Universidad Nacional Autónoma de México
Escuela Nacional de Estudios Profesionales
"Acatlán"

Análisis, Diseño e Implementación de un Sistema de Administración para el Centro Electrónico de Cálculo del Instituto Tecnológico de Estudios Superiores de Monterrey

T E S I S

Que para obtener el Título de

Licenciado en Matemáticas Aplicadas y Computación

PRESENTAN
Botello Rivera Maricela
Caballero Botello Ofelia
López Velázquez Raquel

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Agradecimientos

Papá, Mamá
Gracias por su amor, impulso y apoyo
sé que cuento con mis queridos padres
en todo momento de mi vida.

Gracias Marcela, Gracias Marisol
Mis incomparables hermanas
Mis amigas incondicionales

A mis maestros con admiración y respeto

A Ofelia y Raquel
Por su colaboración increible
por su entusiasmo y paciencia
y sobre todo por su amistad y compañerismo

Maricela

El agradecimiento es una una forma de expresar un sentimiento, es algo que se brinda muy especialmente a las personas que comparten los momentos buenos y malos, las experiencias alegres, tristes, pasajeras, en fin... cualquier momento.

Y es precisamente ese agredecimiento el que sinceramente doy:

🕮 **A Dios, por ser tan buen amigo y compañero en todo momento.**

🕮 **A mi Papá y mi Mamá, porque son los mejores e incodicionales amigos, y porque éste trabajo es también labor suya.**

🕮 **A cada uno de mis hermanos, porque de ustedes he podido aprender lo mejor y por lo tanto obtener mejores satisfacciones.**

🕮 **A todos mis amigos y amigas, porque son parte importante en el aprendizaje diario de mi vida.**

🕮 **A Maricela y a Raquel, ya que juntas hemos logrado éste trabajo, compartiendo muchos ratos terribles, pero siempre esperando salir adelante.**

🕮 **Y especialmente a ti Alejandro, por ser parte importante en mi vida, gracias por el apoyo y las palabras de aliento que junto con tu familia me has brindado.**

Con mucho cariño
Ofelia

A mi madre

Por tantas ocasiones
en que debí decir...¡Gracias!

Conocemos mucha gente
en nuestro paso por la vida,
pero solo unos pocos
nos dejan una impresión duradera
en el espíritu y el corazón...

Gracias David...y a todos mis amigos

A Leonel
Simplemente...por todo

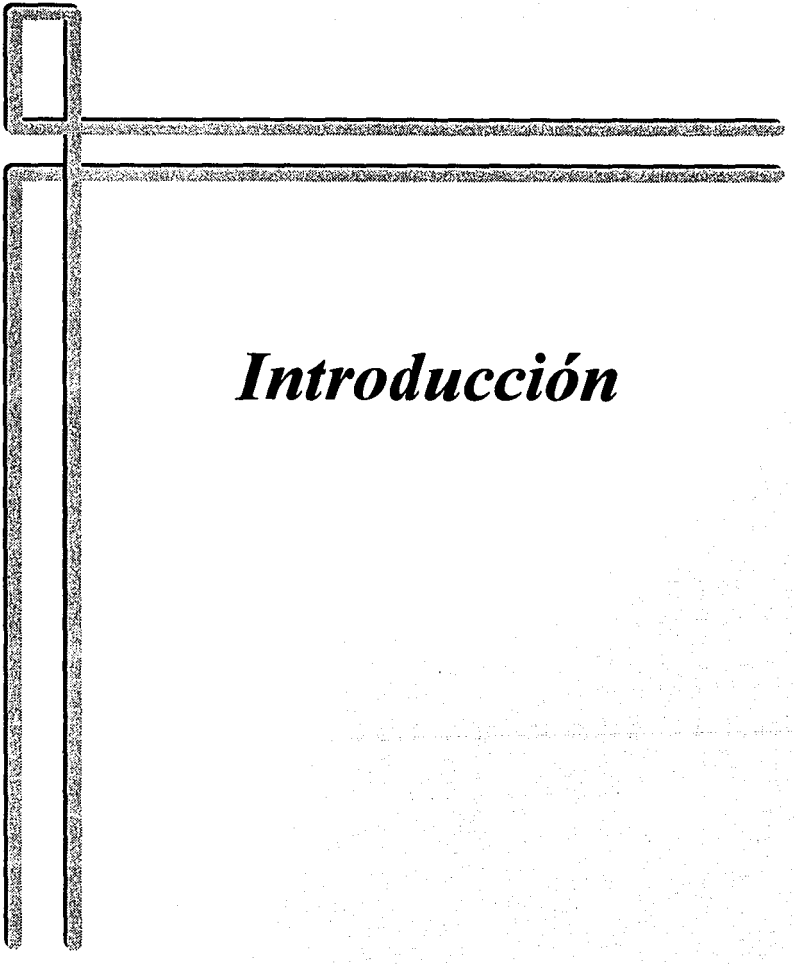
A la memoria de mi padre...

RAQUEL

INDICE

III. Planteamiento del Problema y Propuesta de Solución

IV. Desarrollo del Sistema



Introducción

INTRODUCCION

En la actualidad el uso de las computadoras se está convirtiendo en una actividad muy común y cotidiana, tanto en el trabajo como en el hogar, pero principalmente en las instituciones educativas de nivel medio superior y profesional. Esto conlleva a que las instituciones educativas se preocupen por mejorar sus instalaciones de cómputo, así como el servicio que prestan del mismo.

El Instituto Tecnológico de Estudios Superiores de Monterrey campus Ciudad de México, no es la excepción y en su afán por alcanzar y mantener una excelencia académica ha dedicado esfuerzos y recursos para proporcionar a su población escolar los equipos más actualizados y modernos para su desarrollo académico.

El propósito de la presente tesis es proporcionar al Instituto Tecnológico de Estudios Superiores de Monterrey campus Ciudad de México, un Sistema de Administración para el Centro de Cálculo. Dicho sistema, permitirá administrar y controlar los recursos del mismo, en lo que se refiere a inventario de equipo y de usuarios, control en los accesos al Centro de Cálculo, las cancelaciones y reasignaciones de equipo; así mismo, se podrán obtener estadísticas en cuanto a la oferta y demanda de equipo en cada hora para cada tipo de máquina durante cada ciclo escolar como apoyo a la toma de decisiones gerenciales. Además de prestar un servicio ágil y eficiente en el

apartado de equipo por parte de los usuarios y cumpliendo con el reglamento interno del Centro de Cálculo.

El presente trabajo ha sido desarrollado en los siguientes capítulos:

Capítulo I, el cual habla sobre la Coordinación de Atención a Usuarios, sus funciones generales y su papel dentro de la organización del Instituto Tecnológico de Estudios Superiores de Monterrey.

El Capítulo II, contiene información teórica sobre aspectos de ingeniería de software, técnicas de análisis y diseño de sistemas; aspectos generales sobre las diferentes herramientas utilizadas en el diseño del sistema como son: el lenguaje de programación Pascal Versión 7.0 de Borland, el manejador de archivos Btrieve y Visual Basic 3.0 de Microsoft. Así como, conceptos generales de redes.

En el Capítulo III, se aborda la situación actual del sistema, el planteamiento del problema del Centro de Cálculo y la estrategia a seguir en la solución del mismo.

Finalmente el Capítulo IV, presenta el diseño del sistema, las pantallas que lo conforman y su operación en red; así como los manuales técnico y de usuario.

Capítulo I

Coordinación de Atención a Usuarios

I.1 FUNCIONES GENERALES DE LA COORDINACION DE ATENCION A USUARIOS.

La coordinación de atención a usuarios es la parte medular de la operación informática del Instituto Tecnológico y de Estudios Superiores de Monterrey en su Campus Ciudad de México (ITESM-CCM). Básicamente, en esta coordinación se realizan todos los servicios de soporte informático que el Campus requiere para su funcionamiento.

Sin embargo, para ofrecer una mejor comprensión del operar cotidiano de la coordinación, es necesaria una visión general del ITESM-CCM que permita conocer cuál es la misión de la misma en el contexto que plantea del devenir histórico de dicha institución.

El ITESM tiene como misión fundamental, la *de crear profesionales y posgraduados de excelencia en el campo de su especialidad*; esta misión, es la que ha generado lo que hoy conocemos como ITESM.

Pero ¿qué es el ITESM?, para responder a esta pregunta, hay que recurrir a la historia. Un grupo de empresarios de la ciudad de Monterrey, Nuevo León, encabezados por Don Eugenio Garza Sada (1892-1973), toman la iniciativa de crear

una escuela que permitiera la formación de los profesionales que la naciente ciudad industrial requería.

Los sueños de Don Eugenio Garza crecerían hasta crear en el año de 1943, el ITESM. En un principio, se comenzaron las clases con una matrícula de trescientos cincuenta alumnos en el semestre de septiembre de 1943 a enero de 1944 (la matrícula del siguiente ciclo aumentó a cuatrocientos cincuenta y dos).

Las clases comenzaron el 6 de septiembre de 1943 en Monterrey, N.L. que por entonces, era una ciudad provinciana con una industria abundante y progresista sometida al colapso de un mundo enfrascado en una guerra cuyo fin no podía vaticinarse.

El ITESM empezó a funcionar en una casona patriarcal de sillar y techos de viga, con balconería en el segundo piso, varios patios que se localizaba a unas cuadras de la Plaza Zaragoza; se rentaron unas dieciocho habitaciones del Hotel Plaza para el inicio de la modalidad de internado.

En el año de 1947, el ITESM inauguró su primer campus con más de mil alumnos inscritos. Con el paso del tiempo, se inauguraron campus en Saltillo, Querétaro y eventualmente en 22 ciudades más.

Hasta la fecha, el ITESM cuenta con veintiséis campus ubicados en veinticinco ciudades del país.

Desde el año de 1963 en que se creó el primer *Centro Electrónico de Cálculo* (CEC) en el campus Monterrey, el impacto de la informática en el ITESM ha sido tan importante, que hoy en día, no se puede imaginar la operación de los diversos campus sin la presencia de las computadoras.

En el año de 1973, el ITESM inaugura en el centro de la Ciudad de México, *la Escuela de Graduados en Administración* (EGA), con el fin de preparar ejecutivos de alto nivel y bajo el auspicio del Lic. Agustín Legorreta Chauret y otros empresarios.

Su primer domicilio se ubicó en la calle de Dr. Lucio 102 que años después se mudó al local ubicado en la Av. Fray Servando Teresa de Mier 99.

El proyecto del Campus Ciudad de México, surgió a partir de la necesidad de ampliar su estructura debido a la alta demanda de personas altamente calificadas a nivel medio y superior.

Por ello se decidió crear un campus al sur de la ciudad en la delegación de Tlalpan. El nuevo campus, comenzó a operar el 6 de agosto del año de 1990 con una matrícula de 1327 alumnos en una superficie de diez hectáreas ubicada en la Calle del Puente 222.

Bajo un ambiente inspirado en la arquitectura mexicana del siglo XVII, se pueden encontrar en el ITESM-CCM, salones, aulas magnas, laboratorios, salones de cómputo, oficinas, cafetería, biblioteca, estacionamiento y dos auditorios, además de áreas verdes, una planta de tratamiento de aguas residuales y canchas deportivas.

En el esfuerzo por crear graduados de excelencia, se ha puesto especial énfasis al aspecto informático y computacional en el ITESM, por lo que en 1963 se crea el primer CEC y se adquiere la primera computadora que fue una IBM 1620.

Debido a que en 1965 se abrió la carrera de Técnico en Procesamiento de Datos, en febrero de 1967 se creó la licenciatura correspondiente que se denominó Ingeniero en Sistemas Computacionales.

Hacia el año de 1970, se instaló la computadora CDC 3300 (Control Data) con lo que se implementaron nuevos servicios y se propició que el 80% de las carreras profesionales, tuvieran al menos un curso de computación. Así mismo, se crea el Departamento de Sistemas Computacionales para coordinar estos cursos.

En el año de 1971 se abre la carrera de Licenciado en Sistemas de Computación Administrativa que se orientaría principalmente a la organización de los procesos administrativos de la organización.

Se instaló una computadora IBM 370-158 en el año de 1975 con lo que se logra que los alumnos tengan acceso mediante terminales. Se ofreció entonces el servicio de proceso de trabajos por medio de tarjetas perforadas y como innovación, se implementó el servicio *expreso* para el proceso de un programa en treinta segundos como tiempo de respuesta.

El primer sistema de evaluación de exámenes por computadora, se instaló hacia el año de 1976 así como la creación de la carrera de Ingeniero Químico y de Sistemas y la maestría en Sistemas de Información. En 1977 se crea la carrera de Ingeniero en Sistemas Electrónicos orientada al estudio del **hardware** (componentes físicos) de las computadoras.

Las primeras computadoras personales en llegar al ITESM, fueron las Apple II, en el año de 1980 con lo que se pone la computación personal al alcance de los alumnos.

En 1983, se implementa un sistema de inscripciones en el campus Eugenio Garza Sada con más de 300 computadoras conectadas en red para el acceso y asignación de horarios instantáneamente a alumnos.

Hacia el año de 1985, se instalan cuatro computadoras IBM 4381 ubicadas dos de ellas en el campus Monterrey, una más en el campus Querétaro y otra en el campus Estado de México.

Se inicia con sistemas tales como el CADAM (Computer Aided Design And Manufacturing -diseño y manufactura asistidos por computadora-) en 1986, año en que también se incorporan las primeras IBM PC así como equipos Macintosh. También en este año, el ITESM se conecta a la red internacional BITNET.

Las primeras redes locales para uso de alumnos se elaboran en 1987 y comienzan a dar servicio en 1988.

En 1989 se instaló un equipo NeXT para personal docente con estaciones DECstation y VAXstation y adicionalmente un equipo VAX 6310 para estudiantes.

Las primeras estaciones HP Apollo, se instalan en el año 1990 y un año después, se monta un equipo IBM RS/6000 para alumnos.

Como se puede observar, el crecimiento informático del ITESM ha sido la piedra angular en el desarrollo tanto de las disciplinas ajenas a la computación, como para la creación de nuevas carreras enfocadas específicamente en la computación.

Con este antecedente, se puede mencionar que es necesario un organismo que regule la amplia gama de servicios otorgados por el instituto así como las relaciones que se originan entre los prestadores de servicio y los usuarios.

Bajo este marco, la Coordinación de Atención a Usuarios, es la encargada de optimizar las relaciones antes citadas así como la abocada al cumplimiento de la misión de encontrarse en vanguardia tecnológica en la oferta de servicios Computacionales.

Hoy en día, los servicios que se coordinan en Atención a Usuarios, son los siguientes:

- Software (programas de cómputo),
- Salas de Cómputo,
- Centro Electrónico de Cálculo,
- Zona de Impresión.

Para precisar más las funciones de la Coordinación de Atención a Usuarios (CAU), se detallan a continuación cada uno de los puntos de servicio.

- ***Software***

En el campus Ciudad de México, se cuenta con aproximadamente 7500 alumnos y 800 empleados administrativos y docentes. Todos ellos, requieren el acceso a equipo de cómputo y a la explotación del software correspondiente a sus especialidades. En virtud de la heterogeneidad de los perfiles de los usuarios, es necesario contar con una serie de programas de aplicación para las distintas áreas.

Como es de suponerse, el tema de los "Derechos de Autor", es un punto álgido para la organización, ya que ante el advenimiento de una gran cantidad de equipo para uso

del profesorado, se presenta frecuentemente el problema de que se instala software en dichos equipos indiscriminadamente y sin atención a las debidas restricciones de licencias que legalmente representa.

Por lo anteriormente mencionado, es necesaria toda una área de la coordinación encargada de la administración de las licencias de uso de programas que ha adquirido el Instituto.

Los programas de cómputo que se ofrecen a la comunidad del campus, son los siguientes:

Microsoft Office 4.2 para PC y Mac,

Microsoft Word 2.0 para PC,

Microsoft Word 5.1 para Mac,

Microsoft Excel 4.0 para PC y Mac,

Microsoft Power Point 3.0 para PC y Mac,

Aldus Pagemaker 4.5 y 5.0 para PC y Mac,

McAfee Viruscan v214,

Claris Works para Pc y Mac,

MacProject para Mac,

Filemaker Pro para Mac,

Mathematica v2.2 para PC y Mac,

PcTEX para PC,

PCAccess para Mac,

Economics in Action para PC,

Interactive Economics Tutor para PC,

Aldus PhotoStyler para PC,

Adobe Photoshop para Mac,

Infini-D para Mac,

Human Factor para PC,

Mayer Briggs para PC,

Informix 4gl para AIX,

CADAM para AIX,

SAS para PC y AIX,

Microsoft Visual Basic 3.0 para PC,

Microsoft Visual C++ 2.0 para PC,

Microsoft Library 1.0 para PC,

Borland C++ 3.0 para PC,

Borland Turbo Pascal v7.0 para PC,

Ashton Tate dBase III+ v2.0 para PC,

Borland dBase IV v1.5 para PC,

Borland Paradox 3.0 para PC,

Nantucket Clipper 5.01 para PC,

AREMOS para PC,

QSB+ para PC,

Lindo para PC,

GraphEcon II para PC,

etc.

El procedimiento de control de los programas de software es el siguiente: Inicialmente, se tiene una serie de programas instalados para el servicio de alumnos en el CEC, donde se permite que se empleen libremente. De forma análoga, se dispone de una computadora para cada profesor de planta, la cual por supuesto, también requiere software instalado.

Así, se lleva un control minucioso de todo el software que se instala en las máquinas de los profesores y administrativos para realizar la equiparación contra el número de licencias adquiridas.

Adicionalmente, se entrega a cada profesor, una carta personalizada donde se asienta cuál es el software que el ITESM-CCM ampara en cuestión de licencia con lo que se deslinda responsabilidad en el caso de una auditoría de software. Este proceso se lleva a cabo semestralmente. A diario se presentan solicitudes de software y de asesorías que son oportunamente asistidas.

Es responsabilidad de esta coordinación, el correcto funcionamiento de todos los programas instalados en el campus.

- ***Salas de Cómputo***

Para la impartición de cursos que requieren del uso de computadoras, se dispone de distintos salones o salas de cómputo que se asignan a los distintos profesores de las materias que lo ameriten.

Las salas de cómputo, son 7 y se distribuyen de la siguiente manera:

⇒ SC01. Equipada con 50 equipos workstation IBM RS/6000 conectados en red y con salida a Internet (red mundial de computadoras).

⇒ SC02. Equipada con 50 equipos workstation HP Apollo.

⇒ SC03. Equipada con 35 equipos IBM PS Value Point 433 Dx/Dp

⇒ SC04. Equipada con 35 equipos Macintosh Power Mac Modelo 7100/60 en red local.

⇒ SC05. Equipada con 36 equipos IBM PS/2 Modelo 25 conectadas a la red del campus.

⇒ SC06. Equipada con 23 equipos IBM PS/2 Modelo 35 conectadas a la red del campus.

⇒ SC07. Equipada con 36 equipos IBM PS/2 Modelo 25 conectadas a la red del campus.

Las salas se asignan al inicio del semestre, para la impartición de materias de computación, sin embargo, dado que no se ocupan durante el horario completo de labores, se ofrecen para impartir cursos adicionales.

El uso de las salas de cómputo, se encuentra restringido por las horas de clases cotidianas (que tienen prioridad) y se realiza mediante apartado previo. El apartado se debe realizar vía memorándum especificando el equipo que se requiere, el software a emplear y la cantidad de personas que tomarán el curso.

Con la información antes citada, se procede a asignar la sala de cómputo que corresponda y se confirma vía telefónica el apartado para seguridad del usuario.

- ***Centro Electrónico de Cálculo***

El Centro Electrónico de Cálculo, es el área en la que las personas que requieran de servicios de informática, pueden recurrir para el uso de computadoras en préstamo y software que se encuentra instalado en las mismas.

El CEC se encuentra equipado con los siguientes equipos:

64 Workstation IBM RS/6000 conectadas a un servidor IBM RISC 6000 mod 990

64 IBM PS Value Point 433 Dx/Dp bajo una red Ethernet Novell 3.12

48 Macintosh PowerMac 6100/60 bajo una red Ethernet Novel 3.12

Los equipos que se encuentran en el CEC, se encuentran para el uso de los alumnos del campus, los profesores, los empleados administrativos, los exalumnos con identificación y visitantes para proyectos especiales.

