ciclo para presentar estos en columnas dándole colores y diseño con **PHP** y **HTML**.

```
<html>
<head>
<title>Resultado de la Asignación</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="Pragma" content="no-cache" />
<style type="text/css">
<!--
a.p:link {
   color: #0066FF;
   text-decoration: none;
}
a.p:visited {
   color: #0066FF;
   text-decoration: none;
}
a.p:active {
   color: #0066FF;
   text-decoration: none;
}
a.p:hover {
   color: #0066FF;
   text-decoration: underline;
}
a.ord:link {
   color: #000000;
   text-decoration: none;
}
a.ord:visited {
   color: #000000;
   text-decoration: none;
}
```

```
a.ord:active {
   color: #000000;
   text-decoration: none;
}
a.ord:hover {
   color: #000000;
   text-decoration: underline;
}
-->
</style>
</head>
<body bgcolor="#FFFFFF">
<center><br><br>
 <br>
 <img src="Imagenes/logocopadi.jpg" width="128" height="130" align="left"><font color="#0000CC" size="7" face="Times New Roman, Times, serif"><img src="Imagenes/escudoazul.GIF" width="176" height="197" align="right" ></font>
 <font color="#0000CC" size="7" face="Times New Roman, Times, serif">S I A T T </font>
 <p><font color="#0000CC" size="5" face="Times New Roman, Times, serif">Sistema de Informaci&oacute;n para la Asignaci&oacute;n de tutores en el &Aacute;rea de Tutor&iacute;a para Todos</font></p>
 <br>
<script language="JavaScript">
function muestra(queCosa)
{
   alert(queCosa);
}
</script>
<div align="center"><strong><font color="#000000" size="2" face="Verdana, Arial, Helvetica, sans-serif">Resultados de la Asignación<br><br></font></strong> </div>
<hr noshade style="color:CC6666;height:1px">
<br>
<?
//inicializo el criterio y recibo cualquier cadena que se desee buscar
$criterio = "";
$txt_criterio = "";
if ($_GET["criterio"]!=""){
  $txt_criterio = $_GET["criterio"];
  $criterio = "AND ALUMNO.clave_division like '%" . $txt_criterio . "%'";
}


$sql="SELECT ALUMNO . * , ASIGNADO.no_tutor FROM ALUMNO, ASIGNADO WHERE ALUMNO.cuenta = ASIGNADO.cuenta ".$criterio;
$res=mysql_query($sql);
$númeroRegistros=mysql_num_rows($res);
```

```
if($númeroRegistros<=0)
{
   echo "<div align='center'>";
   echo "<font face='verdana' size='-2'>No se encontraron resultados</font>";
   echo "</div>";
}else{
   //////////elementos para el orden
   if(!isset($orden))
   {
     $orden="no_tutor";
   }
   //////////fin elementos de orden

   //////////cálculo de elementos necesarios para paginación
   //tamaño de la página
   $tamPag=30;

   //pagina actual si no esta definida y límites
   if(!isset($_GET["pagina"]))
   {
     $pagina=1;
     $inicio=1;
     $final=$tamPag;
   }else{
     $pagina = $_GET["pagina"];
   }
   //cálculo del límite inferior
   $limitInf=($pagina-1)*$tamPag;

   //cálculo del número de paginas
   $numPags=ceil($númeroRegistros/$tamPag);
   if(!isset($pagina))
   {
     $pagina=1;
     $inicio=1;
     $final=$tamPag;
   }else{
     $seccionActual=intval(($pagina-1)/$tamPag);
     $inicio=($seccionActual*$tamPag)+1;

     if($pagina<$numPags)
     {
       $final=$inicio+$tamPag-1;
     }else{
       $final=$numPags;
     }
```

```
      if ($final>$numPags){
        $final=$numPags;
      }
    }

//////////fin de dicho cálculo

//////////creacion de la consulta con limites
$sql="SELECT ALUMNO . * , ASIGNADO.no_tutor FROM ALUMNO, ASIGNADO WHERE ALUMNO.cuenta = ASIGNADO.cuenta ".$criterio." ORDER BY ".$orden." ASC LIMIT ".$limitInf.",".$tamPag;
$res=mysql_query($sql);

//////////fin consulta con limites
echo "<div align='center'>";
echo "<font face='verdana' size='-2'>Se encontraron <b>".$númeroRegistros."</b> resultados<br>";
echo "Ordenados por: <b>".$orden."</b>";
if(isset($txt_criterio)){
   echo "<br>No de División: <b>".$txt_criterio."</b>";
}
echo "</font></div>";
echo "<table align='center' width='80%' border='0' cellspacing='1' cellpadding='0'>";
echo "<tr><td colspan='3'><hr noshade></td></tr>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=cuenta=".$txt_criterio."'>Cuenta</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=nombre=".$txt_criterio."'>Nombre</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=clave_carrera=".$txt_criterio."'>Clave Carrera</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=carrera=".$txt_criterio."'>Carrera</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=bloque=".$txt_criterio."'>Bloque</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=sexo=".$txt_criterio."'>Sexo</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=clave_division=".$txt_criterio."'>Clave_ Division</a></th>";
echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=no_tutor=".$txt_criterio."'>No Tutor</a></th>";
//echo "<th bgcolor='#CCCCCC'><a class='ord' href='".$_SERVER["PHP_SELF"]."?pagina = ".$pagina."&orden=nombre_ap=".$txt_criterio."'>Nombre del Tutor</a></th>";
while($registro=mysql_fetch_array($res))
{
?>
  <!-- tabla de resultados -->
```

```
   <tr bgcolor="#EEEEFF"
onMouseOver="this.style.backgroundColor='#FFFF99';this.style.cursor='hand';"
onMouseOut="this.style.backgroundColor='#EEEEFF'"o"];"
onClick="javascript:muestra('<? echo "[".$registro["cuenta"]."] ".$registro["nombre"]." -
".$registro["clave_carrera"]." - ".$registro["carrera"]." - ".$registro["bloque"]." -
".$registro["sexo"]." - ".$registro["clave_division"]." - ".$registro["no_tutor"]." -
".$registro["nombre_ap"]; ?>');">
   <td><center><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["cuenta"]; ?></b></font></center></td>
   <td><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["nombre"]; ?></b></font></td>
   <td><center><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["clave_carrera"]; ?></b></font></center></td>
   <td><center><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["carrera"]; ?></b></font></center></td>
   <td><center><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["bloque"]; ?></b></font><center></td>
   <td><center><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["sexo"]; ?></b></font></center></td>
   <td><center><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["clave_division"]; ?></b></font></center></td>
   <td><center><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["no_tutor"]; ?></b></font></center></td>
   <td><center><font size="2" face="Verdana, Arial, Helvetica, sans-serif"
color="#000000"><b><? echo $registro["nombre_ap"]; ?></b></font></center></td>
   </tr>
  <!-- fin tabla resultados -->
<?
}//fin while
echo "</table>";
}//fin if
//////////a partir de aqui viene la paginación
?>
   <br>
   <table border="0" cellspacing="0" cellpadding="0" align="center">
   <tr><td align="center" valign="top">
<?
   if($pagina>1)
   {
     echo "<a class='p' href='".$_SERVER["PHP_SELF"]."?pagina=".($pagina-
1)."&orden=".$orden."&criterio=".$txt_criterio."'>";
     echo "<font face='verdana' size='-2'>anterior</font>";
     echo "</a> ";
   }

   for($i=$inicio;$i<=$final;$i++)
   {
```

```
      if($i==$pagina)
      {
        echo "<font face='verdana' size='-2'><b>".$i."</b> </font>";
      }else{
        echo "<a class='p' href='".$_SERVER["PHP_SELF"]."?pagina=".$i."&orden=".$orden."&criterio=".$txt_criterio."'>";
        echo "<font face='verdana' size='-2'>".$i."</font></a> ";
      }
    }
    if($pagina<$numPags)
   {
      echo " <a class='p' href='".$_SERVER["PHP_SELF"]."?pagina=".($pagina+1)."&orden=".$orden."&criterio=".$txt_criterio."'>";
      echo "<font face='verdana' size='-2'>siguiente</font></a>";
   }
//////////fin de la paginación
?>
    </td></tr>
    </table>
<hr noshade style="color:CC6666;height:1px">
<div align="center"><font face="verdana" size="-2"><a class="p" href="paginación2.php">::Inicio::</a></font></div>
<br><br><center>
<form action="paginación.php" method="get">
Criterio de búsqueda:<br>Por División:


<select name="criterio" size="2" id="criterio">
       <option value="NULL" selected></option>
       <option value="2">División de Ingeniería Civil Topográfica y Geodesta</option>
       <option value="3">División de Ingeniería en Ciencias de la Tierra</option>
          <option value="4">División de Ingeniería Eléctrica</option>
          <option value="5">División de Ingeniería Mecánica Industrial</option>
                    </select>
<input type="submit" value="Buscar">
</form></center>
<td><form name="form1" method="post" action="asignar.php">
        <div align="right">
          <input type="submit" name="Submit1" value="   Regresar a Menú de Asignación ">
        </div>
       </form></td>
</body>
</html>
<?
```

```
   mysql_close($conexion);
?>
```