Las políticas de uso del equipo, implican que cada persona en una máquina, es responsable por el uso que se dé al equipo, por lo que se requiere que no se emplee para actividades ajenas a las académicas, es decir, no será posible desplegar imágenes no adecuadas, instalar juegos de cualquier especie, etc.

El reglamento de CEC, indica que las sanciones para la utilización de equipo para fines no académicos, varían desde una suspención del servicio del CEC durante una semana hasta la suspensión de todos los servicios de Informática (Biblioteca, Salas de Cómputo, CEC, Impresiones, etc.) por un semestre o hasta la expulsión del campus en casos que así lo ameriten por su gravedad y/o reincidencia.

Será posible apartar un equipo para garantizar el acceso a las instalaciones o emplearlo sin apartado en el caso en que haya disponibilidad de equipo.

En el caso de requerir de asesoría en cuento a problemas que se presenten en la operación del equipo, se cuenta con asesores que pueden resolver las dudas que surjan y que pueden auxiliar a los usuarios en el manejo de la paquetería.

El software que se encuentra instalado en el CEC, es principalmente:

Microsoft Office 4.2 para PC y Mac,

McAfee Viruscan v214,

Claris Works para Pc y Mac,

Mathematica v2.2 para PC y Mac,

PCAccess para Mac,

Economics in Action para PC,

Interactive Economics Tutor para PC,

CADAM para AIX,

Microsoft Visual Basic 3.0 para PC,

Borland C++ 3.0 para PC,

Borland Turbo Pascal v7.0 para PC,

Ashton Tate dBase III+ v2.0 para PC,

QSB+ para PC,

Lindo para PC,

GraphEcon II para PC,

etc.

El software antes instalado, se encuentra en un servidor Novell 3.12 para el servicio de PC's aunque en el caso de una contingencia con la red, es posible trabajar localmente ya que dichos paquetes se encuentran instalados en disco duro de cada máquina.

El CEC se encuentra organizado en islas a razón de 16 máquinas por isla (4 de WS, 4 de PC y 4 de Mac); en cada isla de PC se da **login** (acceso a la red) y mediante una dirección Ethernet cada isla accesa a una cuenta. Así, se tienen cuatro cuentas de Novell correspondiendo cada una a una isla.

Adicionalmente, se dispone de una cuenta más, que aloja la *imagen* del software que se emplea.

El caso de los equipos Macintosh, es distinto ya que si bien se encuentran instaladas en red, solamente emplean la red para fines de impresión y acceso a Internet, por lo que la paquetería estándar, se encuentra instalada en disco duro.

De igual manera que en PC, se dispone de una computadora Macintosh dedicada exclusivamente a contener la imagen de los paquetes que se deben encontrar en todas las máquinas de su tipo.

Las workstation, se encuentran conectadas a un servidor IBM RS/6000 mod 990, que se encarga de ofrecer las aplicaciones de este tipo de máquina y adicionalmente, es el servidor de comunicaciones para la red del campus y su acceso satelital a Internet.

- ***Zona de Impresión***

Se puede realizar mediante la red que da servicio al CEC, y es posible que desde cada una de las computadoras que se encuentran conectadas dependiendo de su tipo, se genere la impresión.

El equipo de impresión de que se dispone en el CEC, es el siguiente:

- Dos impresoras Hewlett Packard modelo IVsi de 6ppm y 4Mb de RAM para dar servicio a las workstation RS/6000.
- Una impresora Hewlett Packard modelo IIIsi de 4ppm y 2Mb en RAM para servicio a Macintosh.
- Una impresora Hewlett Packard modelo Ivsi de 6ppm y 4Mb en RAM para servicio a Macintosh.
- Dos impresoras Xerox modelo 4220 de 30 pum y 20Mb de RAM para servicio a PC.
- Una Impresora Xerox modelo 4700 II de 20 ppm a 256 millones de colores para servicio de todo el campus en impresiones a color.

La forma de empleo de los servicios de impresión, es desde el CEC, de cualquiera computadora, emplear la impresora predeterminada y la impresión se generará en la impresora correspondiente en la Zona de Impresión donde se puede recoger en breve.

En el caso de las impresiones a color (a diferencia de las blanco y negro que son gratuitas), se cobra una cantidad por cada hoja impresa, la impresora se encuentra en estado de *retención*, en el cual solamente se imprimirán las copias, previa presentación del recibo de pago correspondiente.

La impresora a color, es visible por todas las computadoras ya que solamente se imprimirán las que se amparen por el recibo antes citado. Las impresoras de Macintosh, solamente son visibles desde ese tipo de máquinas y las impresoras de PC, son visibles desde PC y Macintosh.

I.2 UBICACION ORGANIZACIONAL DENTRO DEL INSTITUTO.

El ITESM-CCM está organizado en un esquema de direcciones en el que se diferencian las concernientes a las áreas de servicios y a las áreas académicas.

Las áreas académicas, son las abocadas a la impartición de cátedras en el instituto, lo que implica que todo el profesorado, los coordinadores de carrera, los directores de áreas académicas y otros funcionarios académicos, pertenecen a estas áreas.

Las áreas de servicios, son las dedicadas a la prestación de los servicios que las áreas académicas así como el alumnado, requieran. Bajo este marco, la labor de ambas áreas, es esencial para el correcto funcionamiento de instituto como entidad educativa.

Las áreas académicas son las siguientes:

División de Preparatoria,

División de Ingeniería,

División de Administración y Ciencias Sociales,

División de Graduados e Investigación,

Dirección de Promociones,

Dirección de Servicios Escolares,

Dirección de Desarrollo Académico,

Dirección de Informática,

Dirección de Servicios Administrativos,

Dirección de Planta Física,

Dirección de Asuntos Estudiantiles.

Sus respectivas funciones dentro de la estructura del instituto, son las siguientes:

⇨ *División Preparatoria.* Supervisa el desempeño y la carga académica de los profesores, así como el funcionamiento del programa y los planes de estudio en vigor del área de Preparatoria.

⇨ *División de Ingeniería.* Se encarga de la supervisión del personal docente y de su carga académica. Así como el funcionamiento de las ingenierías con respecto a los planes de estudio en vigor.

⇨ *División de Administración y Ciencias Sociales.* De igual manera que las divisiones anteriores, supervisa a los profesores y su carga académica y elabora las adecuaciones pertinentes a los planes de estudio de las licenciaturas a su cargo.

- *División de Graduados e Investigación.* Además de las funciones de las áreas anteriormente citadas, diseña, promueve, organiza y coordina la operación de programas de educación continua poniendo especial interés a los dirigidos al desarrollo de ejecutivos.

- *Dirección de Promociones.* Es la responsable de difundir los servicios y programas académicos con el fin de atraer al campus a los mejores alumnos.

- *Dirección de Servicios Escolares.* Es la responsable de la organización y coordinación del reclutamiento, admisión e inscripción de los alumnos, elaboración de horarios de cursos, control de expedientes, captura y procesamiento de la información del historial académico, expedición de certificados y elaboración o detección de indicadores académicos.

- *Dirección de Desarrollo Académico.* Es la responsable del reclutamiento, selección inducción, capacitación y desarrollo de todo el personal docente y profesionista del campus.

- *Dirección de Informática.* Es la responsable de proporcionar todo el apoyo en materia computacional, como: equipo, software, coordinación de la red satelital, así como de la coordinación y administración de la biblioteca.

⇨ *Dirección de Servicios Administrativos.* Es la responsable de la información contable y financiera, del control presupuestal, de las compras de materiales y equipo, de la mensajería interna y externa, de las cafeterías y de la selección y desarrollo del personal administrativo y de servicios, así como de la contratación y remuneración de todo el personal del campus.

⇨ *Dirección de Planta Física.* Es la responsable de la conservación y mantenimiento de las instalaciones y equipos, los servicios de aseo, los servicios de transportación y la seguridad de las instalaciones. Así como del adecuado funcionamiento de la planta de tratamiento de aguas residuales.

⇨ *Dirección de Asuntos Estudiantiles.* Es la responsable de propiciar el ambiente universitario que brinde atención a las necesidades, retos y habilidades de la comunidad estudiantil a través de fomentar y preservar la salud física y sicológica, desarrollar actitudes de apreciación de las diversas manifestaciones culturales y artísticas, implementar programas de apoyo económico, promover la educación física, la recreación y el deporte, generar espacios y estrategias de comunicación efectiva y apoyar la integración social y el desarrollo de eventos y proyectos especiales.

La dirección de informática, como ya se mencionó, es la encargada de dotar al campus, del soporte computacional necesario para el desarrollo del campus, tanto el soporte a las otras áreas de servicio como del apoyo a las áreas académicas.

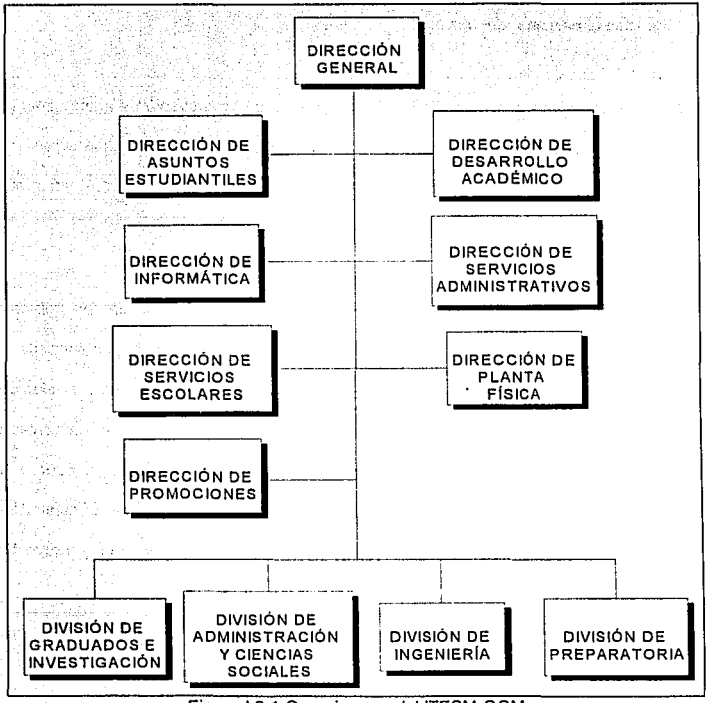


Figura I.2.1 Organigrama del ITESM-CCM.

Para su organización, la dirección de informática, se conforma en una serie de departamentos que son a saber:

- Servicios de Apoyo,
- Telecomunicaciones,
- Telefonía,
- Proyectos Especiales,
- Sistemas Abiertos,
- Biblioteca,
- Servicios Computacionales.

Las funciones principales de cada uno de los departamentos es la siguiente:

Servicios de Apoyo. Es el departamento encargado, de proveer la infraestructura necesaria para la impartición de clases y el soporte audiovisual académico.

Ofrece la posibilidad de emplear servicios de adujo, retroproyectores, datashow (proyector de cristal líquido), reproductor de disco láser, videograbadora, cañón de video, etc. Previa reservación del equipo.

Telecomunicaciones. Es la encargada de proveer los servicios de red e interconectividad que se requieren en el campus, administra el sistema de correo

electrónico, realiza las instalaciones de red así como las configuraciones de los servidores (únicamente Novell) existentes en el campus. Es responsable de la asignación de cuentas y la seguridad de la información en la red. Realiza los trabajos necesarios para la interconexión de los equipos Macintosh, IBM PS, AIX, Santa Cruz Operation, Solaris, Silicon Graphics y en general, de todas las plataformas que se emplean en el ITESM-CCM.

Telefonía. Es el departamento abocado a la comunicación telefónica dentro del campus. Son responsables del PABX modelo Meridian II de Northern Telcom que se emplea para la comunicación de voz y datos. Asigna también las extensiones telefónicas así como los aparatos telefónicos y los facsímiles. Cuenta también, con un servicio de recepción de solicitudes de servicios de informática mediante extensiones dedicadas que reciben los reportes y los canalizan a las instancias debidas.

Proyectos Especiales. En este departamento, se realizan las labores correspondientes a los diversos servicios de informática que tienen una duración breve o que dependen de apoyo de entidades ajenas al ITESM-CCM (Apple-Macintosh, Sun MicroSystems, Silicon Graphics, Microsoft, etc.); se encarga por ejemplo, de ofrecer los cursos de capacitación para empresas y particulares, de mantener las áreas de multimedios y animación por computadora y videoproceso.

Sistemas Abiertos. En este departamento, se administran todos los recursos de cómputo de equipos UNIX como son las computadoras IBM RS/6000 modelos 220, 250, 970 y 990, servidores de UNIX SantaCruz Operation y de la red de workstations de las máquinas antes citadas. Manejan además, el software que se emplea en dichos equipos y de la administración de las cuentas de acceso a la red Internet. Es su responsabilidad el correcto funcionamiento de los servicios que ofrece la red mundial de computadoras Internet. Supervisa el funcionamiento de las salas de cómputo de equipos RS/6000 y HP Apollo.

Biblioteca. En este departamento, se verifica lo relacionado al acervo bibliográfico, hemerográfico y de medios masivos de información electrónica que se ofrecen en el campus. Administra las áreas de préstamo de acervo, adquisición de colecciones y catalogación en línea mediante el sistema Dynix. Supervisa además, el correcto funcionamiento del Centro de Información Financiera y Económica (CIFE) que es el encargado de obtener electrónicamente, la información más reciente en el ramo.

Servicios Computacionales. Este departamento es uno de los más bastos que se tienen en la dirección de informática, ya que en él, se administran los servicios de apoyo informático para el área de Servicios Escolares (equipo HP 3000 de Hewlett Packard), el desarrollo de los sistemas de cómputo que se requieren en el campus (nómina, personal, desarrollo académico, etc.), la administración y operación del Centro de Evaluación Automatizada (en el que se aplican y evalúan exámenes de

alumnos automáticamente), el centro electrónico de cálculo, las salas de cómputo de equipo PC y Macintosh y todo lo concerniente al software de equipos PC y Macintosh.

La coordinación de atención a usuarios, se encuentra ubicada dentro del departamento de Servicios Computacionales y se encuentra a la par de otras coordinaciones que son las siguientes:

- Equipos Mayores,
- Desarrollo de Sistemas,
- Centro de Evaluación Automatizada,
- Atención a Usuarios.

Las funciones de cada coordinación, se detallan a continuación:

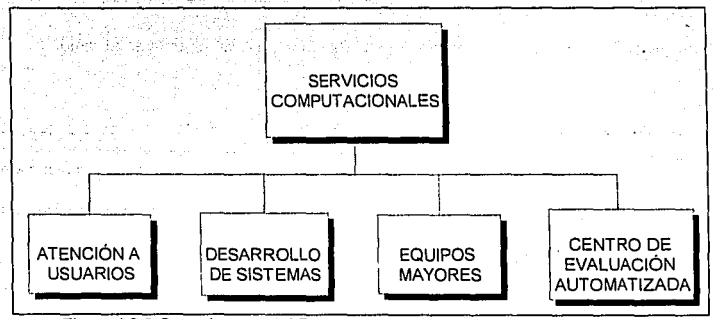


Figura I.2.2 Organigrama del Departamento de Servicios Computacionales.

Equipos Mayores. En esta coordinación, se realiza la administración del equipo HP 3000 en el que reside toda la información de Servicios Escolares como los datos personales de cada alumno, su historia académica y todos los detalles de su trayectoria en el ITESM-CCM.

Desarrollo de Sistemas. Los sistemas que el ITESM-CCM requiere para su operación cotidiana, se elaboran en esta coordinación, la cual tiene a su cargo el desarrollo de nuevos sistemas, así como el mantenimiento de los sistemas operantes como son el de nómina, desarrollo académico, personal y otros.

Centro de Evaluación Automatizada. Es la coordinación que se encarga de la operación, la confiabilidad y la seguridad en el Centro de Evaluación Automatizada, que es el sitio donde, a partir de bancos de preguntas, se elabora un examen personal para cada alumno de las distintas materias, se ofrece el espacio para su resolución, y se entrega la evaluación en aproximadamente 2 minutos después de la entrega del examen mediante pantallas de televisión que se encuentran a la salida del CEA.

Atención a Usuarios. Las funciones de la coordinación de atención a usuarios, ya fueron objeto de un texto en este trabajo y se pueden consultar en el tema I.1.

I.2.1 CONCEPTOS Y DEFINICIONES MANEJADAS POR ESTA COORDINACION.

Como es de suponer, en cualquier organización es necesario comprender la jerga que se emplea para la definición de los procesos internos a fin de compenetrarse con la estructura y la terminología que se utiliza.

En el caso de la coordinación de atención a usuarios, se manejan una serie de términos que es necesario que se aclaren antes de avanzar en el estudio de la misma.

Se definen los siguientes términos:

- *Usuario.* Se entiende por usuario, toda aquella persona que en sus diferentes modalidades, tienen derecho a hacer uso de los servicios del Centro Electrónico de Cálculo.
- *Máquina.* Se denomina de esta forma, a cada uno de los equipos de cómputo de que se dispone en el CEC sin distinción del tipo de equipo del que se esté hablando (Workstation, Macintosh o PS Value Point).
- *Apartado.* El apartado, es el acto de reservar el derecho de uso de algún equipo de cierto tipo para una hora específica en un día determinado. Mediante el apartado, el usuario garantiza que se dispone del equipo, el cual le será asignado para su uso en las especificaciones acordadas.

- *Tipo de Máquina.* Es una característica de las máquinas, que depende de su arquitectura y fabricante. Por el momento, solamente se dispone de tres tipos de máquinas que son las Workstation RS/6000, las Power Mac Macintosh, y las PS Value Point IBM.
- *Servidor.* El concepto de servidor, desde el punto de vista de la coordinación, no es el de un equipo que da servicio a una serie de estaciones conectadas entre si, sino que se emplea, simplemente para determinar a qué servidor se encuentra conectado cada equipo del CEC. Por lo que la única referencia a servidor, será para identificar las distintas partes de la red de computadoras en el caso de fallas en la misma.
- *Tipo de Usuario.* Como ya se mencionó, no toda la gente que ingresa al CEC, tiene la misma característica que le permite la entrada, sino que pueden entrar por diversos motivos. Por ello, se maneja de manera separada, a cada usuario dependiendo de el motivo que origina que ingrese al CEC. Se dispone actualmente de cuatro tipos de usuarios con derecho de ingreso al CEC que son: Alumnos, Personal Docente, Personal Administrativo, Exatec (Ex-alumnos del ITESM-CCM) y Visitantes (en Proyectos Especiales).
- *Entrada.* Se realiza una entrada, cuando se ocupa un equipo por un usuario a una hora determinada.
- *Cancelación.* Cuando un usuario con un apartado decide no hacer uso del mismo y avisa a los encargados del CEC, se genera una cancelación.

- *Salida*. Tiene la finalidad de emplear con mayor eficiencia el equipo instalado, mediante el registro de salidas que se generan cuando una persona ha terminado de emplear el equipo que se le asignó y avisa tal evento al encargado del CEC.
- *Hora*. La hora es un aspecto importante ya que el reglamento del CEC previene que se asignarán máquinas en múltiplos enteros de horas, por lo que es importante registrar la hora en que se realiza algún evento como un apartado, una entrada o una salida.

De manera general, esta es la nomenclatura que se emplea en la coordinación, que sin embargo, se puede ver enriquecida por otros términos que se irán definiendo según se requiera.

Para reforzar la situación actual del ITESM-CCM, se anexa un croquis que ilustra su ubicación en la Ciudad de México.

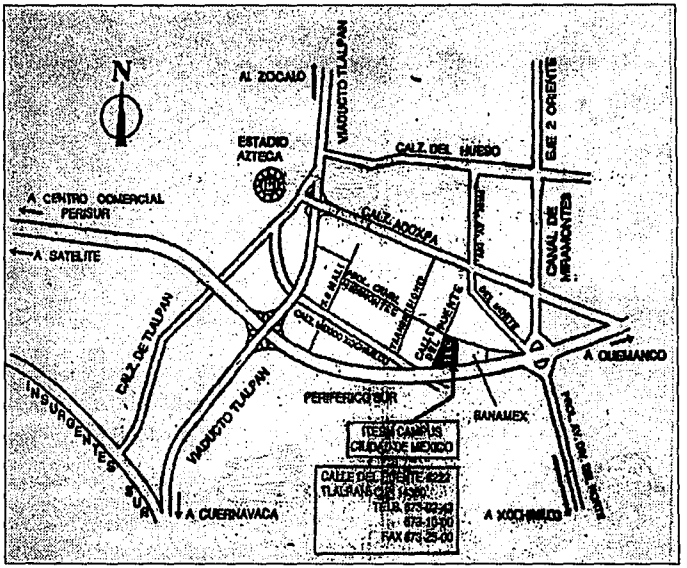


Figura I.2.1.1 Localización del ITESM-CCM.