La programación se realizó de forma modular, con una gran atención en lo que se refiere a la unión de las diferentas partes que integran el sistema.

Esta unión se llevó a cabo con éxito, debido principalmente a que se diseño un Modelado del sistema y a que se siguió en tiempo y forma nuestro modelo en Espiral planteado en el Capítulo 4.

Ya hemos realizado la integración del sistema, sólo nos falta hacerle pruebas de operación tiempos, y posibles errores de programación. En el siguiente capítulo se practicaran estas pruebas.

CAPÍTULO 7

Pruebas, Análisis de Resultados y Conclusiones

7.1 Pruebas y Análisis de Resultados

Es importante definir que es una prueba de software, en este trabajo de tesis, ya que es un conjunto de actividades o pasos que se planifican por adelantado y se llevan a cabo sistemáticamente, todo esto con el objetivo de llevar a cabo una correcta construcción del software.

La prueba debe estar integrada por, **la planificación**, el **diseño de casos de prueba**, la **ejecución de las pruebas** y la **agrupación y evaluación de los datos** resultantes.

El sistema que hemos desarrollado (**SIATT**), fue diseñado bajo una arquitectura cliente/servidor, por lo que la planificación de cómo se llevaran a cabo las pruebas depende mucho de no perder de vista este enfoque.

En general, las pruebas de software de las arquitecturas cliente/servidor se producen en tres niveles diferentes:

1) Las aplicaciones de cliente individuales se comprueban de modo “desconectado” (el funcionamiento del servidor y la red no se consideran).

2) Las aplicaciones de software de cliente y del servidor saciado se prueban al mismo tiempo, pero no se ejecutan operaciones de red.

3) Se comprueba la arquitectura cliente/servidor completa, incluyendo el rendimiento y funcionamiento de la red.

Para este trabajo, el nivel de prueba más significativo, es el tercero, debido a que el **SIATT** trabaja bajo una arquitectura cliente/servidor, pero el servidor se encuentra en la misma máquina en la que se ejecuta el cliente, por tanto es factible hacer una prueba que abarque todo el conjunto y verificar el buen funcionamiento de ambos al mismo tiempo.

Aun cuando se efectúen varias clases distintas de pruebas en cada uno de los niveles de detalle anteriores, es frecuente encontrar los siguientes enfoques de comprobación para aplicaciones cliente/servidor.

a) **Comprobaciones de función de aplicación.** Se comprueba la funcionalidad de las aplicaciones cliente empleando los métodos descritos anteriormente en este libro. En esencia, la aplicación se comprueba en solitario en un intento de descubrir errores de su funcionamiento.

b) **Comprobaciones de servidor**. Se comprueban la coordinación y las funciones de gestión de datos del servidor. También se considera el rendimiento del servidor (tiempo de respuesta y trasvase de datos en general).

c) **Comprobaciones de bases de datos.** Se comprueba la precisión e integridad de los datos almacenados en el servidor. Se examinan las transacciones enviadas por las aplicaciones cliente pata asegurar que los datos se almacenen, actualicen y recuperen adecuadamente. También se comprueba el archivado.

d) **Comprobación de transacciones.** Se crea una serie de comprobaciones adecuada para comprobar que todas las clases de transacciones se procesen de acuerdo con los requisitos. Las comprobaciones hacen especial hincapié en la corrección de procesamiento, y también en los temas de rendimiento (p. ej.: tiempo de procesamiento de transacciones y comprobación de volúmenes de transacciones).

e) **Comprobaciones de comunicaciones a través de la red**. Estas comprobaciones verifican que la comunicación entre los nudos de la red se produzca correctamente, y que el paso de mensaje, las transacciones y el tráfico de red relacionado tenga lugar sin errores.

Para llevar a cabo la mayoría de estas comprobaciones, tendremos que hacer uso de un software, que nos mida el rendimiento de nuestro sistema, mientras este se está ejecutando, este software es **phpMyAdmin**, el cual es una interfaz Web para administrar nuestras bases de datos en **MySQL,** dicho software está programado en **PHP**, es de libre distribución y no necesitamos pagar licencias por el mismo.

El programa **phpMyAdmin,** en su sección de **información acerca del tiempo de marcha** puede darnos información del sistema en cualquier momento que lo solicitemos, esta información tiene que ver con el **tráfico en el servidor**, **estadísticas de consultas**, el **tiempo que el servidor MySQL ha estado corriendo**, y **como están las variables en el estado actual**, además de que nos ayuda a desplegarnos gráficamente las tablas y hasta podemos hacer consultas directamente.

Lo que se va a llevar a cabo para comprobar que nuestro sistema funciona correctamente, es ejecutar el programa de asignación de alumnos que se llama asignación.php e inmediatamente se ejecutara el programa **phpMyAdmin** para registrar los datos y analizarlos posteriormente.

La prueba se llevó a cabo en un equipo con las siguientes características:

Notebook Sony

Modelo: Vaio PCG-K43F

Procesador: Intel Celeron D 345 a 3.06 GHzMemoria

Memoria RAM: 512 MB DDR SDRAM

Memoria Caché: 256 KB integrada al segundo nivel

Velocidad del Bus: 533 MHz

Disco duro: 40GB

Sistemas operativos: Microsoft Windows XP Home Edition
Linux SUSE 9.3

El hecho de que el sistema (**SIATT**) no esté construido para utilizar un hardware y un software especificado, tiene su impacto sobre la comprobación del sistema. La naturaleza multiplataforma en red de los sistemas cliente/servidor requiere que se preste bastante atención a la comprobación de configuraciones y a la comprobación de compatibilidades.

En nuestro caso la instalación de todas la herramientas (el servidor **Apache**, el manejador de bases de datos **MySQL** y el lenguaje **PHP)** se llevó a cabo de manera exitosa, comprobamos que las herramientas funcionan perfectamente en ambos sistemas operativos (Windows XP y Linux SUSE 9.3)

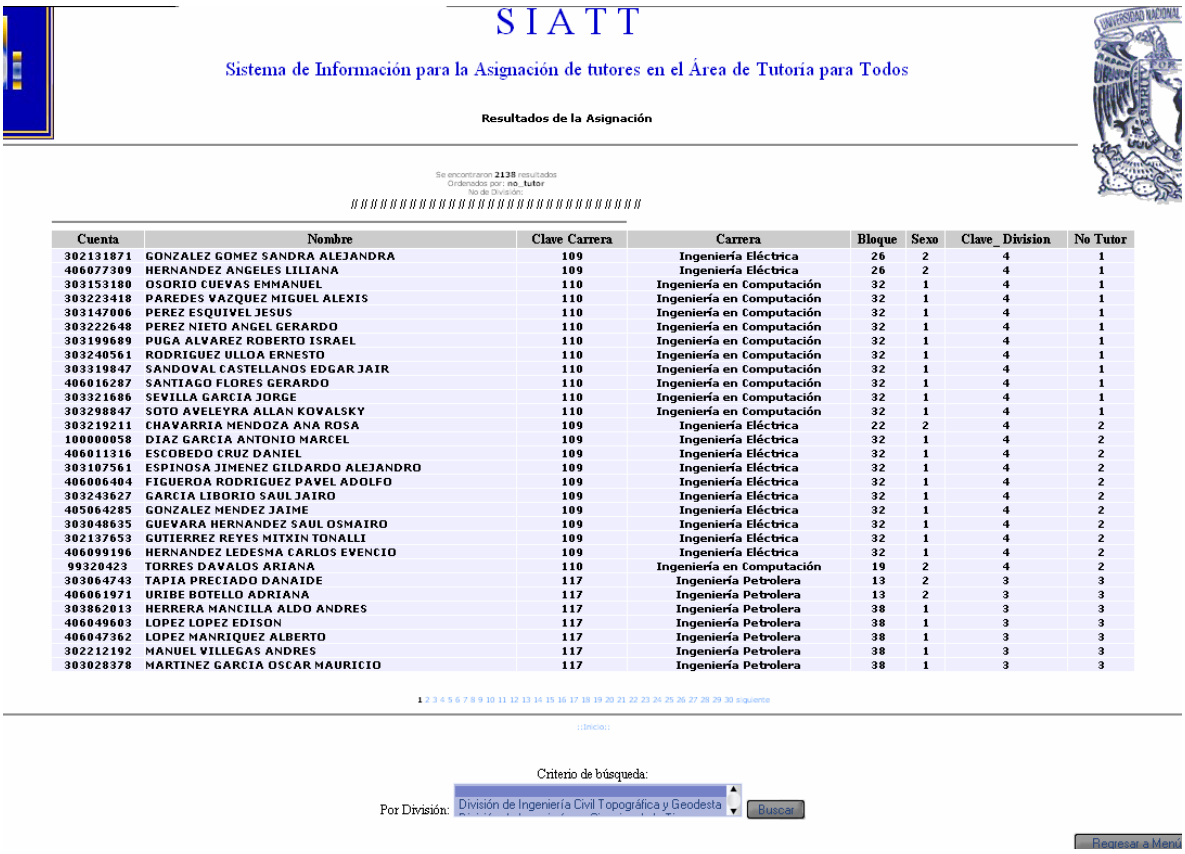
Se instaló el sistema **SIATT** en ambos sistemas operativos y se puso en funcionamiento, se subió la base de datos asignación y se comprobó que todos os registros estuvieran como aparecen en el archivo original de **EXCEL**.

Al trabajar con el sistema **SIATT** en ambos sistemas operativos, la comprobación de compatibilidades asegura una interfaz funcionalmente consistente entre plataformas de software y de hardware, por lo tanto la interfaz de ventanas puede ser visualmente distinta,

dependiendo del entorno de implementación (sistema operativo), pero los mismos comportamientos básicos del usuario deberán dar lugar a los mismos resultados, independientemente de si es Linux, Windows, OS/2, Unix etc.

Nuestro sistema trabaja de la misma forma en Linux o en Windows, lo que difiere en cada uno es la velocidad de procesamiento.

A continuación se presenta la información de tiempo de marcha de nuestro programa de asignación de alumnos (asignación.php) en el sistema operativo Windows XP.



SIATT

Sistema de Información para la Asignación de tutores en el Área de Tutoría para Todos

Resultados de la Asignación

Se encontraron 2138 resultados
Ordenados por: no_tutor
No de División:

Cuenta	Nombre	Clave Carrera	Carrera	Bloque	Sexo	Clave_Division	No Tutor
302131871	GONZALEZ GOMEZ SANDRA ALEJANDRA	109	Ingeniería Eléctrica	26	2	4	1
406077309	HERNANDEZ ANGELES LILIANA	109	Ingeniería Eléctrica	26	2	4	1
303153180	OSORIO CUEVAS EMMANUEL	110	Ingeniería en Computación	32	1	4	1
303223418	PAREDES VAZQUEZ MIGUEL ALEXIS	110	Ingeniería en Computación	32	1	4	1
303147006	PEREZ ESQUIVEL JESUS	110	Ingeniería en Computación	32	1	4	1
303222648	PEREZ NIETO ANGEL GERARDO	110	Ingeniería en Computación	32	1	4	1
303199689	PUGA ALVAREZ ROBERTO ISRAEL	110	Ingeniería en Computación	32	1	4	1
303240561	RODRIGUEZ ULLOA ERNESTO	110	Ingeniería en Computación	32	1	4	1
303319847	SANDOVAL CASTELLANOS EDGAR JAIR	110	Ingeniería en Computación	32	1	4	1
406016287	SANTIAGO FLORES GERARDO	110	Ingeniería en Computación	32	1	4	1
303321686	SEVILLA GARCIA JORGE	110	Ingeniería en Computación	32	1	4	1
303298847	SOTO AVELEYRA ALLAN KOVALSKY	110	Ingeniería en Computación	32	1	4	1
303219211	CHAVARRIA MENDOZA ANA ROSA	109	Ingeniería Eléctrica	22	2	4	2
100000058	DIAZ GARCIA ANTONIO MARCEL	109	Ingeniería Eléctrica	32	1	4	2
406011316	ESCOBEDO CRUZ DANIEL	109	Ingeniería Eléctrica	32	1	4	2
303107561	ESPINOSA JIMENEZ GILDARDO ALEJANDRO	109	Ingeniería Eléctrica	32	1	4	2
406006404	FIGUEROA RODRIGUEZ PAVEL ADOLFO	109	Ingeniería Eléctrica	32	1	4	2
303243627	GARCIA LIBORIO SAUL JAIRO	109	Ingeniería Eléctrica	32	1	4	2
405064285	GONZALEZ MENDEZ JAIME	109	Ingeniería Eléctrica	32	1	4	2
303048635	GUEVARA HERNANDEZ SAUL OSMAIRO	109	Ingeniería Eléctrica	32	1	4	2
302137653	GUTIERREZ REYES MITXIN TONALLI	109	Ingeniería Eléctrica	32	1	4	2
406099196	HERNANDEZ LEDESMA CARLOS EVENCIO	109	Ingeniería Eléctrica	32	1	4	2
99320423	TORRES DAVALOS ARIANA	110	Ingeniería en Computación	19	2	4	2
303064743	TAPIA PRECIADO DANAIDE	117	Ingeniería Petrolera	13	2	3	3
406061971	URIBE BOTELLO ADRIANA	117	Ingeniería Petrolera	13	2	3	3
303862013	HERRERA MANCILLA ALDO ANDRES	117	Ingeniería Petrolera	38	1	3	3
406049603	LOPEZ LOPEZ EDISON	117	Ingeniería Petrolera	38	1	3	3
406047362	LOPEZ MANRIQUEZ ALBERTO	117	Ingeniería Petrolera	38	1	3	3
302212192	MANUEL VILLEGAS ANDRES	117	Ingeniería Petrolera	38	1	3	3
303028378	MARTINEZ GARCIA OSCAR MAURICIO	117	Ingeniería Petrolera	38	1	3	3