Capítulo II

Conceptos de Análisis, Diseño de Sistemas y Redes

II.1 BASES DE DATOS RELACIONALES

II.1.1 SISTEMA DE BASE DE DATOS

La informática ha ido evolucionando y las necesidades de los usuarios son cada vez mayores. A los tratamientos masivos de archivos en los centros de cálculo sucedieron aplicaciones explotadas en tiempo real desde terminales conectadas a una computadora central a través de diferentes medios de comunicación.

Los tiempos de respuesta exigidos y el acceso concurrente de varios usuarios a un mismo archivo para su actualización, hicieron que las estructuras de archivos fueran insuficientes.

Cuestiones como la integridad de los archivos, mecanismos de seguridad ante caídas del sistema, etc. empiezan a preocupar y la imposibilidad de que cada programador se ocupe de estos temas en cada programa hacen que aparezcan los primeros Sistemas de Administración de Base de Datos (DBMS, Data Base Managment System).

El objetivo primordial de un DBMS es crear un ambiente en que sea posible guardar y recuperar información de la Base de Datos en forma conveniente y eficiente.

Un sistema de bases de datos consiste en un conjunto de datos relacionados entre si y un grupo de programas para tener acceso a esos datos. El conjunto de datos se conoce comúnmente como Base de Datos.

Los sistemas de bases de datos se diseñan para manejar grandes cantidades de información. El manejo de los datos incluye tanto la definición de las estructuras para el almacenamiento de la información como los mecanismos para el manejo de la misma. Además, el sistema de base de datos debe cuidar la seguridad de la información almacenada en la base de datos, tanto contra las caídas del sistema como contra los accesos no autorizados. Si los datos van a ser compartidos por varios usuarios, el sistema debe evitar la posibilidad de obtener resultados anómalos.

Debido a la importancia que tienen la información en casi todas las organizaciones, la base de datos es un recurso valioso. Esto condujo al desarrollo de un gran número de conceptos y técnicas para manejar los datos en forma eficiente.

ABSTRACCION DE LA INFORMACION

Un sistema de manejo de base de datos es un conjunto de archivos interrelacionados y una serie de programas que permiten a varios usuarios tener accesos a estos archivos y modificarlos.

Sin embargo, para que el sistema sea útil, la información se debe recuperar en forma eficiente. Uno de los objetivos principales de un sistema es proporcionar a los usuarios una visión abstracta de la información. Es decir, el sistema oculta ciertos detalles relativos a la forma como los datos se almacenan y mantienen.

La búsqueda de la eficiencia conduce al diseño de las estructuras de datos complejas para representar la información de la base de datos. Pero como los sistemas de base de datos muchas veces son utilizados por personal que no cuenta con conocimientos de computación, esta complejidad debe estar escondida para los usuarios. Para ocultarla, se definen varios niveles de abstracción en los que puede observarse la base de datos:

- **Nivel físico.** Este es el nivel más bajo de abstracción, en el que se describe cómo se almacenan realmente los datos. En este nivel se describen en detalle las estructuras de datos complejas del nivel más bajo.

- **Nivel conceptual.** En este nivel se describen cuáles son los datos reales que están almacenados en la base de datos y que relaciones existen entre ellos. Este nivel contiene toda la base de datos en términos de unas cuantas estructuras relativamente sencillas. Aunque es posible que la implantación de las estructuras simples del nivel conceptual requiera estructuras complejas en el nivel físico, no es

forzoso que el usuario del nivel conceptual se dé cuenta de ello. El nivel conceptual de abstracción lo utilizan los administradores de base de datos; quienes deciden qué información se guarda en la base de datos.

- **Nivel de visión.** Este es el nivel de abstracción más alto, en el cual se describe una parte solamente de la base de datos. Aunque en el nivel conceptual se utilizan estructuras más simples, todavía queda una forma de complejidad que resulta de gran tamaño de la base de datos. Muchos usuarios no tendrán que ocuparse de toda está información, más bien necesitarán solamente una parte de la base de datos. Para simplificar la interacción entre estos usuarios y el sistema, se define el nivel de abstracción visión. El sistema puede proporcionar muchas vistas diferentes de la misma base de datos.

INSTANCIAS Y ESQUEMAS DE UNA BASE DE DATOS

Las bases de datos cambian con el tiempo al insertarse y eliminarse información en ellas. El conjunto de información almacenado en la base de datos en cierto momento se denomina una instancia en la base de datos. El diseño general de la base de datos se llama esquema de la base de datos. Los esquemas se alteran muy raras veces, o nunca.

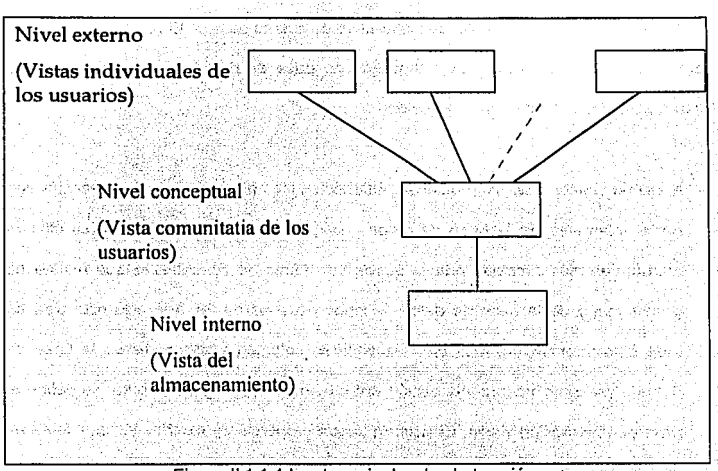


Figura II.1.1.1 Los tres niveles de abstracción.

Existen varios esquemas en la base de datos, y estos de dividen de acuerdo con los niveles de abstracción ya mencionados. En el nivel más bajo se tiene el esquema físico; en el nivel intermedio está el esquema conceptual, mientras que en el nivel más alto existe un subesquema.

INDEPENDENCIA DE LOS DATOS

Se definieron tres niveles de abstracción en los cuales se puede ver a la base de datos. La capacidad de modificar una definición de esquema en un nivel sin afectar la definición del esquema en el nivel inmediato superior se denomina independencia de los datos.

Existen dos niveles de tal independencia:

- **Independencia física.** Es la capacidad de modificar el esquema conceptual sin obligar a que se vuelvan a escribir los programas de aplicaciones. En algunas ocasiones son necesarias modificaciones en el nivel físico para mejorar el rendimiento.

- **Independencia lógica.** Es la capacidad de modificar el esquema conceptual sin obligar a que se vuelvan a escribir los programas o aplicaciones. Las modificaciones en el nivel conceptual son necesarias siempre que se altera la estructura lógica de la base de datos.

La independencia lógica de los datos es más difícil de lograr que la independencia física, ya que los programas de aplicaciones dependen en alto grado de la estructura lógica de los datos a los que tienen acceso.

El concepto de independencia de los datos es similar en muchos aspectos al concepto de tipos abstractos de datos en los lenguajes de programación modernos. Ambos ocultan los detalles de la puesta en marcha, lo cual permite a los usuarios concentrarse en la estructura general más bien que en los detalles de la implantación en el nivel más bajo.

LENGUAJE DE DEFINICION DE DATOS

Un esquema de base de datos se especifica por medio de una serie de definiciones que se expresan en un lenguaje especial llamado lenguaje de definición de datos (**DDL**, data definition lenguaje). El resultado de la compilación de las proposiciones en DDL es un conjunto de tablas que se almacena en un archivo especial llamado diccionario (o directorio) de datos.

Un **diccionario de datos** es un archivo que contiene metadatos, es decir, datos acerca de los datos. Este archivo se consulta antes de leer o modificar los datos reales en el sistema de base de datos.

La estructura de almacenamiento y los métodos de acceso empleados por el sistema de base de datos se especifican por medio de un conjunto de definiciones de un tipo especial de DDL llamado lenguaje de almacenamiento y definición de los datos. El resultado de la compilación de estas definiciones es una serie de instrucciones que

especifican los detalles de implantación de los esquemas de base de datos que normalmente no pueden ver los usuarios.

LENGUAJE DE MANEJO DE DATOS

Los niveles de abstracción que se mencionaron no solamente se aplican a la definición o estructuración de los datos, sino también al manejo de los datos; esta manipulación consiste en:

- La recuperación de información almacenada en la base de datos.
- La inserción de información nueva en la base de datos.
- La eliminación de información en la base de datos.

En el nivel físico, deben definirse algoritmos que permitan tener acceso a los datos en forma eficiente. En los niveles de abstracción más altos lo importante es la facilidad de uso. El objetivo es lograr una interacción eficiente entre las personas y el sistema.

Un lenguaje de manejo de datos (**DML**, data manipulation language) permite a los usuarios manejar o tener acceso a los datos que estén organizados por medio del modelo apropiado. Existen básicamente dos tipos de DML:

- De procedimientos, necesitan que el usuario especifique cuáles datos quiere y como deben obtenerse.

- Sin procedimientos, requieren que el usuario especifique cuáles datos quiere sin especificar como obtenerlos.

Los DML sin procedimientos son por lo general más fáciles de aprender y utilizar que los de procedimientos. Sin embargo, ya que el usuario no tiene que especificar la forma de obtención de los datos, estos lenguajes podrían generar un código menos eficiente que el producido por los lenguajes de procedimientos. Tal problema puede resolverse empleando diversas técnicas de optimización.

Una consulta es una proposición que solicita la recuperación de información. La parte de un DML que implica la recuperación de información se conoce como lenguaje de consultas. Aunque técnicamente es incorrecto, suelen utilizarse los términos lenguaje de consultas y lenguaje de manejo de datos como sinónimos.

MANEJADOR DE BASE DE DATOS

Generalmente las bases de datos requieren una gran cantidad de espacio y almacenamiento. Las bases de datos de las empresas comúnmente se miden en términos de gigabytes de información. Un gigabyte equivale a 1000 megabytes o mil

millones de bytes. Puesto que la memoria principal de la computadora no puede almacenar esta información , se guarda en discos. Los datos se transfieren entre el almacenamiento en disco y la memoria principal, según se requiera. Ya que el movimiento de los datos del disco y al disco es lento comparado con la velocidad de la unidad central de procesamiento de las computadoras, es imperativo que el sistema de base de datos estructure la información de tal manera que se reduzca la necesidad de transferir datos entre el disco y la memoria principal.

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

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

Un manejador de base de datos es un módulo de programa que constituye la interface entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicaciones y las consultas hechas al sistema. El manejador de base de datos es responsable de las siguientes tareas:

- **Interacción con el manejador de archivos**. Los datos sin procesar se almacenan en el disco mediante el sistema de archivos proporcionado normalmente por un sistema operativo convencional. El manejador de base de datos traduce las diferentes proposiciones en DML a comandos de sistema de archivos de bajo nivel. Así , el manejador de base de datos se encarga realmente del almacenamiento, recuperación y actualización de los datos en la base de datos.

- **Implantación de la integridad**. Los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de limitaciones de consistencia. Si se especifican estas limitaciones. Entonces el manejador de la base de datos puede verificar si las actualizaciones a la base de datos resultan en la violación de cualquiera de estas limitantes, y si así es, podrá realizar la acción apropiada.

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

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

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

Algunos sistemas de bases de datos, diseñados para utilizarse en computadoras personales pequeñas, no cuentan con varias de las funciones mencionadas. Esto da como resultado un manejador de datos de menor tamaño. Un manejador de datos pequeño requiere menos recursos físicos, sobre todo memoria principal, y su implantación es mas económica. Por ejemplo, muchos sistemas pequeños incluyen la restricción de que sólo un usuario puede tener acceso a la base datos en un momento dado. Otros dejan las tareas de respaldo, recuperación e implantación de la seguridad al usuario. Aunque este enfoque de bajo costo y funciones limitadas es suficiente para

bases de datos personales, no es adecuada para cumplir con los requerimientos de una empresa mediana o grande.

ADMINISTRADOR DE BASES DE DATOS

Una de las razones principales para contar con sistemas de manejo de base de datos es tener un control centralizado tanto de los datos como de los programas que tienen acceso a ellos. La persona que tiene este control centralizado sobre el sistema es el **DBA** (administrador de bases de datos). Las funciones del DBA son:

- Definición de esquema, es decir, la creación del esquema original de la base de datos. Esto se logra escribiendo una serie de definiciones que el compilador de DDL traduce a un conjunto de tablas que se almacenan permanentemente en el diccionario de datos.
  Definición de la estructura de almacenamiento y del método de acceso. Esto se lleva a cabo escribiendo una serie de definiciones que posteriormente son traducidas por el compilador del lenguaje de almacenamiento y de definición de datos.

- Modificación del esquema y de la organización física. Esto se logra escribiendo una serie de definiciones utilizadas ya sea por el compilador de DDL o por el compilador

del lenguaje de almacenamiento y definición de datos para generar modificaciones a las tablas internas apropiadas del sistema, por ejemplo el diccionario de datos.

Concesión de autorización para el acceso a los datos. Esto permite al DBA regular cuales son las partes de la base de datos a las que van a tener acceso diversos usuarios.

- Especificación de las limitantes de integridad. Estas se conservan en una estructura especial del sistema que consulta el DBA cada vez que se lleva a cabo una actualización en el sistema

ESTRUCTURA GENERAL DE UN SISTEMA DE BASES DE DATOS

Un sistema de base de datos se divide en módulos que se encargan de cada una de las tareas del sistema general. Algunas de las funciones del sistema de base de datos pueden ser realizadas por el sistema operativo. En la mayor parte de los casos, el sistema operativo proporciona únicamente los servicios más elementales y la base de datos debe partir de ese fundamento. Así, el diseño de la base de datos debe incluir una consideración de la interfaz entre el sistema de base de datos y el sistema operativo.

Un sistema de base de datos consiste en varios componentes.

- **El manejador de archivos.** Se encarga de asignar espacio en el disco y de las estructuras de datos que se van a emplear para representar la información almacenada en el disco.

- **El manejador de bases de datos.** Constituye la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicaciones y las consultas que se hacen al sistema.

- **El procesador de consultas.** Traduce los proposiciones en lenguaje de consulta a instrucciones de bajo nivel que puede entender el manejador de la base datos. Además, el procesador de consultas trata de convertir la solicitud del usuario a una forma equivalente pero más eficiente, encontrando una estrategia adecuada para ejecutar la consulta.

- **El precompilador de DML.** Convierte las proposiciones en DML incrustadas en un programa de aplicaciones llamadas normales a procedimientos en el lenguaje huésped. El precompilador debe interactuar con el procesador de consultas para generar el código apropiado.

- **El compilador DDL.** Convierte Las proposiciones en DDL en un conjunto de tablas que contienen metadatos. Tales tablas se almacenan después en el diccionario de datos.

Además, se requieren varias estructuras de datos como parte de la implantación del sistema físico, incluyendo:

- **Archivos de datos**. Guardan la base de datos.

- **Diccionario de datos**. Almacena la información relativa a la estructura de datos. Se usa constantemente, por lo que debe tenerse mucho cuidado de desarrollar un diseño apropiado y una implantación eficiente.

- **Indices**. Permiten el acceso rápido a elementos de información que contienen valores determinados

Existen diferentes tipos de sistemas de Administración de Bases de Datos, de ellos hay tres claramente diferenciados surgidos a lo largo del tiempo:

- ***Jerárquico***
- ***En red***
- ***Relacional***

Jerárquico: Los datos están estructurados en forma arborescente y las relaciones entre los diferentes tipos de registros se resuelven mediante punteros o enlaces entre ellos.

Se establece una jerarquía de modo que las relaciones entre un registro y otro relacionado con él (relación padre-hijo) tienen como condición que un registro "hijo" no puede existir si no existe el registro "padre" asociado a él.

La estructura sería del modo:

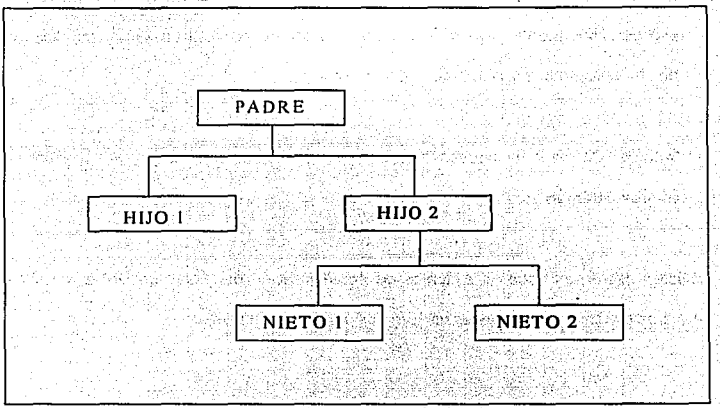


Figura II.1.1.2 Estructura jerárquica

Las características principales de éste tipo de Base de Datos son, su estructura arborescente, la inclusión de los punteros en los registros de datos y la posibilidad de tener diferentes subesquemas.

En red: Si en la estructura arborescente del modelo jerárquico, se permiten relaciones entre "hermanos", es decir, entre registros de un mismo padre lógico, se tendrá una Base de Datos en red.

En ella se pueden construir esquemas del tipo siguiente:

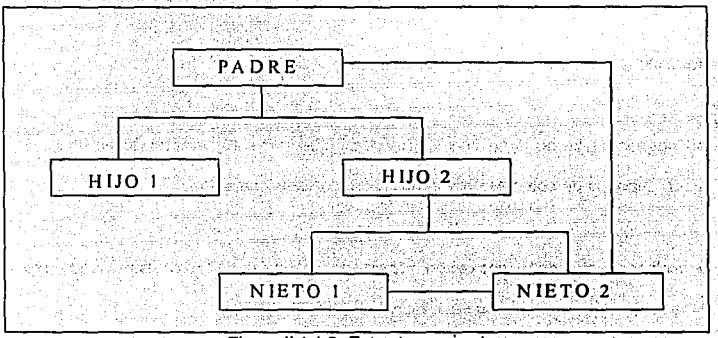


Figura II.1.1.3 Estructura en red

Debe considerarse a la Base de Datos Jerárquica como un tipo de Base de Datos en Red con ciertas restricciones.

Relacional: Se describe en la siguiente sección.

II.1.2 EL MODELO RELACIONAL

El modelo relacional de datos es propuesto en 1970 por el Dr. F. Codd que lo define por medio de una serie de reglas cuyo objetivo es lograr la independencia de la representación lógica de los datos en su almacenamiento físico.

Esta independencia física/lógica se refiere a tres aspectos:

- Independencia de la ordenación, es decir, que el resultado obtenido en un acceso no dependa de cómo estén ordenados los datos físicamente.

- Independencia de la indexación, separando los índices de los datos haciendo que la creación y mantenimiento sean manejados por el sistema.

- Independencia de los caminos de acceso, haciendo que la navegación a través de los datos no tenga que estar previamente establecida consiguiendo así unas formas de acceso más flexibles.

Por ello, Codd a través de sus reglas pretende los siguientes objetivos:

- Independencia física/lógica

- Eliminación de redundancia
- Flexibilidad
- Uniformidad
- Sencillez
- Sólido fundamento teórico

CONCEPTOS FUNDAMENTALES

Una característica atractiva del modelo relacional es su sencillez. Una relación es una tabla de datos, y puede constar sólo de una fila y una columna, proporcionando así la estructura de datos más sencilla posible que puede utilizarse como común denominador de todas las estructuras de datos.

Esto simplifica el diseño del esquema conceptual, puesto que hay sólo una estructura lógica de datos, esto es, la relación, a considerar.

Además, el modelo relacional proporciona una libertad sin precedentes al programador de aplicaciones, permitiéndole acceder directamente a cualquier valor de atributo de la

base de datos; el mecanismo de acceso asociativo o direccionable por el contenido como atributo es aparentemente por su valor, en vez de por su posición o por un puntero.

En terminología relacional cada fila de la Tabla se conoce por Tupla y cada columna por Atributo.Cardinalidad de una relación será el número de Tuplas que la componen y Grado el número de atributos o columnas de la misma.