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 siguiente

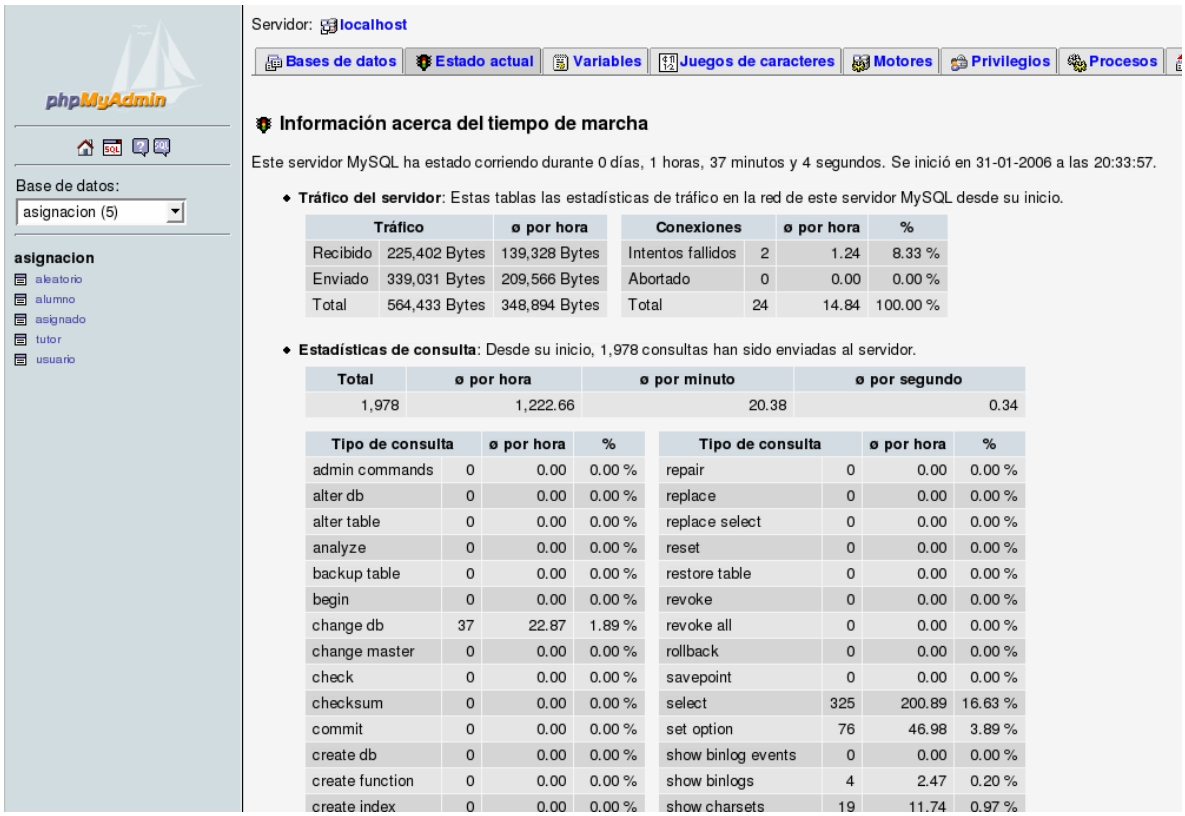
::Inicio::

Criterio de búsqueda:

Por División: División de Ingeniería Civil Topográfica y Geodesta Buscar

Regresar a Menú

Figura 7.1 Asignación realizada por el SIATT corriendo sobre Windows XP



Tráfico		ø por hora
Recibido	225,402 Bytes	139,328 Bytes
Enviado	339,031 Bytes	209,566 Bytes
Total	564,433 Bytes	348,894 Bytes

Conexiones		ø por hora	%
Intentos fallidos	2	1.24	8.33 %
Abortado	0	0.00	0.00 %
Total	24	14.84	100.00 %

Total	ø por hora	ø por minuto	ø por segundo
1,978	1,222.66	20.38	0.34

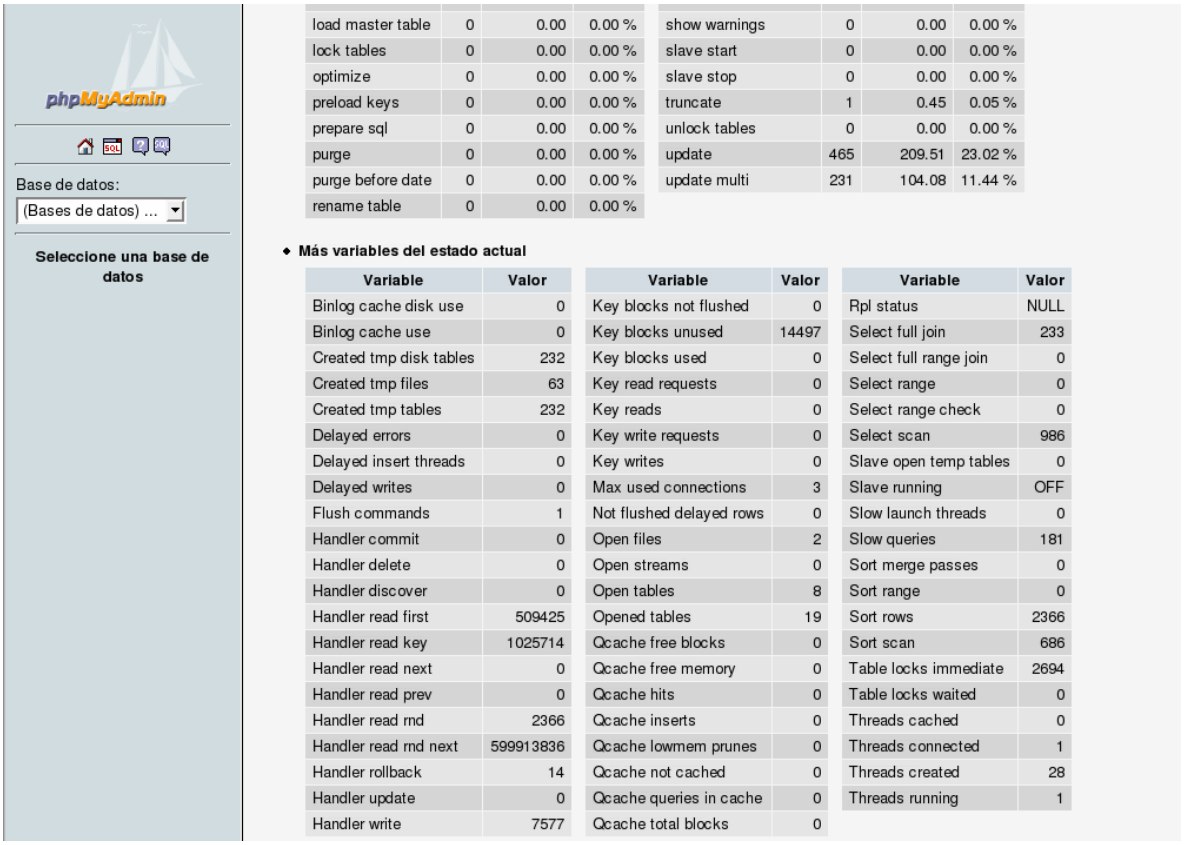
Tipo de consulta		ø por hora	%	Tipo de consulta		ø por hora	%
admin commands	0	0.00	0.00 %	repair	0	0.00	0.00 %
alter db	0	0.00	0.00 %	replace	0	0.00	0.00 %
alter table	0	0.00	0.00 %	replace select	0	0.00	0.00 %
analyze	0	0.00	0.00 %	reset	0	0.00	0.00 %
backup table	0	0.00	0.00 %	restore table	0	0.00	0.00 %
begin	0	0.00	0.00 %	revoke	0	0.00	0.00 %
change db	37	22.87	1.89 %	revoke all	0	0.00	0.00 %
change master	0	0.00	0.00 %	rollback	0	0.00	0.00 %
check	0	0.00	0.00 %	savepoint	0	0.00	0.00 %
checksum	0	0.00	0.00 %	select	325	200.89	16.63 %
commit	0	0.00	0.00 %	set option	76	46.98	3.89 %
create db	0	0.00	0.00 %	show binlog events	0	0.00	0.00 %
create function	0	0.00	0.00 %	show binlogs	4	2.47	0.20 %
create index	0	0.00	0.00 %	show charsets	19	11.74	0.97 %

Figura 7.2 phpMyAdmin, Información del tiempo de marcha al terminar la asignación



create index	0	0.00	0.00 %	show charsets	19	11.74	0.97 %
create table	0	0.00	0.00 %	show collations	19	11.74	0.97 %
dealloc sql	0	0.00	0.00 %	show column types	0	0.00	0.00 %
delete	0	0.00	0.00 %	show create db	11	6.80	0.56 %
delete multi	0	0.00	0.00 %	show create table	0	0.00	0.00 %
do	0	0.00	0.00 %	show databases	4	2.47	0.20 %
drop db	0	0.00	0.00 %	show errors	0	0.00	0.00 %
drop function	0	0.00	0.00 %	show fields	0	0.00	0.00 %
drop index	0	0.00	0.00 %	show grants	2	1.24	0.10 %
drop table	0	0.00	0.00 %	show innodb status	0	0.00	0.00 %
drop user	0	0.00	0.00 %	show keys	0	0.00	0.00 %
execute sql	0	0.00	0.00 %	show logs	0	0.00	0.00 %
flush	0	0.00	0.00 %	show master status	0	0.00	0.00 %
grant	0	0.00	0.00 %	show new master	0	0.00	0.00 %
ha close	0	0.00	0.00 %	show open tables	0	0.00	0.00 %
ha open	0	0.00	0.00 %	show privileges	0	0.00	0.00 %
ha read	0	0.00	0.00 %	show processlist	0	0.00	0.00 %
help	0	0.00	0.00 %	show slave hosts	0	0.00	0.00 %
insert	0	0.00	0.00 %	show slave status	0	0.00	0.00 %
insert select	462	285.58	23.64 %	show status	2	1.24	0.10 %
kill	0	0.00	0.00 %	show storage engines	0	0.00	0.00 %
load	0	0.00	0.00 %	show tables	32	19.78	1.64 %
load master data	0	0.00	0.00 %	show variables	38	23.49	1.94 %
load master table	0	0.00	0.00 %	show warnings	0	0.00	0.00 %
lock tables	0	0.00	0.00 %	slave start	0	0.00	0.00 %
optimize	0	0.00	0.00 %	slave stop	0	0.00	0.00 %
preload keys	0	0.00	0.00 %	truncate	1	0.62	0.05 %
prepare sql	0	0.00	0.00 %	unlock tables	0	0.00	0.00 %
purge	0	0.00	0.00 %	update	465	287.43	23.80 %
purge before date	0	0.00	0.00 %	update multi	231	142.79	11.82 %
rename table	0	0.00	0.00 %				

Figura 7.3 phpMyAdmin, Información del tiempo de marcha al terminar la asignación (continuación)



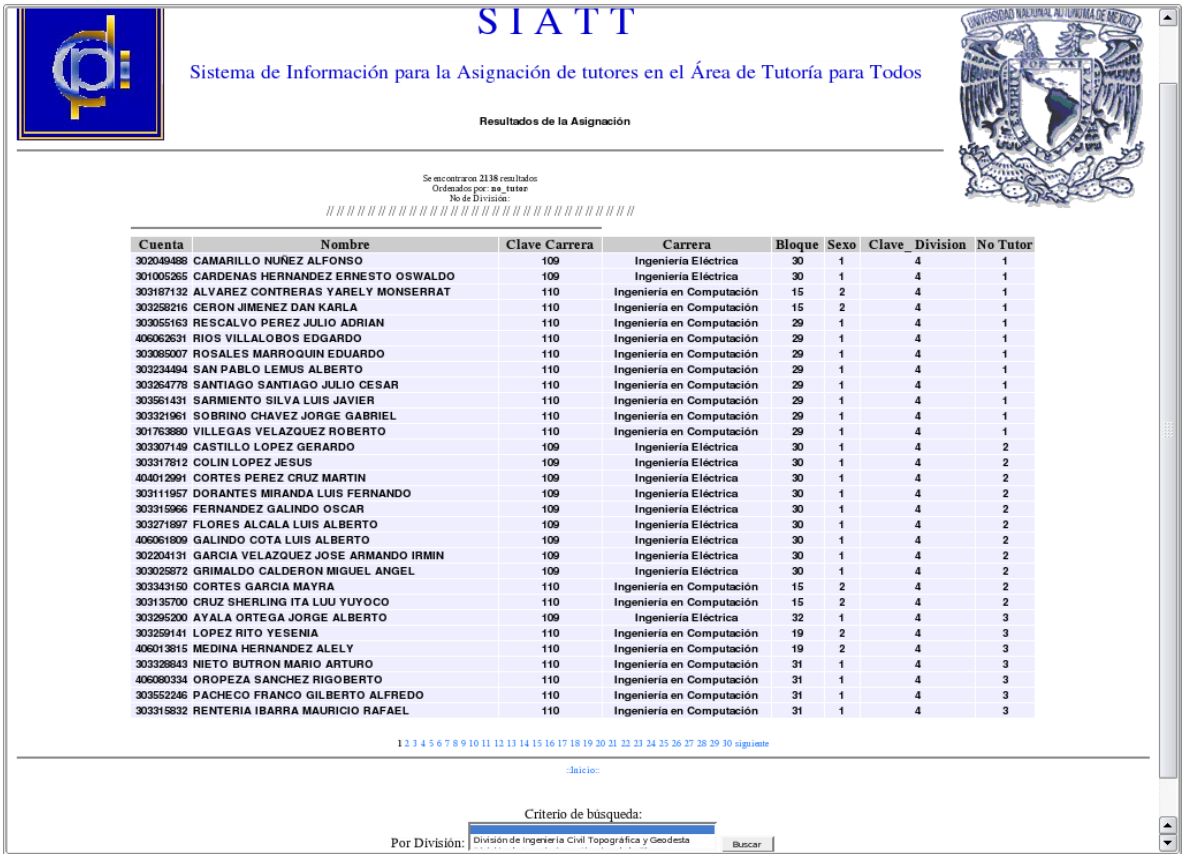
load master table	0	0.00	0.00 %	show warnings	0	0.00	0.00 %
lock tables	0	0.00	0.00 %	slave start	0	0.00	0.00 %
optimize	0	0.00	0.00 %	slave stop	0	0.00	0.00 %
preload keys	0	0.00	0.00 %	truncate	1	0.45	0.05 %
prepare sql	0	0.00	0.00 %	unlock tables	0	0.00	0.00 %
purge	0	0.00	0.00 %	update	465	209.51	23.02 %
purge before date	0	0.00	0.00 %	update multi	231	104.08	11.44 %
rename table	0	0.00	0.00 %				

◆ **Más variables del estado actual**

Variable	Valor	Variable	Valor	Variable	Valor
Binlog cache disk use	0	Key blocks not flushed	0	Rpl status	NULL
Binlog cache use	0	Key blocks unused	14497	Select full join	233
Created tmp disk tables	232	Key blocks used	0	Select full range join	0
Created tmp files	63	Key read requests	0	Select range	0
Created tmp tables	232	Key reads	0	Select range check	0
Delayed errors	0	Key write requests	0	Select scan	986
Delayed insert threads	0	Key writes	0	Slave open temp tables	0
Delayed writes	0	Max used connections	3	Slave running	OFF
Flush commands	1	Not flushed delayed rows	0	Slow launch threads	0
Handler commit	0	Open files	2	Slow queries	181
Handler delete	0	Open streams	0	Sort merge passes	0
Handler discover	0	Open tables	8	Sort range	0
Handler read first	509425	Opened tables	19	Sort rows	2366
Handler read key	1025714	Qcache free blocks	0	Sort scan	686
Handler read next	0	Qcache free memory	0	Table locks immediate	2694
Handler read prev	0	Qcache hits	0	Table locks waited	0
Handler read rnd	2366	Qcache inserts	0	Threads cached	0
Handler read rnd next	599913836	Qcache lowmem prunes	0	Threads connected	1
Handler rollback	14	Qcache not cached	0	Threads created	28
Handler update	0	Qcache queries in cache	0	Threads running	1
Handler write	7577	Qcache total blocks	0		

Figura 7.4 phpMyAdmin, Información del tiempo de marcha al terminar la asignación (continuación)

Ahora se ejecuta nuestro programa de asignación de alumnos (asignación.php) en el sistema operativo Linux SUSE 9.3.