Las Relaciones son Tablas con las siguientes propiedades:

- Todos los elementos o entradas de una columna son del mismo tipo
- Se asignan nombre distintos a las columna, llamados nombres de atributo
- No hay Tuplas iguales
- El orden de las Tuplas no es significativo
- El orden de los Atributos no es significativo
- En una Tupla determinada, cada Atributo puede tomar sólo un valor (no se admite grupos repetitivos)

La extensión de una relación es el conjunto de las tuplas de la relación en un determinado momento, y por tanto, varía con el tiempo, pues se van insertando o borrando tuplas. Otro termino asociado es intensión, que se refiere a las características permanentes de una relación, tales como nombre de la relación y nombres de los atributos, incluyendo todas las restricciones de integridad pertinentes.

La intensión de una relación puede encontrarse en el esquema conceptual de la base de datos, y la extensión en los datos de la base de datos; la intensión define todas las extensiones permisibles.

El término dominio significa una colección de valores con el mismo tipo de propiedades de los cuales uno o más atributos o columnas obtienen sus valores reales. Aunque en algunos casos entre columnas y dominios hay una correspondencia uno a uno en otros no lo hay. El mismo dominio puede ser compartido por diferentes columnas, cada una con nombres de atributos posiblemente diferentes, en la mismo o en diferentes relaciones.

La figura II.1.2.1 muestra un ejemplo de este tipo de base de datos, la de proveedores y partes, utilizaremos ésta en casi todos los ejemplos de esta sección.

S

S#	SNOMBRE	SITUACION	CIUDAD
s1	Salazar	20	Londres
s2	Jaimes	10	París
s3	Bernal	30	París
s4	Corona	20	Londres
s5	Aldana	30	Atenas

SP

S#	P#	CANT
s1	p1	300
s1	p2	200
s1	p3	400
s1	p4	200
s1	p5	100
s1	p6	100
s2	p1	300
s2	p2	400
s3	p2	200
s4	p2	200
s4	p4	300
s4	p5	400

P

P#	PNOMBRE	COLOR	PESO	CIUDAD
p1	Tuerca	Rojo	12	Londres
p2	Perno	Verde	17	París
p3	Birlo	Azul	17	Roma
p4	Birlo	Rojo	14	Londres
p5	Leva	Azul	12	París
p6	Engrane	Rojo	19	Londres

Figura II.1.2.1 La base de datos de proveedores y partes

Como puede verse en la figura, la base de datos se compone de tres tablas, cuyos nombres son S, P Y SP.

- La tabla S representa proveedores. Cada uno tiene un número de proveedor (S#) único; un nombre de proveedor (SNOMBRE) no, por fuerza único, un valor de calificación o situación (SITUACION); y una localidad (CIUDAD). Supondremos que cada proveedor está situado en una sola ciudad.
- La tabla P representa tipos de parte, cada tipo tiene un número de parte (P#), el cual es único; un nombre de parte (PNOMBRE); un color (COLOR); un peso

(PESO); y una localidad donde se almacenan partes de ese tipo (CIUDAD). Suponemos que cada tipo de parte tiene un solo color y se almacena en una bodega de una sola ciudad.

- La tabla SP representa envíos. Sirve en cierto sentido para conectar entre sí las otras dos tablas. Por ejemplo, la primera fila de la tabla SP en la figura II.1.2.1 conecta un proveedor específico de la tabla S (el proveedor S1) con una parte específica de la tabla P (la parte P1); dicho de otro modo, representa un envío de partes de la clase P1 hecho por el proveedor S1 (y la cantidad enviada es 300). Así cada envío tiene un número de proveedor (S#), un número de parte (P#), y una cantidad (CANT).

Los proveedores y las partes pueden considerarse entidades, y un envío puede considerarse una interrelación entre un determinado proveedor y una parte específica, pero lo mejor es considerar las interrelaciones como un caso especial de las entidades.

Una de las ventajas de las bases de datos relacionales es que todas las entidades, aunque en realidad sean interrelaciones, se representan de la misma manera, es decir, mediante tablas.

CLAVES Y ATRIBUTOS

Las tuplas de una relación pueden identificarse de forma única por una o más claves, cada una conteniendo uno o más atributos no redundantes. Estas claves se llaman claves candidatas. Una de esas claves puede elegirse arbitrariamente para identificar las tuplas, esta clave se llama clave primaria. Si hay solamente una clave candidata para una relación, entonces esta clave debe ser la clave primaria. En principio, cada relación debe tener una clave primaria.

Si se consideran las relaciones S y P de la figura II.1.2.1 en la relación S, S# es la única clave candidata, y en consecuencia, también es la clave primaria, en el caso de la relación P, P# es la única clave candidata por lo que constituye su clave primaria.

Una clave candidata no debe incluir ningún atributo redundante. Otra restricción del modelo relacional es que una clave primaria no puede tener valores nulos, porque un valor nulo no puede identificar una tupla. Valor nulo en un símbolo convencional que se usa para representar una información desconocida o inaplicable. Esto se define como integridad de entidad.

Si un atributo (o colección de atributos) de una relación contiene el valor de clave primaria de otra relación ese atributo (o esa colección) se llama clave externa. Así pues, S# y P# son las claves externas de las relación SP. Una clave externa puede

tener sólo dos valores posibles, el valor de la clave primaria pertinente o un valor nulo.

No se permite ningún otro valor. Esto se conoce como integridad referencial, porque los valores se obtienen por referencia a los valores de clave primaria del propietario.

Una clave externa puede o no formar parte de una clave primaria de miembro, según la situación, en el ejemplo S# y P# son la clave primaria de la relación SP.

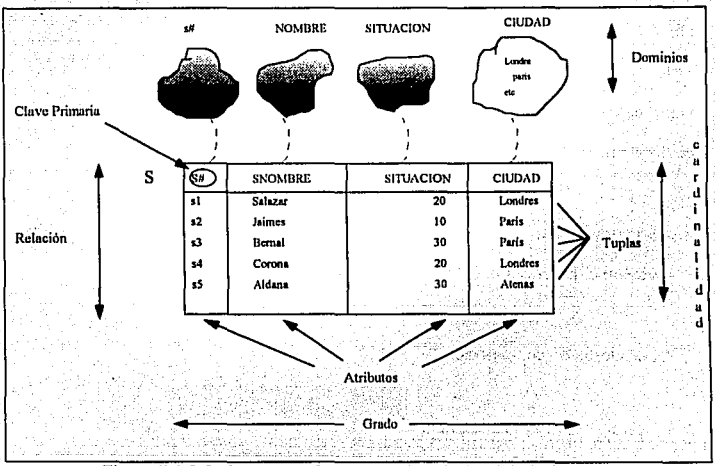


Figura II.1.2.2 Conceptos fundamentales del Modelo Relacional

DINAMICA DEL MODELO RELACIONAL

A la parte estática del modelo, es decir, a las estructuras vistas anteriormente y a sus propiedades y restricciones, está asociada otra parte que constituye la dinámica del modelo, esto es, una serie de operadores que permiten hacer transformaciones y manipular las relaciones definidas.

Estos operadores permitirán insertar una Tupla en una relación, modificarla, borrarla, etc.

Codd define esta dinámica del modelo relacional en 1972 y clasifica los operadores de la siguiente forma:

1. Operadores del álgebra relacional: Unión, intersección, diferencia, producto cartesiano, proyección, selección.

2. Operadores de combinación

3. Operaciones con valores nulos

Veamos estos operadores con algunos ejemplos.

UNION

Construye una relación formada por todas las tuplas que aparecen en cualquiera de las dos relaciones especificadas. La unión de dos relaciones A y B compatibles respecto a la unión, A UNION B, es una relación cuya cabecera es idéntica a la de A o B y cuyo cuerpo está formado por todas las tuplas t pertenecientes a A o a B (o a las dos).

Ejemplo: Sean A y B las relaciones presentadas en la figura II.1.2.3 (A contiene en términos intuitivos, los proveedores de Londres y B contiene los proveedores que suministran la parte P1). Entonces A UNION B - parte (a) de la figura - consistirá en los proveedores que o bien están situados en Londres, o que suministran la parte P1 (o las dos cosas). El resultado tiene tres tuplas y no cuatro ya que se eliminan las tuplas repetidas.

INTERSECCION

Construye una relación formada por aquellas tuplas que aparezcan en las dos relaciones especificadas. La intersección de dos relaciones compatibles respecto a la unión A y B, A INTERSECCION B, es una relación cuya cabecera es idéntica a la de A o B y cuyo cuerpo está formado por todas las tuplas t pertenecientes tanto a A como a B.

Ejemplo: Sean A y B las relaciones presentadas en la figura II.1.2.3 . Entonces A INTERSECCION B - parte (b) de la figura II.1.2.3 consistirá en los proveedores situados en Londres, y que suministran la parte P1.

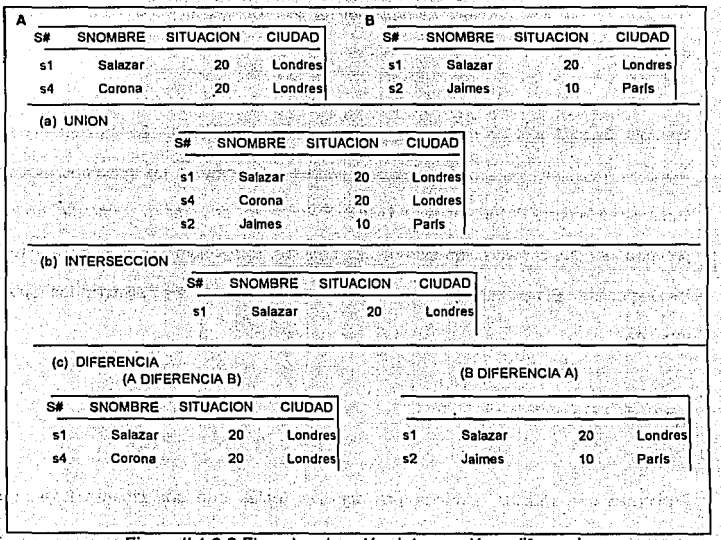


Figura II.1.2.3 Ejemplos de unión, intersección y diferencia

DIFERENCIA

Construye una relación formada por todas las tuplas de la primera relación que no aparezcan en la segunda de las dos relaciones especificadas.

La diferencia entre dos relaciones compatibles respecto a la unión A y B, A DIFERENCIA B, es una relación cuya cabecera es idéntica a la de A o B y cuyo cuerpo está formado por todas las tuplas t pertenecientes a A pero no a B.

Ejemplo: Sean A y B las relaciones presentadas en la figura II.1.2.3. Entonces A DIFERENCIA B contendrá los proveedores situados en Londres que no suministran la parte P1, y B DIFERENCIA A incluirá los proveedores que suministran la parte P1 y no están situados en Londres.

PRODUCTO CARTESIANO

A partir de dos relaciones especificadas, construye una relación que contiene todas las combinaciones posibles de tuplas, una de cada una de las dos relaciones.
El producto cartesiano (compatibles respecto al producto) A y B , A PRODUCTO B , se define como una relación cuya cabecera es la combinación de las cabeceras de A y B y cuyo cuerpo está formado por el conjunto de todas las tuplas t tales que t es la combinación de una tupla a perteneciente a A y una tupla b perteneciente a B.

Ejemplo: Sea A y B las relaciones presentadas en la figura II.1.2.4 (A en términos intuitivos, consiste en todos los números de proveedores vigentes y B en todos los números de parte vigentes).

Entonces A PRODUCTO B estará formada por todas las combinaciones de número de proveedores/número de parte vigentes.

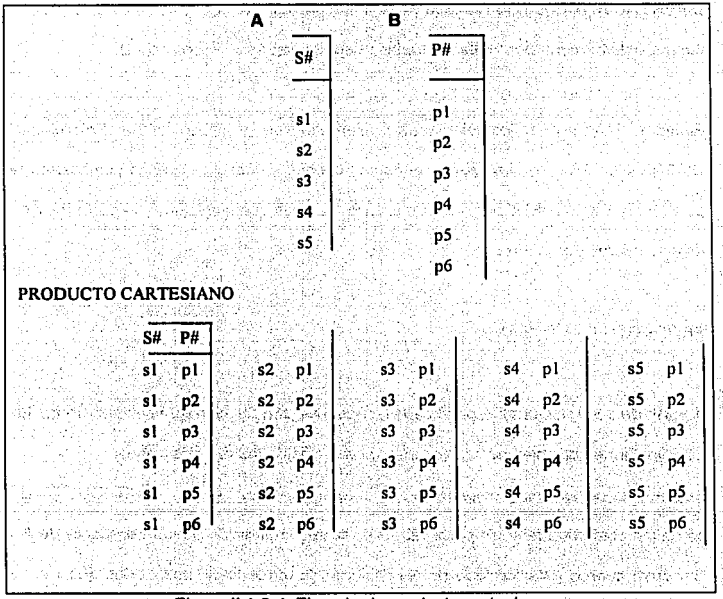


Figura II.1.2.4 Ejemplo de producto cartesiano

PROYECCION

Extrae los atributos especificados de una relación dada. La proyección de la relación A según los atributos X, Y, ...Z-

A (X, Y, ..., Z)

- Es una relación con (X, Y, ..., Z) como cabecera y cuyo cuerpo está formado por el conjunto de todas las tuplas (X:x, Y:y, ..., Z:z) tales que una tupla t aparece en A con el valor x en X, el valor y en Y .... y el valor z en Z. Así el operador de proyección produce un subconjunto "vertical" de una relación dada; o sea, el subconjunto obtenido mediante la selección de los atributos seleccionados.

En la figura II.1.2.5 se presentan algunos ejemplos de proyección. Obsérvese en el primer ejemplo (la proyección de proveedores según el atributo CIUDAD) que, aunque la relación S tiene cinco tuplas y por tanto cinco ciudades, solo hay tres ciudades en el resultado: las tuplas repetidas se eliminan.

Lo mismo puede decirse para el otro ejemplo.

S (CIUDAD)

CIUDAD
Londres
Paris
Atenas

P (COLOR, CIUDAD)

COLOR	CIUDAD
Rojo	Londres
Verde	Paris
Azul	Roma
Azul	Paris

Figura II.1.2.5 Proyección

SELECCION

Extrae las tuplas especificadas de una relación dada (o sea, restringe la relación solo a las tuplas que satisfagan una condición especificada).

Sea theta la representación de cualquier operador de comparación escalar simple (por ejemplo =, <>, >, >=, etc.). La restricción theta de la relación A según los atributos X y Y

-A WHERE X theta Y

- es una relación con la misma cabecera que A y con un cuerpo formado por el conjunto de todas las tuplas t de A tales que la evaluación de la comparación "X theta Y" resulta verdadera en el caso de esa tupla t.

S WHERE CIUDAD = 'Londres'

S#	SNOMBRE	SITUACION	CIUDAD
s1	Salazar	20	Londres
s4	Corona	20	Londres

P WHERE PESO < 14

P#	PNOMBRE	COLOR	PESO	CIUDAD
p1	Tuerca	Rojo	12	Londres
p5	Leva	Azul	12	París

SP WHERE S# = 'S1'
AND P# = 'P1'

S#	P#	CANT
s1	p1	300

Figura II.1.2.6 Ejemplos de selección

El operador de restricción theta produce un subconjunto "horizontal" de una relación dada; es decir, el subconjunto de las tuplas de la relación dada para las cuales se satisface una comparación especificada.

En la figura II.1.2.6 se presentan algunos ejemplos de restricción.

COMBINACION (JOIN)

A partir de dos relaciones especificadas, construye una relación que contiene todas las posibles combinaciones de tuplas, una de cada una de las dos relaciones, tales que las dos tuplas participantes en una combinación dada satisfagan alguna condición especificada.

La operación selección tiene varias formas distintas. Definitivamente, la más importante es la reunión natural, que se define como sigue.

Sean las cabeceras de las relaciones A y B

(X1, X2,...., Xm, Y1, Y2,..., Yn)

y

(Y1,Y2,....., Yn, Z1,Z2,...., Zp)

respectivamente; es decir, los atributos Y1, Y2, ... Yn son (los únicos) comunes a las dos relaciones, los atributos X1, X2,... Xn son los demás atributos de A, y los atributos Z1, Z2, ...Zp son los demás atributos de B.

Suponiendo que los atributos correspondientes, es decir con el mismo nombre, estan definidos sobre el mismo dominio.

Consideremos ahora (X1, X2, ... Xm), (Y1,Y2, ... Yn) y (Z , Z2, ... Zp) como tres atributos compuestos X,Y y Z, respectivamente. La reunión natural de A y B -

A JOIN B

- Es una relación con la cabecera (X;Y;Z) y un cuerpo formado por el conjunto de todas las tuplas (X: x, Y:y, Z:z) tales que una tupla a aparezca en A con el valor de x en X y el valor y en Y, y una tupla b aparezca en B con el valor y en Y y el valor z en Z.

En la figura II.1.2.7 se presenta un ejemplo de reunión natural (la reunión natural S JOIN P, según el atributo común CIUDAD).

OPERACIONES BOOLEANAS CON VALORES NULOS

Veremos los cuadros de resultados de los operadores lógicos AND, OR y NOT cuando interviene como operando el valor nulo, es decir, un valor desconocido o no aplicable.

S#	SNOMBRE	SITUACION	CIUDAD	P#	PNOMBRE	COLOR	PESO
s1	Salazar	20	Londres	p1	Tuerca	Rojo	12
s1	Salazar	20	Londres	p4	Birlo	Rojo	14
s1	Salazar	20	Londres	p6	Engrane	Rojo	19
s2	Jaimes	10	París	p2	Perno	Verde	17
s2	Jaimes	10	París	p5	Leva	Azul	12
s3	Bernal	30	París	p2	Perno	Verde	17
s3	Bernal	30	París	p5	Leva	Azul	12
s4	Corona	20	Londres	p1	Tuerca	Rojo	12
s4	Corona	20	Londres	p4	Birlo	Rojo	14
s4	Corona	20	Londres	p6	Engrane	Rojo	19

Figura II.1.2.7 La reunión natural S JOIN P

En los cuadros usaremos los símbolos:

V - Verdadero

F - Falso

? - Valor Nulo

AND	V	?	F
V	V	?	F
?	?	?	F
F	F	F	F

OR	V	?	F
V	V	V	V
?	V	?	?
F	V	?	F

NOT	V
V	F
?	?
F	V

Figura II.1.2.8 Operadores lógicos con valores nulo

REGLAS DE CODD

Una vez conocidos los conceptos sobre los que se basa el modelo relacional pueden ser mas entendibles las reglas que dio Codd en 1970 como condicionantes para que un sistema administrador de Base de Datos (DBMS) pueda ser considerado como relacional.

REGLA 0 Un DBMS debe ser capaz de administrar la Base de Datos completamente a través de sus propiedades relacionales.

REGLA 1 Toda la información de la Base de Datos se debe representar como valores en las tablas o relaciones.

REGLA 2 Cada dato de la Base de Datos relacional debe de identificarse totalmente sabiendo el nombre de la tabla, el valor de la clave primaria de esa tabla y el nombre de la columna o atributo.

REGLA 3 El valor nulo representará información desconocida o inaplicable en un dato. El DBMS debe poder tratar este valor con los diferentes operadores lógicos, algebraicos, etc.

REGLA 4 La descripción de la estructura de la Base de Datos (Diccionario) debe

estar contenida en tablas para el uso de un mismo lenguaje por parte del usuario en la manipulación y consulta de la misma.

REGLA 5 El DBMS debe poseer un lenguaje para realizar las siguientes funciones:

- Definición de datos
- Definición de vistas lógicas de usuario
- Manipulación de datos
- Controles de integridad
- Controles de acceso a los datos
- Controles de transacciones lógicas (inicio, final correcto, final erróneo)

REGLA 6 Todas las vistas lógicas actualizables lo serán por el sistema

REGLA 7 La posibilidad de manipulación de una relación con un lenguaje de alto nivel será aplicable a la consulta, inserción, actualización y borrado de datos.

REGLA 8 Los programas de aplicación y los usuarios no deben sufrir variaciones o interferencias por los cambios físicos en el almacenamiento de los datos o en los cambios de acceso a los mismos (independencia física de los datos).

REGLA 9 Los programas de aplicación y los usuarios no deben sufrir variaciones o interferencias por los cambios lógicos .

II.1.3 NORMALIZACION

Codd ha identificado ciertas características estructurales en las relaciones que crean problemas de recuperación y actualización. Estas características indeseables pueden eliminarse descomponiendo una relación en otras de estructuras deseables. Este proceso se conoce como normalización, y puede definirse como un proceso reversible paso a paso para transformar progresivamente una relación no normalizada en relaciones de estructura más sencilla. Como el proceso es reversible, no se pierde ninguna observación durante la transformación. Inicialmente Codd definió tres tipos de propiedades indeseables - agrupaciones de datos, dependencia parcial de clave y dependencia indirecta de clave - que pueden eliminarse en tres etapas de normalización llamadas primera (1FN), segunda(2FN), y tercera(3FN) forma normal. Existe una cuarta (4FN) e incluso una quinta (5FN) forma normal.

Una relación se llama no normalizada cuando contiene agrupaciones de datos. En la primera forma normal las agrupaciones de datos se eliminan descomponiendo la relación no normalizada, si es necesario, en otras varias relaciones. La segunda y tercera forma normales elimina las dependencias parcial e indirecta de los atributos sobre las claves candidatas. En cada etapa de normalización una relación se descompone en otras varias; sin embargo, es posible que una relación esté en tercera o incluso en forma superior al comenzar. Como el proceso de normalización es

sucesivo, una relación en la enésima forma normal está también en la (n-1) - enésima forma normal.

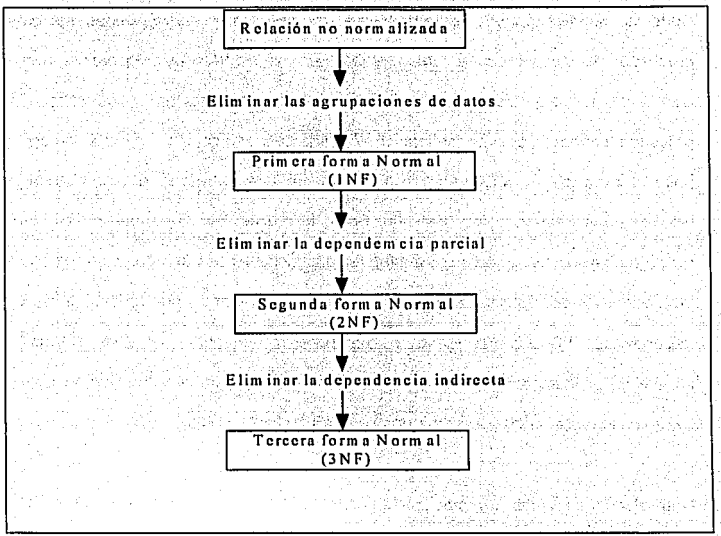


Figura II.1.3.1 Tres niveles de normalización

PRIMERA FORMA NORMAL

La finalidad de la primera forma normal es simplificar la estructura de una relación, asegurando que contenga solamente datos elementales, y no agrupaciones de datos.

De manera más formal

> Una relación esta en primera forma normal si cada uno de sus componentes es atómico.

Un componente es atómico cuando no se puede descomponer en unidades más pequeñas. Como esta es una estructura muy sencilla, se dice que una relación en 1FN .esta normalizada.

Consideremos la relación ANTES de la figura II.1.3.2 es una relación en la cual uno de los dominios subyacentes esta valuado en relaciones. Un dominio valuado en relaciones es un dominio cuyos elementos son en sí relaciones, en vez de escalares simples. En el ejemplo, el atributo PC se define sobre un dominio valuado en relación, cuyos elementos son relaciones binarias; esas relaciones binarias se definen a su vez sobre dos dominios simples, P# y CANT. En una base de datos relacional no se permiten relaciones como ANTES, debe ser sustituida por alguna relación equivalente en lo semántico, pero normalizado. La relación DESPUES de la figura cumple esas condiciones. El grado de la relación DESPUES es 3, y los tres dominios subyacentes son todos simples, como debe ser.

ANTES

S#	PC	
	P#	CANT
S1	P1	300
	P2	200
	P3	400
	P4	200
	P5	100
	P6	100
S2	P1	300
	P2	400
S3	P2	200
S4	P2	200
	P4	300
	P5	400

DESPUES

S#	P#	CANT
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

Figura II.1.3.2 Un ejemplo de primera forma Normal

DEPENDENCIA FUNCIONAL

El atributo Y de una relación A es funcionalmente dependiente del atributo X de A, si en cada instante, cada valor de X está asociado en no más de un valor de Y dentro de una relación A.

Decir que Y es funcionalmente dependiente de X es equivalente a decir que X identifica a Y. En otros términos, si en cualquier instante es conocido el valor de X, el valor de Y queda determinado.

En la base de datos de proveedores y partes, por ejemplo, los atributos SNOMBRE, SITUACION Y CIUDAD de la relación S son todos funcionalmente dependiente del atributo S# de la relación S, porque, dado un valor específico de S.S#, existe solo un valor correspondiente de S.SNOMBRE, de S.SITUACION y de S.CIUDAD.

S.S# ----► S.SNOMBRE

S.S# ----► S.SITUACION

S.S# ----► S.CIUDAD

o, en forma más concisa,

S.S# ----► S.(SNOMBRE, SITUACION, CIUDAD)

Un ejemplo opuesto:

P.COLOR --\-► P.PESO

En la relación P no sucede que para cada color haya un solo peso; por ejemplo, P1 es roja y tiene un peso de 12, P6 también es rojo pero tiene un peso de 19.

Conviene representar las dependencias funcionales en una relación dada mediante un diagrama de dependencias funcionales (Diagrama DF). Los diagramas DF para las relaciones S, P y SP se presentan en la figura II.1.3.3.

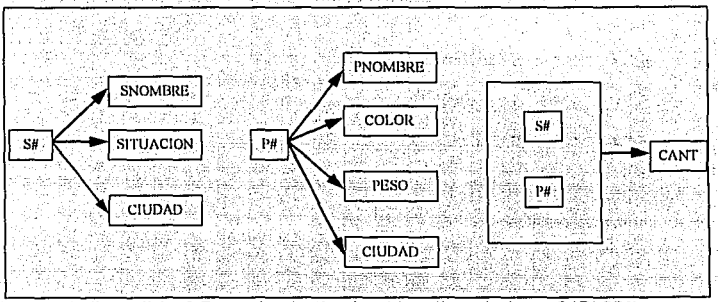


Figura II.1.3.3 Dependencias funcionales en las relaciones S, P, SP.

La dependencia funcional es un concepto semántico. Reconocer las dependencias funcionales es parte del proceso de entender qué significan los datos.

DEPENDENCIA FUNCIONAL COMPLETA

Un atributo o colección de atributos Y, de una relación A es dependiente funcional completo de otra colección de atributos X, de la relación A, si Y es funcionalmente dependiente del total de X pero no de ningún subconjunto de X.

Por ejemplo en la relación SP', de la figura II.1.3.4. el atributo CIUDAD depende funcionalmente del atributo compuesto (S#, P#) :

SP'. (S#,P#) ---→ SP'.CIUDAD

Sin embargo, no es una dependencia funcional completa, porque desde luego tenemos la dependencia funcional:

SP'.S# ------→ SP'.CIUDAD

(CIUDAD también depende funcionalmente de S# solo). Si ya depende funcionalmente de X, pero no por completo, X debe ser compuesto. De aquí en adelante supondremos que "Dependencia funcional" se refiere a una dependencia funcional completa a menos que se diga de manera explícita lo contrario

SP'

S#	CIUDAD	P#	CANT
s1	Londres	p1	300
s1	Londres	p2	200
s1	Londres	p3	400
s1	Londres	p4	200
s1	Londres	p5	100
s1	Londres	p6	100
s2	Paris	p1	300
s2	Paris	p2	400

Figura II.1.3.4 Tabulación SP'

SEGUNDA FORMA NORMAL

Una relación A se halla en la segunda forma normal si esta en la primera forma normal y cada uno de sus atributos no primos es dependiente funcional completo de cada clave candidata de A.

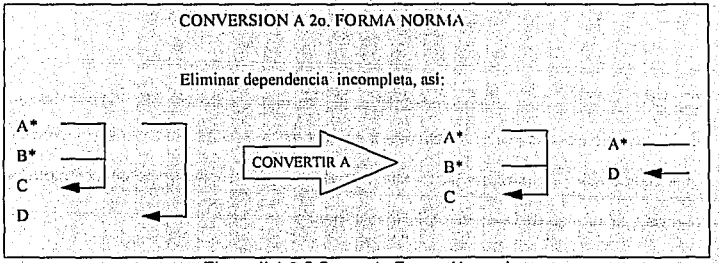


Figura II.1.3.5 Segunda Forma Normal

Las relaciones SEGUNDA y SP de la figura II.1.3.4 están en 2FN, las claves primarias son S#, y la combinación (S#, P#), respectivamente. La relación PRIMERA de la figura II.1.3.6 no esta en 2FN. Cuando una relación esta en 1FN pero no en 2FN, el proceso de reducción consiste en sustituir la relación en 1FN por proyecciones apropiadas; el conjunto de proyecciones así obtenido es equivalente a la relación original, en el sentido de que la relación original siempre se podrá recuperar efectuando la reunión (natural) de esas proyecciones, de manera que no se pierde información con la reducción, en otras palabras el proceso es reversible. En el ejemplo, SEGUNDA y SP

son proyecciones de PRIMERA y PRIMERA es la reunión de SEGUNDA y SP según S#.

PRIMERA

S#	SITUACION	CIUDAD	P#	CANT
s1	20	Londres	p1	300
s1	20	Londres	p2	200
s1	20	Londres	p3	400
s1	20	Londres	p4	200
s1	20	Londres	p5	100
s1	20	Londres	p6	100
s2	10	Paris	p1	300
s2	10	Paris	p2	400
s3	10	Paris	p2	200
s4	20	Londres	p2	200
s4	20	Londres	p4	300
s4	20	Londres	p5	400

SEGUNDA

S#	SITUACION	CIUDAD
s1	20	Londres
s2	10	Paris
s3	10	Paris
s4	20	Londres
s5	30	Atenas

SP

S#	P#	CANT
s1	p1	300
s1	p2	200
s1	p3	400
s1	p4	200
s1	p5	100
s1	p6	100
s2	p1	300
s2	p2	400
s3	p2	200
s4	p2	200
s4	p4	300
s4	p5	400

Figura II.1.3.6 Tabulación de PRIMERA, SEGUNDA, SP

TERCERA FORMA NORMAL

Suelen presentarse algunas anomalías similares a las tres mencionadas anteriormente, aún cuando la relación se halle en la 2FN. Para eliminarlas, se recurre a un último paso de Normalización con el que se va de la segunda a la tercera forma normal. Con este paso se elimina lo que se llama dependencia transitiva. Supongamos que X, Y y Z son tres atributos o tres colecciones de atributos, de una relación A. Si Z es funcionalmente dependiente de Y e Y lo es de X, entonces Z es funcionalmente dependiente de X. Si la correspondencia inversa no es simple, esto es, si X no es funcionalmente dependiente de Y o Y no es funcionalmente dependiente de Z, se dice es transitivamente dependiente de A.

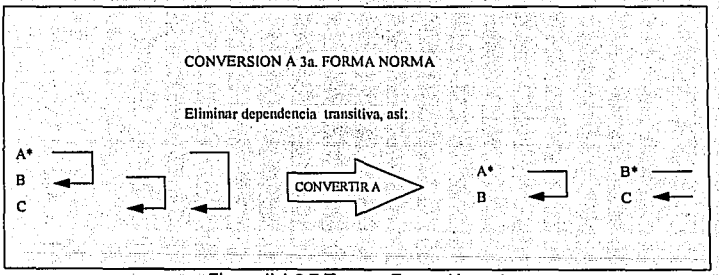


Figura II.1.3.7 Tercera Forma Normal

Las relaciones SC y CS de la figura II.1.3.8 están en 3FN (las claves primarias son S# y CIUDAD, respectivamente). La relación SEGUNDA no esta en 3FN. Cuando una relación esta en 2FN pero no en 3FN siempre podrá reducirse a un conjunto equivalente de relaciones 3FN. Resumiendo, el procedimiento de Normalización es sacar proyecciones para eliminar dependencias transitivas.

SEGUNDA

S#	SITUACION	CIUDAD
s1	20	Londres
s2	10	París
s3	10	París
s4	20	Londres
s5	30	Atenas

SC

S#	CIUDAD
s1	Londres
s2	París
s3	París
s4	Londres
s5	Atenas

CS

CIUDAD	SITUACION
Londres	30
París	10
París	10
Londres	20
Atenas	30

Figura II.1.3.8 Proyección de SEGUNDA en SC y CS

FORMA NORMAL BOYCE/CODD

La definición original de Codd de 3FN tenía ciertas deficiencias. En términos más precisos, no manejaba de manera satisfactoria el caso de una relación en la cual

1. Hay varias claves candidatas,
2. Esas claves candidatas son compuestas, y
3. Las claves candidatas se traslapan (tienen por lo menos un atributo en común)

Así pues, la definición original de 3FN se sustituyo por una definición más sólida, debida a Boyce y Codd la cual atendía también a ese caso, sin embargo esta nueva definición define en realidad una forma normal estrictamente más fuerte que la antigua 3FN por lo que se introdujo un nuevo término para ella, forma normal Boyce/Codd (FNBC).

Para definir FNBC es conveniente introducir un término nuevo; determinante. Se define como un atributo del cual depende funcionalmente por completo algún otro atributo por ejemplo, en la relación PRIMERA de la Figura II.1.3.6, los atributos S#, CIUDAD y (S#, P#) son todos determinantes.

Una relación esta en FNBC si y solo si todo determinante es una clave candidata.

Ahora se habla en términos de llaves candidatas, no sólo de llaves primarias. La motivación para introducir la FNBC estriba en que la definición original de 3FN no maneja satisfactoriamente el caso de una relación que posea dos o más llaves candidatas compuestas y traslapadas. Aunque la FNBC es más (más destructiva) que la 3FN, sigue siendo cierto que cualquier relación se puede descomponer sin pérdidas en un conjunto equivalente de relaciones en FNBC.

CUARTA FORMA NORMAL

Supóngase que se da una relación **no normalizada** que contiene información acerca de cursos, profesores y textos. Cada registro en la relación se compone de un nombre de curso, más un grupo de repetición de nombres de profesores, más un grupo de repetición de nombres de texto. La figura II.1.3.5 muestra esos registros.

El significado de un registro específico en esta relación no normalizada es que el curso indicado lo puede dictar cualquiera de los profesores indicados y en él se utilizan todos los textos indicados.

CPT	CURSO	PROFESOR	TEXTO
	Física	Rosas	Mecánica básica
		Ramos	Principios de óptica
	Matemáticas	Rosas	Mecánica básica
			Análisis vectorial
			Trigonometría

Figura II.1.3.9 Tabulación de muestra de CPT (no normalizada)

Se supone que, para un curso dado, puede existir cualquier número de profesores correspondientes y cualquier número de texto correspondientes; además, se supone -quizá en forma no muy realista- que los profesores y los textos son independientes entre sí (es decir, independientemente de quién enseñe en realidad una instancia particular de un curso dado, se usan los mismos textos). También se supone que un profesor o un texto específico puede estar asociado con cualquier número de cursos. Obsérvese primero que en los datos no existen dependencias funcionales en absoluto.

El significado de la relación normalizada CPT es el siguiente: un tupla <c,p,t,> aparece en CPT y sólo si el curso "c" lo imparte el profesor "p" y usa el texto "t" como

referencia. Para un curso dado, aparecen todas las combinaciones posibles de profesor y texto -es decir, CPT satisface la restricción:

si las dos tuplas < c, p1, t1> , < c, p2, t2> aparecen,

entonces las tuplas < c, p1, t2 >, < c, p2, t1 > también aparecen.

CPT

CURSO	PROFESOR	TEXTO
Física	Rosas	Mecánica básica
Física	Rosas	Principios de óptica
Física	Ramos	Mecánica básica
Física	Ramos	Principios de óptica
Matemáticas	Rosas	Mecánica básica
Matemáticas	Rosas	Análisis vectorial
Matemáticas	Rosas	Trigonometría

Figura II.1.3.10 Tabulación de muestra de CPT (normalizada)

Es claro que la relación CPT contiene muchas redundancias, lo que, ocasiona problemas con las operaciones de actualización; por ejemplo, para adicionar la información de que el curso de física se usa un nuevo texto llamado **Mecánica avanzada,** es necesario crear **tres** tuplas nuevas, una para cada uno de tres profesores. Sin embargo, CPT está en FNBC, por que es <<toda llave>> y no hay otros determinantes funcionales. La existencia de tales relaciones <<problema>> en FNBC ha sido reconocida desde hace algún tiempo. En lo relativo a la relación CPT, resulta claro que las dificultades se deben al hecho de que los profesores y los textos son

independientes entre sí; es también fácil advertir que la situación mejoraría si CPT se reemplazara por sus dos proyecciones CT (CURSO, PROFESOR) y CX(CURSO, TEXTO).

CP

CURSO	PROFESOR
Física	Rosas
Física	Ramos
Matemáticas	Rosas

CT

CURSO	TEXTO
Física	Mecánica básica
Física	Principios de óptica
Matemáticas	Mecánica básica
Matemáticas	Análisis vectorial
Matemáticas	Trigonometría

Figura II.1.3.11 Tabulación de muestra CP y CT.

Véase la figura II.1.3.11 (CT y CX son <<sólo llave>> y, por tanto, ambas están en FNBC). No obstante, ya se ha dicho que la descomposición de la figura II.1.3.9 no se puede hacer sobre la base de las dependencias funcionales.

En cambio, se hace sobre la base de las dependencias funcionales. En cambio se hace sobre la base de un nuevo tipo de dependencia, la dependencia **multivaluada**. Las dependencias multivaluadas DMVs son una generalización de las dependencias funcionales (esto es, una DF es un caso especial de una DMV). Hay dos DMVs en la relación de CTX:

CTX.CURSO ---> ---->CTX. PROFESOR

CTX.CURSO ---> ----> CTX. TEXTO

(La declaración de DMV <<R.A --> --> R.B>> se lee como <<el atributo R.B es **multidependiente** del atributo R.A>>, o, en forma equivalente, <<el atributo R.A. **multidetermina** al atributo R.B>>.)

Ahora se da una definición de DMV.

- Dada una relación R con atributos AB y C, la **dependencia multivaluada**

R.A --> --> R.B

Se cumple en R si y sólo si el conjunto de valores de B que corresponde a un par (valor de A, valor de C) dado en R depende tan sólo del valor de A y es independiente del valor de C). Como siempre, A, B y C pueden ser compuestos.

Nótese que las DMVs, como se han definido, pueden existir solo si la relación R tiene al menos tres atributos.

- Una relación R está en **cuarta forma normal** (4FN) si y sólo si (siempre que exista una DMV en R, por ejemplo A --> -->B) todos los atributos de R también son **funcionalmente** dependientes de A (es decir A --> X para todos los atributos X de R).

En otras palabras, las únicas dependencias (DFs o DMVs) en R son de la forma K ---> X (esto es, una dependencia funcional de una llave candidata K a algún otro atributo X).

Ahora se puede advertir que la relación CPT no está en 4FN, porque corresponde una DMV que no es en absoluto una DF, y mucho menos una DF donde el determinante es una llave candidata. Sin embargo, las dos proyecciones CT y CX están en 4FN. De esta manera, la 4FN es un mejoramiento sobre la FNBC, en el sentido de que elimina otra forma de estructura indeseable.

Fagin demuestra otros dos resultados importantes, que permiten incorporar a la 4FN en el procedimiento global de la normalización :

1. La 4FN es estrictamente más fuerte que la FNBC es decir, cualquier relación en 4FN está por fuerza en FNBC;

2. Cualquier relación puede descomponerse sin pérdida en un conjunto equivalente de relaciones en 4FN.

En otras palabras, la 4FN siempre se puede lograr, aunque en algunos casos puede no ser deseable llevar la descomposición hasta ese punto (o incluso hasta la FNBC).

QUINTA FORMA NORMAL

Hasta ahora se ha supuesto de modo implícito que la sola operación necesaria o utilizable en el proceso de descomposición es el remplazo de una relación por dos de sus proyecciones.