S I A T T

Sistema de Información para la Asignación de tutores en el Área de Tutoría para Todos

Resultados de la Asignación

Se encontraron 2138 resultados
Ordenados por: no_tutor
No de División:

Cuenta	Nombre	Clave Carrera	Carrera	Bloque	Sexo	Clave_Division	No Tutor
302049488	CAMARILLO NUÑEZ ALFONSO	109	Ingeniería Eléctrica	30	1	4	1
301005265	CARDENAS HERNANDEZ ERNESTO OSWALDO	109	Ingeniería Eléctrica	30	1	4	1
303187132	ALVAREZ CONTRERAS YARELY MONSERRAT	110	Ingeniería en Computación	15	2	4	1
303258216	CERON JIMENEZ DAN KARLA	110	Ingeniería en Computación	15	2	4	1
303055163	RESCALVO PEREZ JULIO ADRIAN	110	Ingeniería en Computación	29	1	4	1
406062631	RIOS VILLALOBOS EDGARDO	110	Ingeniería en Computación	29	1	4	1
303085007	ROSALES MARROQUIN EDUARDO	110	Ingeniería en Computación	29	1	4	1
303234494	SAN PABLO LEMUS ALBERTO	110	Ingeniería en Computación	29	1	4	1
303264778	SANTIAGO SANTIAGO JULIO CESAR	110	Ingeniería en Computación	29	1	4	1
303561431	SARMIENTO SILVA LUIS JAVIER	110	Ingeniería en Computación	29	1	4	1
303321961	SOBRINO CHAVEZ JORGE GABRIEL	110	Ingeniería en Computación	29	1	4	1
301763880	VILLEGAS VELAZQUEZ ROBERTO	110	Ingeniería en Computación	29	1	4	1
303307149	CASTILLO LOPEZ GERARDO	109	Ingeniería Eléctrica	30	1	4	2
303317812	COLIN LOPEZ JESUS	109	Ingeniería Eléctrica	30	1	4	2
404012991	CORTES PEREZ CRUZ MARTIN	109	Ingeniería Eléctrica	30	1	4	2
303111957	DORANTES MIRANDA LUIS FERNANDO	109	Ingeniería Eléctrica	30	1	4	2
303315966	FERNANDEZ GALINDO OSCAR	109	Ingeniería Eléctrica	30	1	4	2
303271897	FLORES ALCALA LUIS ALBERTO	109	Ingeniería Eléctrica	30	1	4	2
406061809	GALINDO COTA LUIS ALBERTO	109	Ingeniería Eléctrica	30	1	4	2
302204131	GARCIA VELAZQUEZ JOSE ARMANDO IRMIN	109	Ingeniería Eléctrica	30	1	4	2
303025872	GRIMALDO CALDERON MIGUEL ANGEL	109	Ingeniería Eléctrica	30	1	4	2
303343150	CORTES GARCIA MAYRA	110	Ingeniería en Computación	15	2	4	2
303135700	CRUZ SHERLING ITA LUU YUYOCO	110	Ingeniería en Computación	15	2	4	2
303295200	AYALA ORTEGA JORGE ALBERTO	109	Ingeniería Eléctrica	32	1	4	3
303259141	LOPEZ RITO YESENIA	110	Ingeniería en Computación	19	2	4	3
406013815	MEDINA HERNANDEZ ALELY	110	Ingeniería en Computación	19	2	4	3
303328843	NIETO BUTRON MARIO ARTURO	110	Ingeniería en Computación	31	1	4	3
406080334	OROPEZA SANCHEZ RIGOBERTO	110	Ingeniería en Computación	31	1	4	3
303552246	PACHECO FRANCO GILBERTO ALFREDO	110	Ingeniería en Computación	31	1	4	3
303315832	RENTERIA IBARRA MAURICIO RAFAEL	110	Ingeniería en Computación	31	1	4	3

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 siguiente

::Inicio::

Criterio de búsqueda:

Por División: División de Ingeniería Civil Topográfica y Geodesta Buscar

Figura 7.5 Asignación realizada por el SIATT corriendo sobre Linux SUSE 9.3.

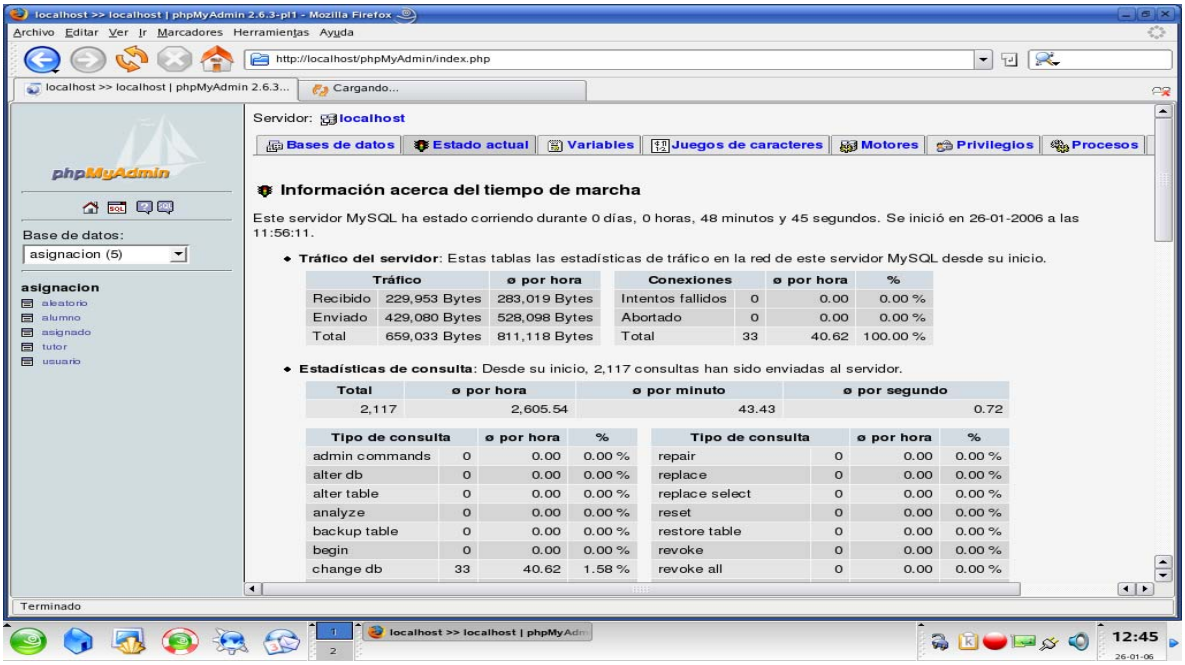


Figura 7.6 phpMyAdmin, Información del tiempo de marcha al terminar la asignación