Esta hipótesis ha llevado con éxito a la 4FN. Sorprende descubrir quizá que existen relaciones que no se pueden descomponer sin pérdidas en dos proyecciones, pero que se pueden descomponer sin perdidas en tres (o más). Este fenómeno fue observado por primera vez por Aho, Beeri y Ullman, y también fue estudiado por Nicolas. Considérese la relación SPJ (figura II.1.3.12).

Esta relación es <<toda llave>> y no comprende DFs ni DMVs no triviales y, por tanto, está en 4FN. La figura II.1.3.12 también muestra (a) las tres proyecciones SP, PJ y JS de SPJ, y (b) el efecto de reunir SP y PJ sobre P # y luego reunir el resultado y JS sobre (J#, S#). Obsérvese que el resultado de la primera reunión es generar una copia de la SPJ original más una tupla adicional (espuria), y que el efecto de la segunda reunión es eliminar esa tupla. (El resultado neto es el mismo cualquiera que sea la pareja de proyecciones que se escoja para la primera reunión, aunque el resultado intermedio es diferente en cada caso.)

El ejemplo de la figura II.1.3.12 desde luego se expresa en términos de extensiones. Sin embargo, la <<descomposición en tres >> (para acuñar un término impropio, pero conveniente) de SPJ podría ser una propiedad **intencional** más fundamental -es decir, una propiedad satisfecha por todas las extensiones legales- si la relación satisface cierta restricción independiente del tiempo. Para entender en qué consiste esta restricción obsérvese primero que la afirmación de que SPJ es igual a la reunión de sus tres proyecciones SP, PJ y JS es equivalente a la proposición:

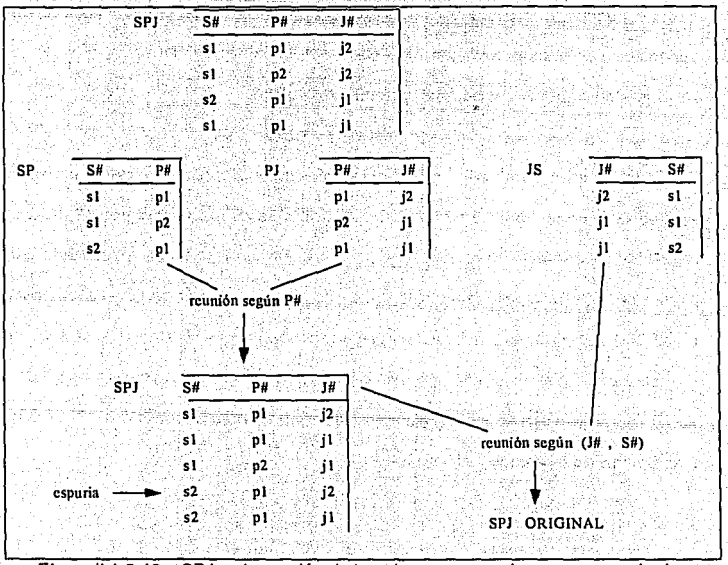


Figura II.1.3.12 SPJ es la reunión de tres de sus proyecciones, pero no de dos cualesquiera

si la pareja <s1,p1> aparece en SP

y la pareja <p1, j1>aparece en PJ

y la pareja <j1, s1> aparece en JS

entonces el triple <s1, p1, j1> aparece en SPJ

(porque el triple <s1, p1, j1> obviamente aparece en la reunión de SP, PJ y JS). Puesto que <s1, p1> aparece en SP si y sólo si s1 y p1 aparecen juntos en SPJ, y lo mismo vale para <p1, j1> y <j1, s1>, se puede reescribir esta última proposición como una **restricción** sobre SPJ:

si <s1, p1, j2 >, <s2, p1, j1 >, <s1, p2, j1 > aparecen en SPJ

entonces <s1, p1, j1 > también aparece en SPJ

Esta es una restricción (aunque más bien rara), tal como lo es en una DF o una DMV. Como se satisface si y sólo si la relación que se estudia es la de algunas de sus proyecciones, tal restricción se llama **dependencia de reunión** (DR). En el ejemplo se dice que SPJ satisface la dependencia de reunión <<*(SP, PJ, JS) >>. En general, la relación R satisface la DR *(X, Y,..., Z) si y sólo si es la reunión de proyecciones sobre X, Y,... , Z, donde X, Y,... , Z son subconjuntos del conjunto de atributo del conjunto de R.

El teorema de Fagin, que dice que R (A, B, C) se puede descomponer sin pérdidas en R1 (A, B) y R2 (A, C) si sólo si A --> --> B/C se cumple en R es equivalente a la afirmación de que R (A, B, C) satisface la DR *(AB, AC) si y sólo si satisface la DMV A --> --> B/C. Como el teorema puede tomarse como una **definición** de DMV se sigue que una DMV es tan sólo un caso especial de DR, o que las DRs son una generalización de las DMVs (así como las DMVs son una generalización de las DFs).

Además, es inmediato la definición que las DRs son la forma **más** general de dependencia posible -es decir, no existe una forma superior de dependencia tal que las DRs sean, a su vez, solamente un caso especial de esa forma superior -en la medida en que se restrinja la atención a las dependencias que tratan de relaciones que se descomponen por medio de la proyección y se reconstruyen por medio de la reunión. (Sin embargo, si se permiten tipos adicionales de operadores en los procesos de descomposición y reconstrucción entonces entrarán en juego los tipos adicionales de dependencia. Se analiza esta posibilidad muy brevemente al final del capítulo).

Regresando al ejemplo, el problema con la relación SPJ es que, aunque está en 4FN, todavía contiene una DR (que no es DF ni DMV). Se ha visto, por tanto, que es posible (y tal vez deseable) descomponer la relación en componentes más pequeños -a saber, en las proyecciones especificadas por la dependencia de reunión-. El proceso de descomposición se puede repetir hasta que todas las proyecciones estén en **quinta forma normal** (5FN).

- Una relación R está en quinta forma normal (5FN) -también llamada forma normal de proyección- reunión (FN/PR) si y sólo si toda dependencia de reunión en R está implicada por las llaves candidatas de R (Se amplía la noción de una DR que está <<implicada por las llaves candidatas >> a continuación.)

La relación SPJ no está en 5FN; si su llave candidata, la combinación S#, P#, J#),en verdad no implica que la relación se pueda descomponer sin pérdidas en sus proyecciones SP, PJ y JS. Las proyecciones SP, PJ y JS están en 5FN por que no contienen ninguna DR en absoluto.

Se observa que, como una DMV es un caso especial de DR, cualquier relación en 5FN también está de manera automática en 4FN. (Fagin muestra que de una DMV implicada por una llave candidata debe en realidad ser una DF donde esa llave es determinante.) Como se indicó antes, cualquier relación se puede descomponer sin pérdidas en un conjunto equivalente de relaciones en 5FN.

Volviendo a hacer referencia a la cuestión de una DR que sea <<implicada>> por las llaves, primero se considera un ejemplo sencillo. La relación proveedor S (S#, NOMS, ESTADO, CIUDAD) con las llaves candidatas S# y NOMS satisface varias dependencias de reunión, por ejemplo la DR:

*((S#, NOMS, ESTADO), (S#, CIUDAD))

Es decir, la relación S es igual a la reunión de sus proyecciones sobre (S#, NOMS, ESTADO) y (S#, CIUDAD). Esta DR está implicada por el hecho de que S# es una llave candidata. La relación S también satisface la DR:

* ((S#, NOMS), (S#, ESTADO), (NOMS, CIUDAD))

Esta DR está implicada por el hecho de que S# y NOMS son llaves candidatas. Fagin da un algoritmo por el cual es posible, dado una DR y un conjunto de llaves candidatas, probar si esa DR está implicada por esas llaves (en general, esto no es inmediatamente obvio -véase el segundo ejemplo anterior-. De esta manera, dado una relación R, se puede decir si R está en 5FN, si se conoce las llaves candidatas y **todas las** DRs en R. No obstante, descubrir todas DRs es de por sí una operación no trivial; es decir, aunque es relativamente fácil hallar las DFs y las DMVs (por que tienen una interpretación directa en el mundo real), no se puede decir lo mismo para una DR que no sea DMV (por que el significado intuitivo de tal DR está lejos de ser directo). De aquí que el proceso de determinar cuándo una relación dada está en 4FN, pero no en 5FN (y, por tanto, tal vez podría descomponerse con ventaja) no está aún claro. Es tentador sugerir que tales relaciones son casos quizá raros en la práctica.

En conclusión, se observa que se sigue de la definición que la 5FN es la **última** forma normal con respecto a la proyección y la reunión. Por que si una relación está en 5FN, las únicas descomposiciones válidas son las que se basan en las llaves candidatas (de manera que cada proyección se componen de una o más llaves candidatas, junto

con cero o más atributos distintos). Por ejemplo, la relación proveedores S están en 5FN. Se puede descomponer sin pérdidas de varias maneras como se vio antes, pero toda proyección aún contendrá al menos una de las dos llaves y por consiguiente, no parece que tal descomposición tenga alguna ventaja.

EJEMPLO DE NORMALIZACION.

Consideremos el caso de una empresa que vende artículos de ferretería, y se cuenta con la siguiente información:

- Nombre y dirección del cliente
- Fecha de la venta
- Vendedor
- Cada uno de los artículos y cantidades vendidas
- Importe unitario de cada artículo vendido
- Subtotal (Cantidad por precio unitario) de cada artículo
- Importe total de la venta

Se debe considerar que:

- Cada cliente puede efectuar varias compras (originar varias facturas)
- Cada compra puede incluir varios artículos
- La información de un cliente (nombre y dirección) raramente cambia

- La información de un artículo raramente cambia

Para registrar la venta se utiliza el nombre y la dirección del cliente, y descripción y precio unitario del artículo. Debemos satisfacer los requisitos de flexibilidad y minimizar la redundancia de datos para reducir el esfuerzo de introducción y el espacio de almacenamiento.

En la **primera forma normal** la idea es eliminar los campos repetidos y crear nuevos archivos o tablas de datos. En nuestro ejemplo cada cliente puede tener varias facturas, por lo tanto crearemos tablas separadas unos para los clientes y otros para las facturas. De la misma forma cada factura puede incluir muchos artículos, por lo tanto, deberemos separar también los artículos en una nueva tabla. Para registrar las facturas habíamos comenzado por esta estructura:

FACTURA

Número de Factura

Fecha de factura

Vendedor

Código del cliente

Nombre del cliente

Dirección del cliente

Código del artículo (1...n)

Descripción del artículo (1...n)

Precio unitario del artículo (1...n)

Cantidad (1...n)

Es importante señalar que la factura puede tener múltiples entradas o grupos repetidos, como el código del artículo o la descripción del artículo (que hemos indicado como la notación 1...n). Si utilizáramos la tabla de datos, tal como aparece arriba, tendríamos que repetir detalles (fecha de factura, vendedor, información del cliente, etc.) en cada uno de los artículo. En su lugar dividiremos los datos en dos tablas separadas, Factura y Detalle-Factura, como aparece abajo. Con esto ahorramos tiempo y espacio en disco y disminuimos la probabilidad de cometer errores . Esta es el primer paso de la normalización.

FACTURA

Número de factura

Fecha de factura

Vendedor

Código del cliente

Nombre del cliente

Dirección del cliente

DETALLE-FACTURA

Número de factura (clave)

Código del artículo (1...n)

Descripción del artículo (1...n)

Precio unitario del artículo

Cantidad

Hay un solo conjunto de datos de tipo FACTURA por cada factura; en cambio, puede haber varios conjuntos de datos de tipo DETALLE-FACTURA por cada factura. Cuando separamos los datos en dos tablas, los elementos que no se repiten quedan en la tabla Factura. Los elementos que sí se repiten pasan a la tabla Detalle-Factura para poder relacionar éste con el registro de la tabla factura pertinente. El número de factura se denomina campo clave.

La clave es un campo o una combinación de campos que identifica unívocamente cada registro. Los números de factura son únicos. Las tablas Factura y Detalle Factura están relacionadas por el campo que tienen en común.

Sin embargo en la tabla Detalle Factura los números de factura no serán únicos si una factura comprende más de un artículo. De hecho es normal que Detalle-Factura contenga más de un registro por cada factura. Decir que el número de factura por sí solo es suficiente como clave para el archivo Detalle-Factura no es correcto.

Para alcanzar **la segunda forma normal,** cada campo de la tabla debe ser funcionalmente dependiente de la clave entera. En nuestro ejemplo no hay necesidad de repetir la descripción y el precio unitario de un artículo cada vez que éste aparece en una factura. La tabla Detalle-Factura necesita un medio de identificar unívocamente sus registros . El numero de factura es insuficiente por sí solo. Sin embargo creamos una clave mediante la combinación de campos. Ni el número de factura ni el código del artículo por separado determinan unívocamente un registro, pero la combinación de ambos sí.

En la segunda forma normal, cada campo debe depender de la clave completa (número de factura y código del artículo). El precio unitario y la descripción del artículo dependen solamente del código del artículo; llamaremos a esta unión dependencia funcional. Ahora debemos de poner los elementos que dependen únicamente del código del artículo en una tabla separada. La consecuencia es que las actualizaciones se facilitan: por ejemplo, un cambio de precio puede hacerse cambiando un solo registro de la nueva tabla Artículo. La separación también elimina la necesidad de escribir la descripción completa y el precio unitario cada vez que queremos incluir un artículo en una factura, ahorrando más tiempo y espacio en disco. Nuestro diseño de tablas en segunda forma normal queda de la forma siguiente:

FACTURA

Número de factura

Fecha de factura

Vendedor

Código del cliente

Nombre del cliente

Dirección del cliente

DETALLE-FACTURA

Número de factura (clave)

Código del artículo (clave)

Cantidad

ARTICULO

Código del artículo (clave)

Descripción del artículo

Precio unitario del artículo

La tabla Factura se relaciona con Detalle-Factura a través del número de factura, mientras que Detalle-Factura se relaciona con Artículo a través del código del artículo.

Para llegar a la **tercera forma normal** cada campo debe ser independiente de cualquier campo no clave. En nuestro ejemplo, el código de cliente depende del número de factura; sin embargo, el nombre y la dirección del cliente dependen tanto

del número de factura como del código del cliente . Por lo cual la información acerca del cliente debe ponerse en otra tabla separada llamada Cliente que se relaciona con Factura vía el código del cliente. Los datos del cliente no dependen de los datos de factura, con lo que el nombre y la dirección del cliente se han de poner en una tabla separada. La ventaja de esta aproximación es que podemos añadir un cliente a nuestra lista antes de emitir ninguna factura a su nombre. El diseño de las tablas en tercera forma normal se muestra en la **figura II.1.3.8** en la cual se muestra la relación entre éstas. Podemos ver que necesitamos al menos cuatro tablas de datos separadas:

Cliente. Información estable sobre clientes

Artículo. Información estable sobre artículos

Factura. Información acerca de la transacción

Detalle-Factura. Cada artículo que aparece en la factura.

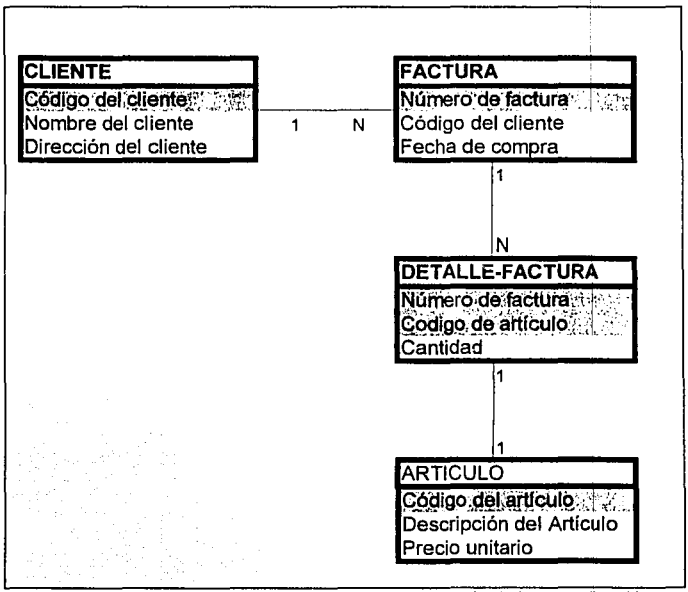


Figura II.1.3.8. Relación entre tablas de ventas después de la normalización

II.1.4 MODELO ENTIDAD - RELACION

A la hora de representar mediante estructuras de datos el mundo real, encontramos bastantes problemas, por las limitaciones y restricciones de las primeras, habrá que tener en cuenta accesos a los datos y cuestiones de eficiencia por lo que el mundo real se ve deformado y difícil de reconocer en el modelo lógico de datos que obtenemos como resultado del diseño.

El modelo ENTIDAD/RELACION surge con la idea de separar el diseño conceptual de datos del diseño físico, es decir, con el fin de plasmar en primer lugar, mediante técnicas de diagramación el mundo real, con sus entidades o agrupaciones de datos con significado dentro del sistema y las relaciones entre ellas y después en un segundo paso, pasar ya al diseño físico del mismo.

No hay que confundir en este modelo el significado de la palabra RELACION con el que tenía en el modelo relacional. Si en éste último RELACION era sinónimo de agrupación de datos, en el modelo ENTIDAD/RELACION su sentido es reflejar la interrelación o asociación entre dos o más entidades del sistema.

El modelo conceptual de datos así conseguido es, por lo general, más estable que el modelo físico y además es independiente del sistema de base de datos elegido.

Por otro lado, a la hora de la confirmación por el usuario de la representación del modelo de datos, el modelo entidad/relación es más fácil de comprender ya que se acerca más a la realidad.

Básicamente en un diagrama de E/R se deben representar los conceptos siguientes:

ENTIDADES Y CONJUNTOS DE ENTIDADES.

Una entidad se compone de una serie de datos que, agrupados tienen cierto significado, es decir, es un objeto que existe y puede distinguirse de otros objetos. Por ejemplo, Juan Pérez, con número de registro federal de causantes PEJJ630410-LK2, es una entidad ya que identifica en forma única una persona específica en el universo. De manera similar, la factura 11296 en la sucursal Centro es una entidad, ya que identifica en forma única una factura determinada. Una entidad puede ser concreta, por ejemplo, una persona o un libro, o abstracta, como un día festivo o un concepto.

Un conjunto de entidades es un grupo de entidades del mismo tipo. El conjunto de todas las personas son clientes de una empresa vendedora de ropa, por ejemplo, puede definirse como el conjunto de entidades cliente. En forma similar, el conjunto de identidades factura podría representar el conjunto de todas las facturas en la empresa.

Una entidad está representada por un conjunto de atributos. Los posibles atributos del conjunto de entidades cliente son nombre, registro-federal, calle y ciudad. Los posibles atributos del conjunto de entidades factura son número y cantidad. Para cada atributo existe un rango de valores permitidos, llamado dominio del atributo. El dominio del atributo nombre podría ser el conjunto de todas las cadenas de texto de cierta longitud. De manera similar, el dominio del atributo número podría ser el conjunto de todos los enteros positivos.

Formalmente, un atributo es una función que mapea un conjunto de entidades a un dominio. Así, cada entidad se describe por medio de un conjunto de parejas (atributo, valor del dato), una pareja para cada atributo del conjunto de entidades. Por tanto, una entidad cliente determinada se escribe por medio del conjunto {(nombre, Juan Pérez), (registro-federal, PEJJ630410-LK2), (calle, Juriquilla), (ciudad, Querétaro)}.

La figura II.1.4.1 muestra una parte de una base de datos que se compone de dos conjuntos de entidades: cliente y factura. Como ejemplo se manejan cinco conjuntos de entidades. Para evitar confusiones se utilizarán nombres únicos para los atributos.

- *sucursal*, conjunto de todas las sucursales de la empresa. Cada una se describe con los atributos nombre-sucursal, ciudad-sucursal y activo.

- *cliente*, el conjunto de todas las personas que tienen una factura en la empresa. Cada uno se describe con los atributos nombre-cliente, registro-federal, calle y ciudad-cliente.
- *empleado*, el conjunto de todas las personas que trabajan en la empresa. Cada uno se describe con los atributos nombre-empleado y teléfono.

NOMBRE	RFC	CALLE	CIUDAD
Juan Perez	PEJJ630410	Juriquilla	Queretaro
Simón Vela	VEHS551010	Tlalpna	D.F.
Sergio Morán	MORS750310	La Viga	D.F.
Arturo Jiménez	JIGA602010	Aeropuerto	D.F.
Carlos Hernandez	HEMC512312	Gustavo Baz	México
Homero Mata	MAMH651020	Rojo Gomez	D.F.
María Hernandez	HEMM762810	Talismán	D.F.

CLIENTE

NUMFACT	CANT
11296	10000
11297	63524
11298	76342
11299	12435
11300	34251
11301	17352
11295	67352
11297	25124
11302	25344
11304	23142
11306	63422
11308	53645
11311	25364
11314	16253
11318	35243

FACTURA

Figura II.1.4.1 Conjunto de entidades cliente y factura

- *factura*, el conjunto de todas las facturas que se mantienen en la empresa. Cada una se describe con los atributos número-factura y cantidad.

- *transacción*, conjunto de todas las transacciones en facturas ejecutadas en la empresa. Cada una se describe con los atributos número-transacción, fecha e importe.

RELACIONES Y CONJUNTOS DE RELACIONES

Constituyen el enlace entre las diferentes entidades del sistema. Por ejemplo, es posible definir una relación que asocia al cliente "Juan Pérez" con la factura 401. Esto especifica que Juan Pérez es un cliente con la factura número 401.

Un conjunto de relaciones es un grupo de relaciones del mismo tipo. Formalmente es una relación matemática de $n \geq 2$ (posiblemente idénticos) conjuntos de entidades. Si E1, E2,..., En son conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de

{(e1, e2,...,en)/ e1 e E1, e2 e E2,..., en e En}

donde (e1, e2,..., en) es una relación.

Para ilustrar esto, considérense los dos conjuntos de entidades cliente y factura de la figura II.1.4.1. Se definirá el conjunto de relaciones ClienFac para denotar la

asociación entre los clientes y las facturas que tienen. Esta asociación se representa en la figura II.1.4.2.

La relación ClienFac es un ejemplo de una relación binaria, es decir, una que implica a dos conjuntos de entidades. La mayor parte de las relaciones en un sistema de base de datos son binarias, pero en ocasiones existen conjuntos de relaciones que incluyen a más de dos conjuntos de entidades. como ejemplo, piense en la relación ternaria (Juan Pérez, 401, Buenavista), que especifica que el cliente Juan Pérez tiene la factura 401 en la sucursal Buenavista. Esta relación es del tipo ternaria CFS (Cliente, Factura y Sucursal), que implica a los conjuntos de entidades cliente, factura y sucursal.

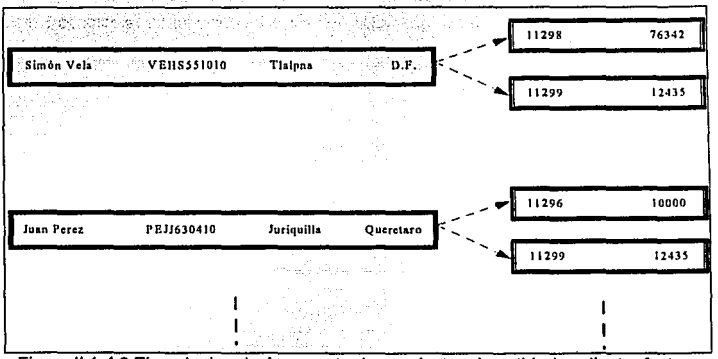


Figura II.1.4.2 Ejemplo de relaciones entre los conjuntos de entidades cliente factura

CARDINALIDAD DE UNA RELACION

Un esquema E-R empresarial puede definir ciertas limitantes con la que deben cumplir los datos contenidos en la base de datos. Una limitante importante es la de la cardinalidad que expresa el número de entidades con las que puede asociarse otra entidad mediante una relación.

Para un conjunto de relaciones R entre los conjuntos de entidades A y B, la cardinalidad debe ser una de las siguientes:

- ***Una a una.*** Una entidad en A está asociada únicamente con una entidad en B, y una entidad en B está asociada sólo con una entidad en A.

- ***Una a muchas.*** Una entidad en A está relacionada con cualquier número de entidades en B, pero una entidad en B puede asociarse únicamente con una entidad en A.

- ***Muchas a una.*** Una entidad en A está vinculada únicamente con una entidad en B, pero una entidad en B está relacionada con cualquier número de entidades en A.

- ***Muchas a muchas.*** Una entidad en A ésta asociada con cualquier número de entidades en B, y una entidad en B está vinculada con cualquier número de entidades en A.

La cardinalidad apropiada para un conjunto de relaciones determinado dependerá, obviamente, del mundo real que el conjunto de relaciones está modelando. Para ilustrar lo anterior, considérese el conjunto de relaciones ClienFac. Si en una empresa una factura puede pertenecer únicamente a un cliente, y un cliente puede tener varias facturas, entonces el conjunto de relaciones es una a muchas de cliente a factura.

Las dependencias de existencia constituyen otra clase importante de limitantes. Específicamente, si la existencia de la entidad x depende de la existencia de la entidad y, entonces se dice que x es dependiente por existencia de y. Funcionalmente, esto quiere decir que si se elimina y, también se eliminará x. Se dice que la entidad y es una entidad dominante y que x es una entidad subordinada.

DIAGRAMA ENTIDAD-RELACION

La estructura lógica general de una base de datos puede expresarse en forma gráfica por medio de un diagrama E-R que se integra con los siguientes componentes:

- Rectángulos, que representan conjuntos de entidades.

- Elipses, que representan atributos.
- Rombos, que representan conjuntos de relaciones.
- Líneas, que conectan los atributos a los conjuntos de entidades, y los conjuntos de entidades a los conjuntos de relaciones.

Cada componente se etiqueta con su nombre correspondiente.

Para ilustrar lo anterior, véase el diagrama de entidades relación de la figura II.1.4.3, que consiste en dos conjuntos de entidades, cliente y factura, vinculados entre sí mediante un conjunto binario de relaciones ClienFac. Los atributos asociados con cliente son números-cliente, registro-federal, calle y ciudad-cliente. Los atributos relacionados con factura son número-factura y cantidad.

El conjunto de relaciones ClienFac puede ser muchas a muchas, una a muchas, muchas a una o una a una. Para distinguir entre éstos, se dibuja una línea con o sin dirección entre el conjunto de relaciones y el de entidades en cuestión.

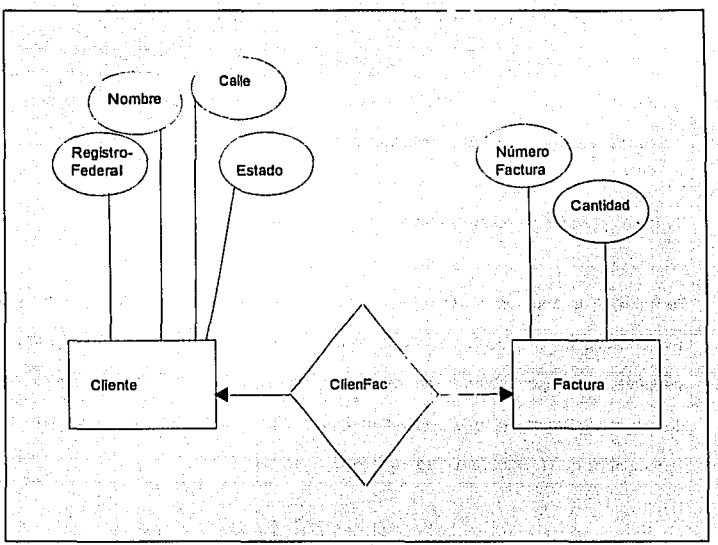


Figura II.1.4.3 Diagrama E-R

Una línea con dirección del conjunto de relaciones ClienFac al conjunto de entidades factura especifica que el conjunto de entidades factura participa, ya sea en una relación una a una o muchas a una con el conjunto de entidades cliente. No puede participar ni en una relación muchas a muchas ni una a muchas con el conjunto de entidades cliente. Una línea sin dirección del conjunto de relaciones ClienFac al conjunto de entidades factura especifica que la entidad factura participa, ya sea en una relación muchas a muchas o muchas a una con el conjunto de entidades cliente.

Volviendo al diagrama E-R de la figura II.1.4.3 puede verse que el conjunto de relaciones ClienFac es muchas a muchas. Si el conjunto de relaciones ClienFac fuera una a muchas, de cliente a factura, entonces la conexión ClienFac tendría una flecha que apuntaría al conjunto de entidades cliente (figura. II.1.4.4. a).

De manera similar, si el conjunto de relaciones ClienFac fuera muchas a una de cliente a factura, entonces la conexión ClienFac tendría una flecha que apuntaría al conjunto de entidades factura (figura II.1.4.4 b). Por lo último, si el conjunto de relaciones ClienFac fuera una a una, entonces la conexión ClienFac tendría dos flechas, una apuntando al conjunto de entidades factura y otra al conjunto de entidades cliente (figura II.1.4. 5).

Los conjuntos de relaciones no binarias pueden especificarse fácilmente en un diagrama E-R. La figura II.1.4.6 consta de tres conjuntos de entidades, cliente, factura y sucursal, relacionados entre sí por medio del conjunto de relaciones CFS.

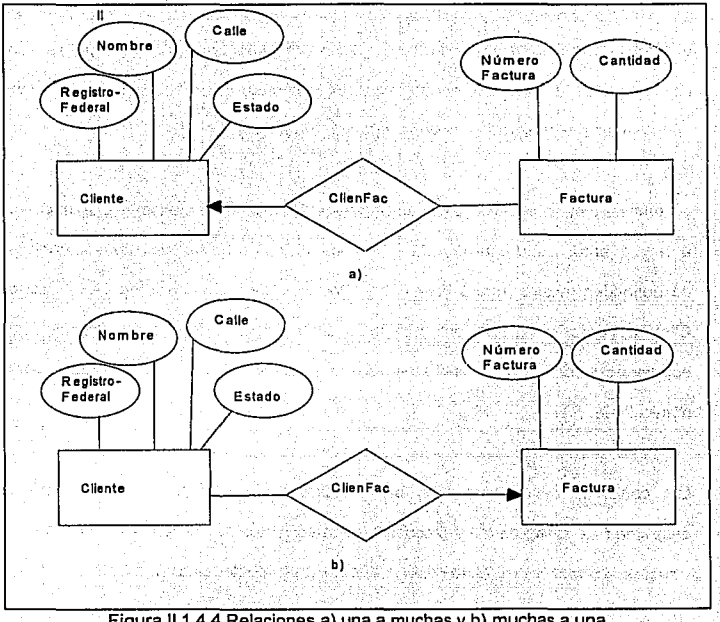


Figura II.1.4.4 Relaciones a) una a muchas y b) muchas a una

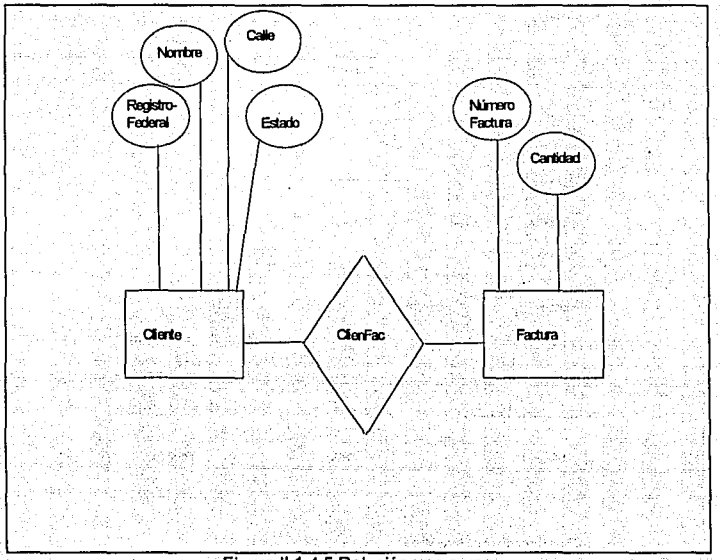


Figura II.1.4.5 Relación una a una

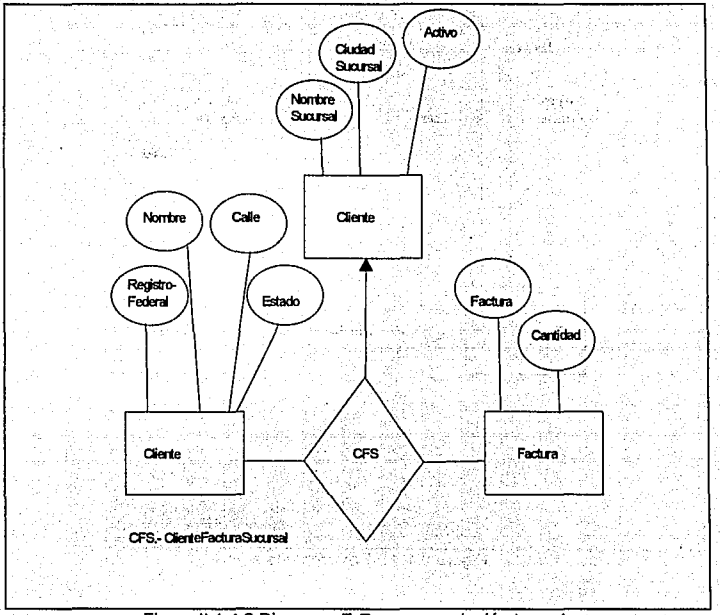


Figura II.1.4.6 Diagrama E-R con una relación ternaria

REDUCCION DE LOS DIAGRAMAS E-R A TABLAS

Una base de datos que se ajuste a un diagrama E-R puede representarse por medio de un conjunto de tablas. Para cada conjunto de entidades y de relaciones en la base de datos, existe una tabla única que recibe el nombre del conjunto de entidades o de

relaciones correspondientes. Cada tabla tiene un número de columnas que, también, tiene nombres únicos.

REPRESENTACION DE CONJUNTOS DE ENTIDADES FUERTES

Sea E un conjunto de entidades fuertes con los atributos descriptivos a1, a2, ... an. Este conjunto de entidades se representa por medio de una tabla denominada E con n columnas diferentes, cada una de las cuales corresponde a uno de los atributos de E. Cada renglón de esta tabla corresponde a una entidad del conjunto de entidades E.

Para ilustrar esto, considérese el conjunto de entidades factura del diagrama E-R. Este conjunto de entidades tiene dos atributos: número factura y cantidad; se representa por medio de la tabla llamada factura, con dos columnas como se muestra en la figura II.1.4.7. El renglón en la tabla factura indica que la factura número 259 tiene un cantidad de $1000. Puede agregarse una entidad nueva a la base de datos insertando un renglón en una tabla. También puede eliminarse o modificarse renglones.

NUMFACT	CANT
11296	10000
11297	63524
11298	76342
11299	12435
11300	34251
11301	17352
11295	67352
11297	25124
11302	25344
11304	23142
11306	63422
11308	53645
11311	25364
11314	16253
11318	35243

FACTURA

Figura II.1.4.7 La tabla factura

En el conjunto de entidades de cliente con los cuatro atributos descriptivos nombre-cliente, registro-federal, calle y ciudad-cliente. La tabla correspondiente a cliente tiene cuatro columnas como se aprecia en la figura II.1.4.8.

NOMBRE	RFC	CALLE	CIUDAD
Juan Perez	PEJJ630410	Juriquilla	Queretaro
Simón Vela	VEHS551010	Tlalpna	D.F.
Sergio Morán	MORS750310	La Viga	D.F.
Arturo Jiménez	JIGA602010	Aeropuerto	D.F.
Carlos Hernandez	HEMC512312	Gustavo Baz	México
Homero Mata	MAMH651020	Rojo Gomez	D.F.
María Hernandez	HEMM762810	Talismán	D.F.

CLIENTE

Figura II.1.4.8 La tabla cliente

REPRESENTACION DE CONJUNTOS DE RELACIONES

Sea R un conjunto de relaciones que implica a los conjuntos de entidades E1, E2,..., En. Sea llave-primaria (E1) el conjunto de atributos que constituye la llave primaria del conjunto de entidades E1. Supóngase que R no tiene atributos descriptivos. Entonces la tabla que corresponde al conjunto de relaciones R tiene el siguiente conjunto de atributos.

$$\bigcup_{i=1}^{n} \text{llave primaria}(E_i)$$

En el caso de que R tenga atributos descriptivos, los cuales serian (a1, a2,..., an), entonces la tabla correspondiente a R tendrá el siguiente conjunto de atributos:

$$\bigcup_{i=1}^{n} \text{llave primaria}\ (Ei) \cup (a1, a2, .., am)$$

Para ejemplificar esto, considere en el conjunto de relaciones ClienFac en el diagrama E-R de la figura II.1.4.5. Este conjunto de relaciones implica a los dos siguientes conjuntos de entidades :

- cliente, cuya llave primaria es registro-federal.
- factura, cuya llave primaria es número-factura.

Puesto que el conjunto de relaciones tiene un atributo descriptivo, fecha, la tabla ClienFac tiene tres columnas tituladas registro-federal, número-factura y fecha, como se muestra en la figura II.1.4.9.

Como ejemplo final, considérese el conjunto ternario de relaciones CFS de la figura II.1.4.10. Esta relación incluye a los tres siguientes conjuntos de entidades:

- cliente, cuya llave primaria es registro-federal
- factura, cuya llave primaria es número-factura

- sucursal, cuya llave primaria es nombre-sucursal

Así, la tabla CFS tiene tres columnas, como se muestra en la figura II.1.4.10.

Registro Federal	Número - Factura	Fecha
PEJJ630410-LK2	11296	20 enero 1994
PEJJ630410-LK2	11297	18 febrero 1994
PEJJ630410-LK2	11298	10 marzo 1994
VEHS551010	11299	23 marzo 1994
MOROS750310	11300	10 abril 1994
MOROS750310	11301	9 mayo 1994
JIGA602010	11295	21 agosto 1994
HEMC512312	11302	28 septiembre 1994
HEMC512312	11304	1 diciembre 1994
MAMH651020	11306	1 enero 1995
MAMH651020	11308	14 febrero 1995
HEMM762810-LKK	11311	21 marzo 1995
HEMM762810-LKK	11314	10 abril 1995
HEMM762810-LKK	11318	17 febrero 1995

Figura II.1.4.9 La tabla ClienFac

Registro Federal	Número - Factura	Sucursal
PEJJ630410-LK2	11296	Querétaro
PEJJ630410-LK2	11297	Tlalpan
PEJJ630410-LK2	11298	Iztacalco
VEHS551010	11299	Coyoacan
MOROS750310	11300	Tlalnepantla
MOROS750310	11301	Naucalpan
JIGA602010	11295	La Villa
HEMC512312	11302	Iztacalco
HEMC512312	11304	Coyoacan
MAMH651020	11306	Tlalnepantla
MAMH651020	11308	Tlalnepantla
HEMM762810-LKK	11311	Naucalpan
HEMM762810-LKK	11314	Miramontes
HEMM762810-LKK	11318	Xochimilco

Figura II.1.4.10 La tabla CFS (Cliente, Factura y Sucursal)

II.2 MODELIZACION DE PROCESOS

El análisis de un sistema tiene como tarea esencial el desarrollo de un modelo antes de la construcción del sistema mismo. Elaborar un modelo permite conocer, resaltar y descartar los elementos que intervienen en el análisis, diseño e implementación de un sistema. Los analistas de sistemas hacen modelos en papel del sistema a realizar, esto es, representaciones abstractas de lo que será después una combinación de hardware y software de la computadora.

En el análisis de sistemas se usan herramientas de modelado para:

- Enfocar características importantes.
- Discutir cambios y correcciones a los requerimientos del usuario con bajo costo y mínimo riesgo.
- Verificar que el analista de sistemas entienda correctamente los requerimientos del usuario y que tenga la documentación en que se basen el diseñador del sistema y los programadores para construir el sistema.

La elección de una herramienta de modelado esta sujeta a la experiencia o preferencia del analizador, la complejidad del sistema o la imposición de un estándar. Algunos sistemas requieren de la combinación de varias herramientas de modelado, cada una de ellas se limita a enfocar un aspecto concreto del sistema, como lo puede ser las estructuras de datos, el comportamiento en el tiempo, el diccionario de datos, las especificaciones de los procesos (seudocódigos), etc.

Cualquier herramienta de modelado deberá tener las siguientes características:

- Debe ser gráfica.
- Debe permitir observar al sistema en forma top-down, en forma particionada.
- Debe ser mínimamente redundante.
- Debe auxiliar al lector a predecir el comportamiento del sistema.
- Debe ser transparente para el lector.