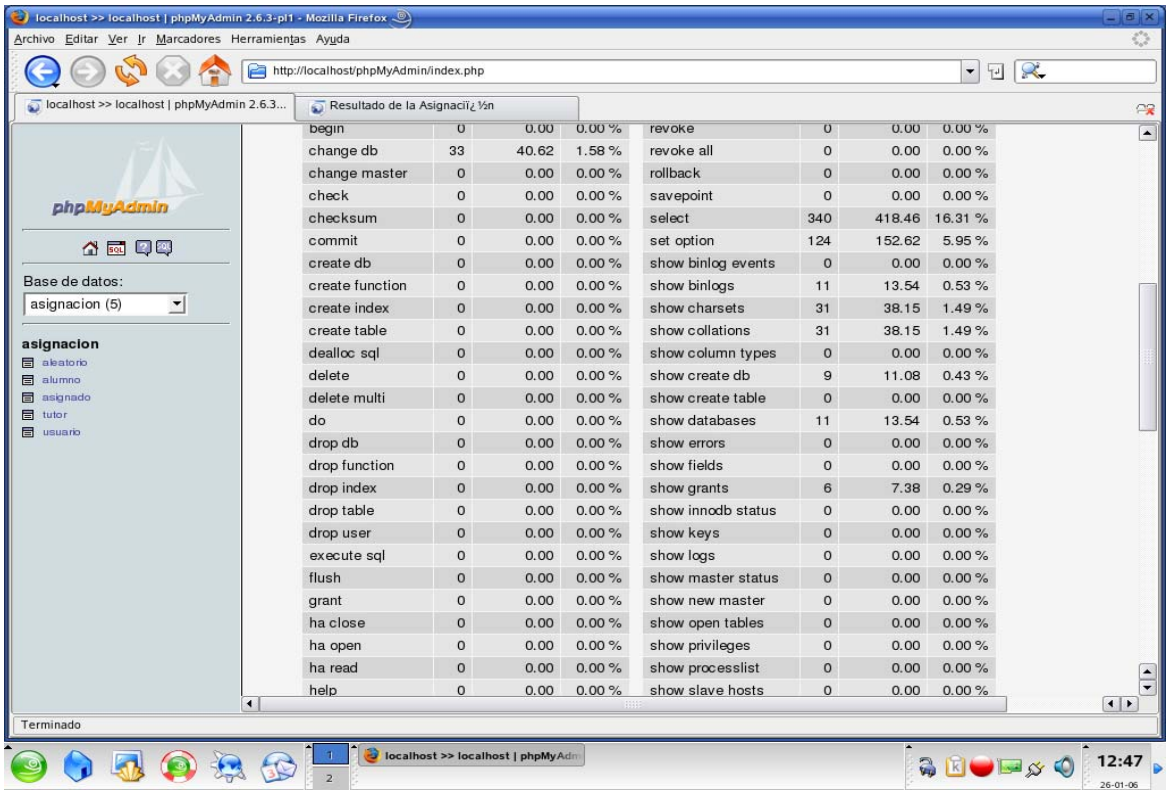


Figura 7.7 phpMyAdmin, Información del tiempo de marcha al terminar la asignación (continuación)

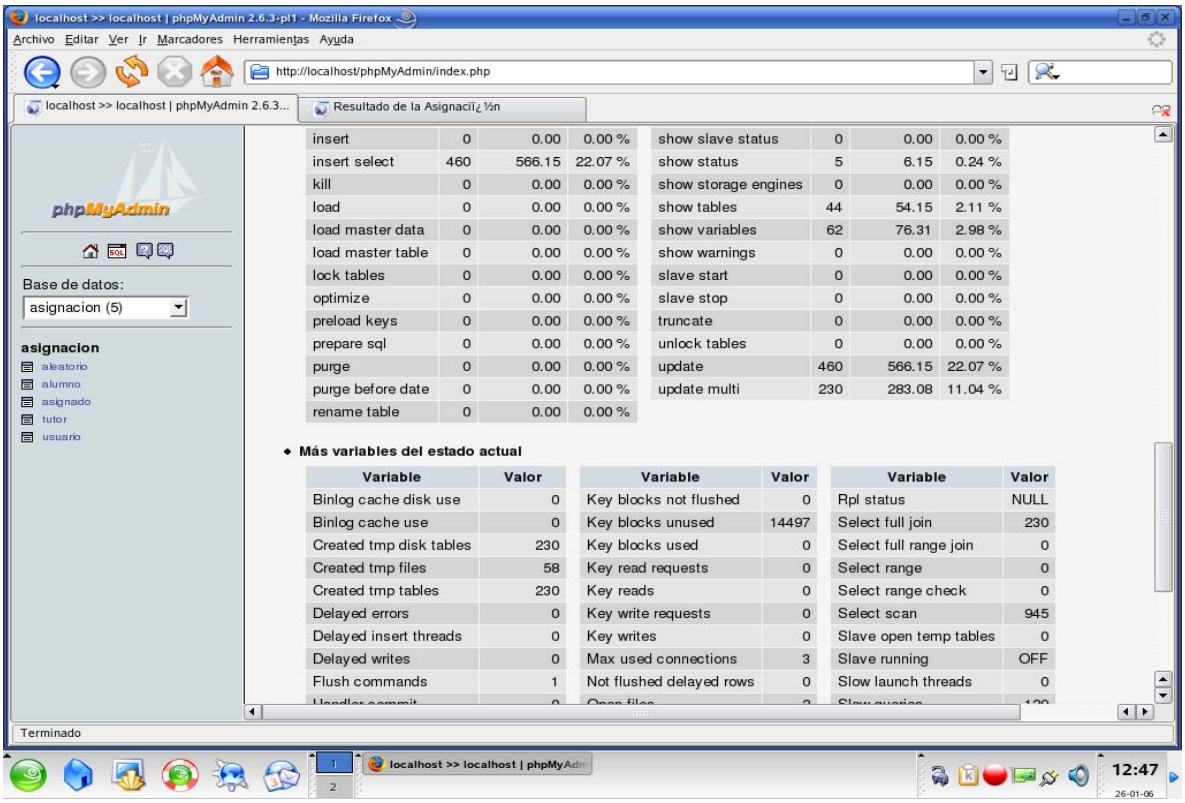
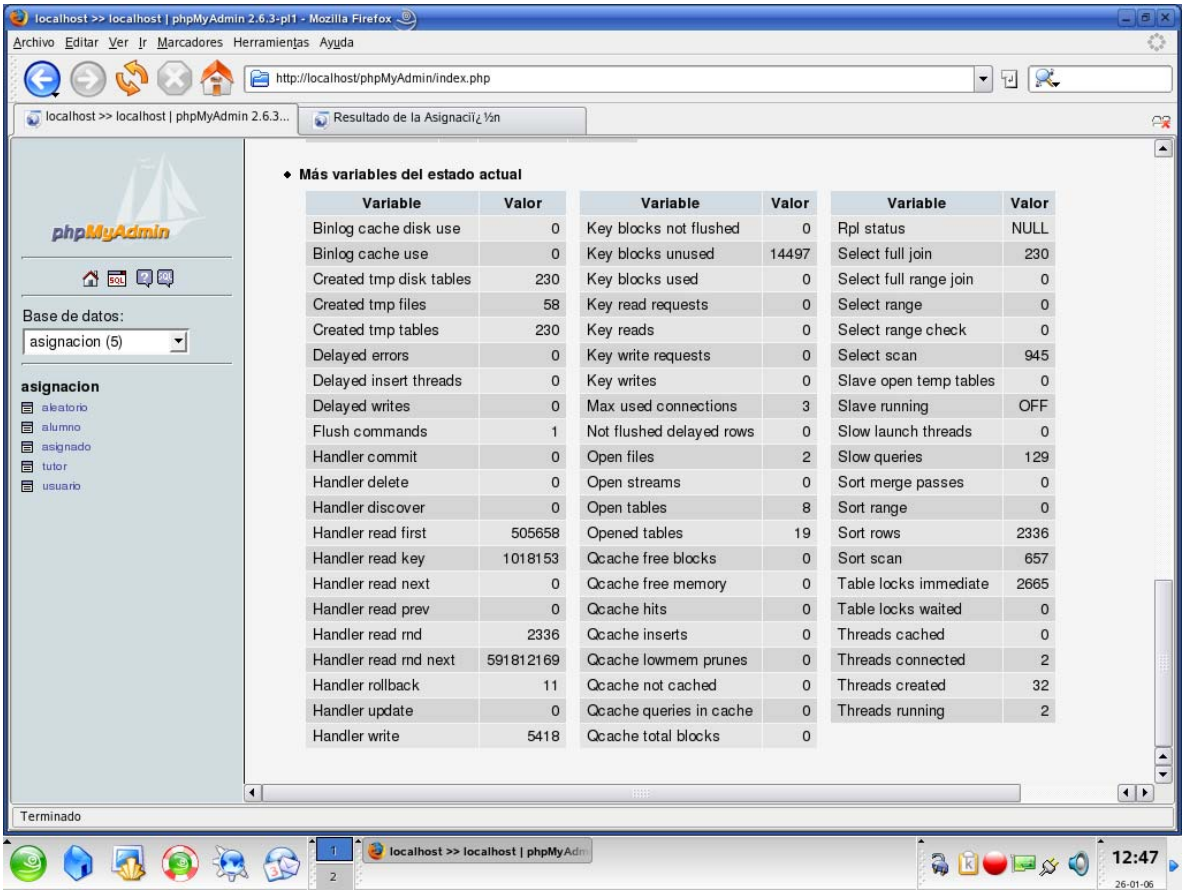


Figura 7.8 phpMyAdmin, Información del tiempo de marcha al terminar la asignación (continuación)



Variable	Valor	Variable	Valor	Variable	Valor
Binlog cache disk use	0	Key blocks not flushed	0	Rpl status	NULL
Binlog cache use	0	Key blocks unused	14497	Select full join	230
Created tmp disk tables	230	Key blocks used	0	Select full range join	0
Created tmp files	58	Key read requests	0	Select range	0
Created tmp tables	230	Key reads	0	Select range check	0
Delayed errors	0	Key write requests	0	Select scan	945
Delayed insert threads	0	Key writes	0	Slave open temp tables	0
Delayed writes	0	Max used connections	3	Slave running	OFF
Flush commands	1	Not flushed delayed rows	0	Slow launch threads	0
Handler commit	0	Open files	2	Slow queries	129
Handler delete	0	Open streams	0	Sort merge passes	0
Handler discover	0	Open tables	8	Sort range	0
Handler read first	505658	Opened tables	19	Sort rows	2336
Handler read key	1018153	Qcache free blocks	0	Sort scan	657
Handler read next	0	Qcache free memory	0	Table locks immediate	2665
Handler read prev	0	Qcache hits	0	Table locks waited	0
Handler read rnd	2336	Qcache inserts	0	Threads cached	0
Handler read rnd next	591812169	Qcache lowmem prunes	0	Threads connected	2
Handler rollback	11	Qcache not cached	0	Threads created	32
Handler update	0	Qcache queries in cache	0	Threads running	2
Handler write	5418	Qcache total blocks	0		

Figura 7.9 phpMyAdmin, Información del tiempo de marcha al terminar la asignación (continuación)

Lo primero que notamos, es que en **Windows XP** la asignación se realiza en **una hora con treinta y siete minutos**, mientras que en **Linux** se realiza en **cuarenta y ocho minutos**, esto es debido a que Linux realiza más del doble de conexiones por minuto a la base de datos para llevar a cabo la asignación.

En Windows el sistema realizó 1978 consultas al servidor, en Linux fueron 2117.

Linux hace 20.38 consultas al servidor por minuto, en cambio Windows XP hace 43.43.

Linux realizó 139 consultas más que WindowsXP, pero la diferencia es que estas consultas las realiza casi en la mitad de tiempo que Windows XP.

Windows realizó 462 actualizaciones de registros igual que Linux, pero este último crea 5 archivos temporales menos, lo que disminuye la cantidad de memoria que ocupa para terminar el proceso.

Entonces podemos concluir que el **SIATT** funciona adecuadamente en las dos plataformas utilizadas para hacer las pruebas, aunque se demostró también que se ejecuta más rápido en Linux.

El sistema se ejecuta mucho mejor en Linux, debido a que las herramientas usadas fueron en un principio diseñadas para Linux, además de que este sistema operativo, a comprobado ser el sistema operativo para servidores Web por excelencia.

Asimismo se puede concluir que corre aceptablemente en Windows XP que es el sistema operativo donde va a ser implantado.

En cuanto a la interfaz gráfica, se comportó de la misma manera en las dos plataformas, sólo con un cambio de dimensiones muy pequeño en ventanas y botones.

También se comprobó que la asignación se llevó a cabo de manera exitosa, esto se hizo explorando los datos arrojados por la asignación, que deberían de ser 2138 alumnos asignados, puesto que ese es el número de alumnos existentes.

Adentrándonos más en los resultados de la asignación, observamos que se cumplen todos los criterios de asignación pedidos por el cliente, como por ejemplo se asignaron primero los alumnos de Ingeniería Civil y al último los que quedaban de Computación, el número de hombres y mujeres es el correcto para cada tutor, la mayoría de los alumnos asignados al tutor son del mismo bloque etc.

7.2 Conclusiones

Con la realización del "**S**istema de **I**nformación para la **A**signación de tutores en el Área de "**T**utoría para **T**odos"(**SIATT**), llegamos a las siguientes conclusiones:

La increíble evolucion de los componentes y servidores de red nos permitio comprobar que, el utilizar la tecnología informática en el país, agiliza los procesos para el manejo de gran cantidad de información, así como, los recursos.

Verificamos la calidad y eficiencia del software libre, comprobando que el dinero no es problema cuando se trata de ser creativos, comprobando que tiene diversas aplicaciones tanto para el desarrollo de sistemas como para el análisis de información.

Para construir este sistema fue necesario utilizar una metodología adecuada para diagnosticar las necesidades del programa ,y seguir todo el análisis y el diseño planteado en la Ingeniería del Software, de esta manera logramos alcanzar nuestros objetivos de forma óptima.

En el caso del sistema SIATT, se comparó el tiempo de proceso para una asignación y el proceso de asignación de forma manual que se realiza en máximo tres semanas tomando en cuenta el cansancio humano, jornada de trabajo, actividades externas y ajenas a la persona, lo que demuestra que aun cuando se realice el trabajo en el mejor y mayor esfuerzo, se puede presentar errores manuales que se van empalmando a los resultados finales.

Al momento de hacer la asignación prueba mediante el sistema SIATT, se eliminan errores humanos y el tiempo de ejecución es mucho menor, lo que comprueba, que al utilizar el software necesario y apropiado se agilizan procesos y se minimizan horas-trabajo, esto se refiere, que en vez de realizar una sola actividad, la persona puede realizar otras actividades y presentar una mayor productividad laboral.

En la realización del sistema SIATT, se tomaron todas las medidas y necesidades para que el sistema sea productivo y también escalable de acuerdo con el incremento de la matrícula de la Facultad de Ingeniería, la matrícula de tutores y el avance tecnológico tanto de software como de hardware.

BIBLIOGRAFÍA

Bibliografía

García y Colomé Pablo, Cano Salazar María Elena
La tutoría en la facultad de ingeniería
Departamento de Publicaciones de la Facultad de Ingeniería UNAM
México
2002

Abraham Silberschatz, Henry F. Korth, S. Sudarshan
Fundamentos de bases de datos
McGraw-Hill/Interamericana de España
España
2004

Alice Y., Tsai H.
Sistema de base de datos: Administración y uso
Prentice Hall Hisponoamericana S.A
Canada
1990

Connolly T., Begg C., A. Strachan
Database Systems. A Practical Approach to Design, Implementation and Management
Addison-Wesley
Estados Unidos
1998.

Folk M.J., Zoellick B
File Structures
Addison-Wesley
Estados Unidos
1992

Guilera Aguera, Llorenc
Introducción a la informática
Edunsa
Barcelona
1988

Marty, Tim. Hartley, Tim
DB2/SQL A professional Progammer's Guide
McGraw Hill Book Company
Estados Unidos
1989

Mota Herranz, Laura
Bases de datos relacionales: teoría y diseño
Universidad Politécnica de Valencia
Valencia
1994

Pressman
Software Engineering. A practitioner approach
McGraw Hill
Estados Unidos
2002

Sommerville
Software Engineering
Addison Wesley
Estados Unidos
2002

Worsley John, Drake Joshua
PostgreSQL Práctico
O'Reilly
2001

Páginas de referencia

http://www.colnod.apc.org/registro/apache.shtml
http://www.ibm.com/mx/products/software/db2/ts/db2.phtml

Páginas de organizaciones

http://www.apache.org

http://www.ibm.com/

http://www.microsoft.com.mx

http://www.mysql.com

http://www.php.net

http://www.oracle.com