Existen distintas herramientas para el modelado de sistemas como lo son diagramas de flujo de datos, diagramas entidad-relación (sección II.1.4), diagramas de flujo,

diagramas HIPO, tablas de decisión, diagramas de estado de transición, diagramas Pert, entre otros.

Estas herramientas se conforman de gráficos y textos descriptivos. Los gráficos proveen una forma fácil de mostrar al usuario la mayoría de los componentes del modelo así como las conexiones o interfaces entre los componentes. El texto proporciona una definición precisa de los principales componentes y sus conexiones. Por ejemplo, en el caso de un diagrama de flujo, se tiene que a través de él es posible representar gráficamente la lógica de procedimiento de un programa de computadora. Como lo muestra la figura II.2.1, este tiene básicamente tres componentes:

- Un cuadro que representa una instrucción ejecutable o una secuencia contigua de instrucciones de computadora.
- El rombo representa una decisión; en el caso sencillo, representa una decisión binaria.
- Flechas, las cuales, conectan a los cuadros y al mismo tiempo representan el flujo del control.

En la figura II.2.2 se muestra un diagrama Pert típico para un proyecto imaginario. Cada rectángulo representa una tarea o actividad, es decir, un fragmento reconocible

de trabajo que debe hacerse. Los cuadros con esquinas redondeadas se conocen como señalamientos y tienen un significado obvio dentro del contexto de un proyecto típico. Las líneas que conectan los cuadros muestran dependencias, es decir, muestran qué actividades deben terminarse antes de comenzar otra. Las líneas más gruesas y obscuras que forman un camino contiguo del principio al final del proyecto representan el camino crítico, es decir, aquellas actividades cuyo retraso obligaría al retraso del proyecto global. Se considera que las actividades que no están en el camino crítico disponen de un tiempo más holgado.

Un DFD (diagrama de flujo de datos) es una de las herramientas de modelado que más se utilizan, especialmente en sistemas en donde las funciones del sistema son más complejas e importantes que los datos que se manipulan. Los diagramas de flujo de datos fueron utilizados por primera vez en la ingeniería de software como una notación para estudiar diferentes tópicos del diseño de sistemas en libros y artículos.

Parte de su notación fue prestada por elementos de la teoría de gráficas, y esta continua siendo usada por ingenieros en software como un medio de representación de los requerimientos de sus usuarios.

Un DFD ilustra los procesos que el sistema realiza con los datos. Por tanto, procesos y datos son los elementos esenciales en un DFD, para incluirlos en éste se debe pensar en lo siguiente:

- ¿Cuáles son los procesos que el sistema desarrolla?.

- ¿Cuáles son sus entradas y cuáles son sus salidas?.

- ¿Cuáles son los tipo de procesos que el sistema realiza?.

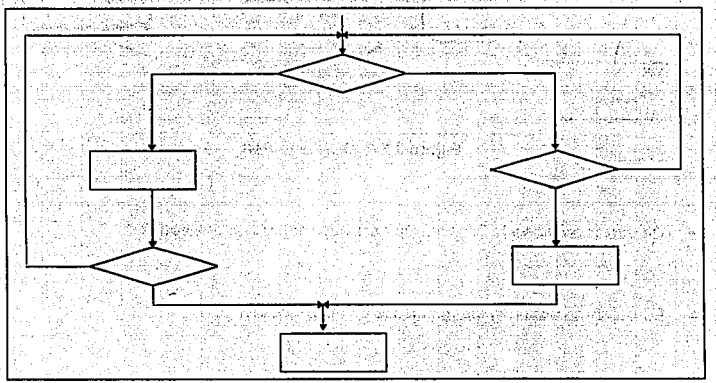


Figura II.2.1 Diagrama de Flujo.

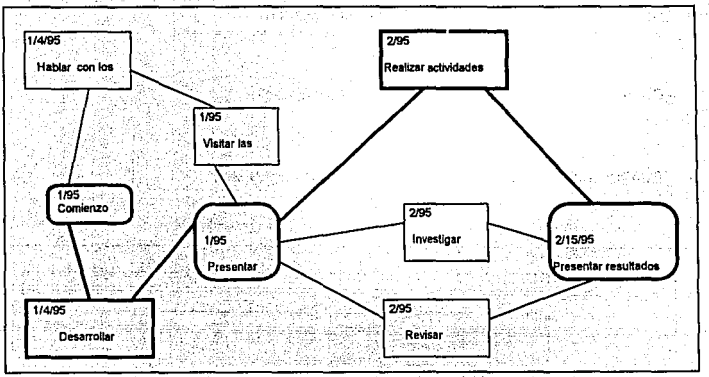


Figura II.2.2 Diagrama Pert.

- ¿De donde proviene la información para que realice sus procesos?.

- ¿En donde se verifican los resultados de las procesos?

Un DFD proporciona una vista amplia y clara de los componentes funcionales de un sistema, pero ningún detalle sobre ellos. Dice muy poco acerca de los detalles de los datos y no indica las relaciones existentes entre ellos. Aún así, en un DFD puede observarse el control de los flujos de datos, el control sobre los procesos y el control sobre fuentes de datos.

Sistemas complejos son modelados con más de un DFD, que pueden ser docenas o miles, los cuales se ordenan en niveles jerárquicos. Además existen convenciones, reglas y formatos para etiquetar y enumerar los detalles en un DFD y reglas que permiten distinguir DFD eficientes. Tales reglas requieren un trato detallado y cuidadoso para crear diagramas que aseguren una completa consistencia de la representación de los requerimientos que se tienen en un sistema.

Esta sección explica los diagramas de flujo de datos examinando:

- sus componentes
- su notación para sistemas de tiempo-real (control de flujos y control de procesos)
- lineamientos que eviten confusiones, errores e inconsistencias en su elaboración
- DFDs multinivel para modelado de sistemas complejos

Un DFD permite observar a un sistema como una red de procesos, conectados unos a otros por "pipelines" (vías de información) y "holding tanks of data" (tanques contenedores de datos).

Los elementos que gráficamente son representados en un DFD son:

- ***Procesos.*** Se muestran con círculos o "burbujas". Representan distintas funciones individuales que el sistema ejecuta. Las funciones transforman las entradas en salidas.

- ***Flujos de datos.*** Se muestran con líneas curvas o rectas dirigidas por una flecha. Estas indican las conexiones entre los procesos (funciones del sistema), y representan la información que el proceso requiere como entrada y/o la información generada como salida.

- ***Almacenes de datos.*** Se muestran con dos líneas paralelas o por una elipse. Muestran una colección de datos que el sistema debe retener por un periodo de tiempo como lo son archivos y bases de datos.

- ***Terminales.*** Muestran entidades externas que se comunican con el sistema. Terminales comunes son personas, grupos de personas (departamentos o divisiones de una empresa), una computadora externa u organizaciones externas.

La figura II.2.3 muestra un típico DFD para un sistema pequeño. Al examinar sus componentes en detalle es posible deducir lo siguiente:

- ☑ Apenas requiere de una breve explicación; uno puede observar el diagrama y lograr entenderlo. La notación es simple en un sentido intuitivamente obvio.
- ☑ El diagrama solo ocupa una página. Esto significa dos cosas: 1) El lector lo observa sin divagaciones y, 2) El diagrama modela a un sistema que no es complejo.
- ☑ El diagrama ha sido dibujado por una computadora, lo que permite mantener la consistencia gráfica de los elementos del diagrama.

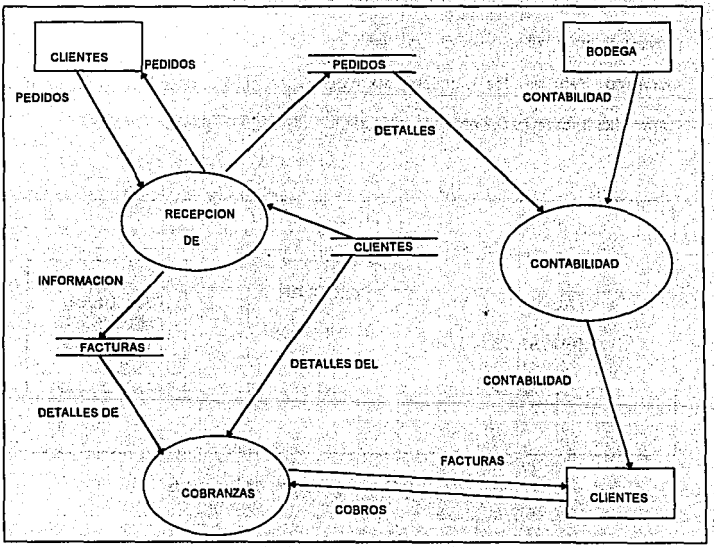


Figura II.2.3 Un típico Diagrama de Flujo de Datos.

PROCESOS

Además de poder indicarse por círculos también es posible representarlos con óvalos, con rectángulos con esquinas redondeadas o con simples rectángulos. En la figura II.2.4 se observa que el nombre del proceso se indica con una palabra, frase, o sentencia. Es recomendable que en cada uno de estos casos indiquen el nombre del proceso utilizando un verbo.

Aún así en algunos casos los procesos contienen nombres de personas o grupos (departamentos, divisiones de una empresa, etc., una computadora o un dispositivo mecánico. Esto es, los procesos en algunas ocasiones describen quienes o que cumple las funciones del proceso, en lugar de describir lo que es el proceso.

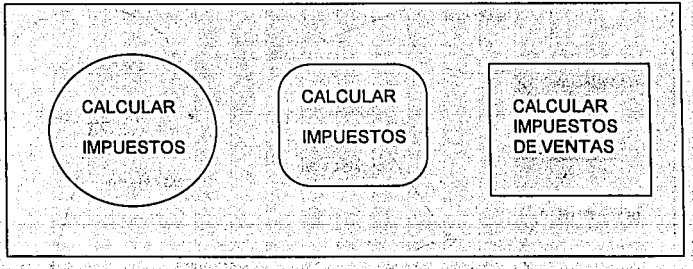


Figura II.2.4 Representación de Procesos.

FLUJO DE DATOS

Un flujo de datos gráficamente es representado por una flecha que sale o llega a un proceso.

El flujo de datos es usado para describir el movimiento de paquetes de información de una parte del sistema a otra. Así las flechas en un diagrama de flujo de datos representan la información que es manipulada por los procesos la cual también puede ser parte integral de la computadora como lo son bits, caracteres, mensajes, números de punto flotante y otros, sino también otro tipos de objetos físicos como se indica en la figura II.2.5.

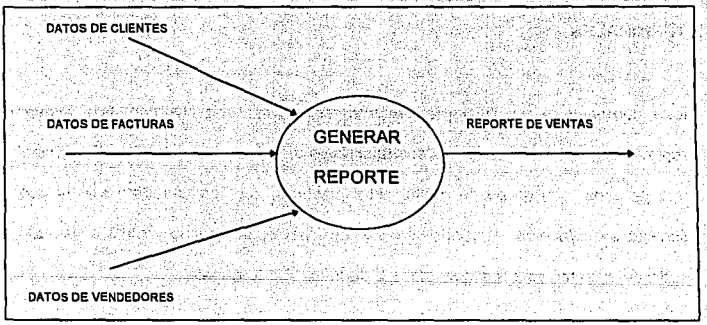


Figura II.2.5. Representación de Flujo de Datos.

Una flecha de flujo de datos puede converger o divergir (figura II.2.6). Conceptualmente esto significa que una flecha se puede fraccionar en varias flechas así como varias flechas se pueden unir en una misma. En el caso de una flecha divergente se entiende que copias de los datos se dirigen a diferentes partes del sistema y en el caso convergente se entiende que un paquete de datos se puede construir por la unión de otros más elementales.

ALMACENES DE DATOS

Los almacenes de datos se utilizan para modelar colecciones de datos. La notación que se utiliza son dos líneas paralelas, u otras alternativas en las figuras siguientes.

TERMINALES

Este componente del DFD gráficamente representa un rectángulo. Los terminales representan entidades externas con las cuales el sistema se comunica. El terminal es una persona , un grupo de gente, otro sistema, otra computadora con la cual el sistema se comunica. Algunas veces el terminal es el usuario y otras el usuario es parte del sistema y es quien ayudará a identificar los terminales.

REGLAS PARA LA CONSTRUCCION DEL DFD

Existen un gran número de reglas para la construcción de un DFD las reglas incluyen lo siguiente:

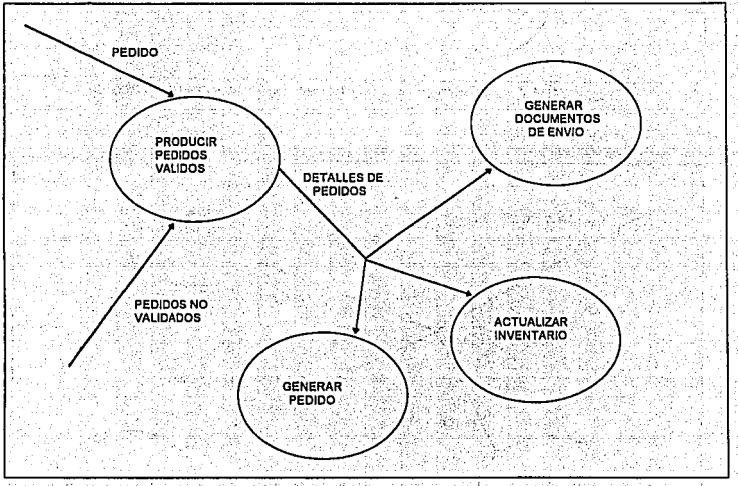


Figura II.2.6 DFD divergente.

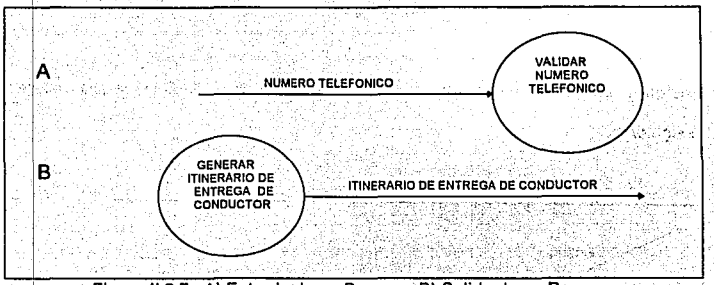


Figura II.2.7. A) Entrada de un Proceso B) Salida de un Proceso

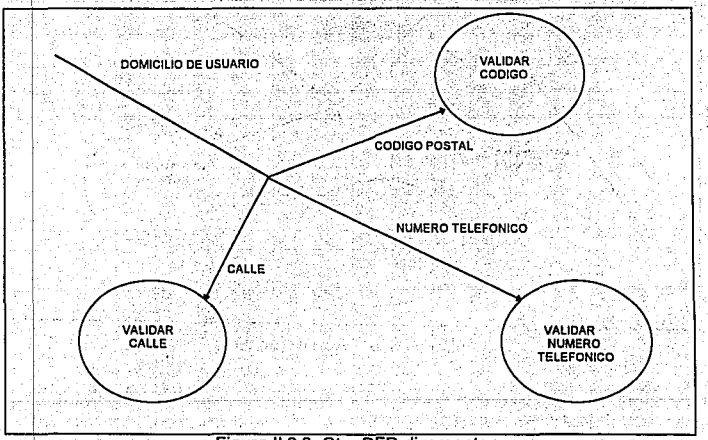


Figura II.2.8. Otro DFD divergente.

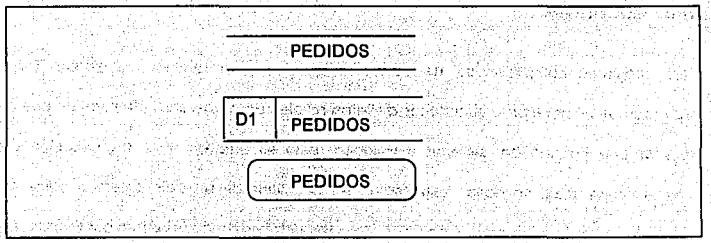


Figura II.2.9 Representación de Almacenes de Datos.

- Escoger los nombre para procesos, flujos , almacenamientos y terminales.
- Número de procesos.
- Redibujar varias veces el DFD por estética.
- Evitar hacer DFDs complejos.
- Estandarizar el DFD a otro DFDs.

DFDs MULTINIVEL

Para sistemas complejos es necesario utilizar un gran número de DFDs. Para organizarlos es necesario establecer distintos niveles para que en cada nivel se pueda detallar una porción del sistema a modelar, esto es análogo a la organización de mapas en un atlas, en donde esperamos ver una vista de un país entero o quizá de todo el mundo; y otros mapas nos muestran los detalles de cada país o continente. La siguiente figura muestra una organización típica de un DFD multinivel.

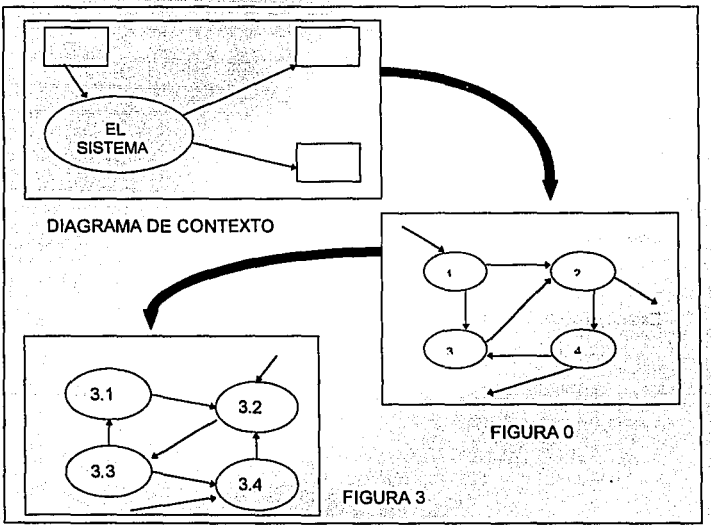


Figura II.2.10. DFD multinivel.

II.3. CONCEPTOS DE INGENIERIA DE SOFTWARE

Para entender los conceptos de ingeniería de software, empezaremos por entender lo que es **el concepto del ciclo de vida de un proyecto**.

Las pequeñas empresas tienden a ser relativamente informales; los proyectos de desarrollo de sistemas surgen como resultado de una plática entre el usuario y el gerente de proyecto (o el analista de sistemas, el programador, el operador de la computadora, o hasta el portero!), y el proyecto procede desde el análisis del sistema hasta el diseño e implementación sin el menor problema.

Por otra parte, en las grandes organizaciones, las cosas son hechas sobre bases mucho más formales. Las diferentes comunicaciones entre usuarios, gerentes, y el equipo del proyecto tienden a ser documentadas de manera escrita; y cada uno entiende que el proyecto irá a través de varias fases antes de que sea terminado. Aún así, es sorprendente ver grandes diferencias, entre dos gerentes, en la forma de conducir sus propios proyectos dentro de la misma organización.

Sin embargo, recientemente la manera de considerar el desarrollo de sistemas ha comenzado a cambiar. Más y más, grandes y pequeñas organizaciones están adoptando un solo y uniforme ciclo de vida en los proyectos -algunas veces conocido como **plan del proyecto** o **metodología de desarrollo de sistemas** o, simplemente, **"la forma en que se hacen las cosas en este lugar"**. Usualmente contenido en un cuaderno tan pesado como los manuales de estándares que se encuentran (sin leer) en todos los escritorios de los analistas y programadores, el ciclo de vida del proyecto proporciona una forma común para todos y cada uno de los elementos del departamento de desarrollo de sistemas. El propósito de tener definido un ciclo de vida del proyecto, son los siguientes tres objetivos :

- Definir las actividades que se llevarán a cabo durante el proyecto de desarrollo del sistema.
- Introducir consistencia entre los diferentes proyectos de desarrollo de sistemas en la misma organización.
- Proporcionar puntos de revisión que permitan un control administrativo para la toma de decisiones.

El primer objetivo es particularmente importante en una gran organización en la que gente nueva está constantemente agregándose a los proyectos. El gerente de proyecto novato, puede no entender o pasar por alto el significado de la importancia de

las fases de un proyecto, cuando solamente sigue su intuición. Naturalmente, puede ocurrir que los programadores y los analistas de sistemas no entiendan, en donde y como, canalizar sus esfuerzos en todo el contexto del proyecto a menos que ellos hayan sido enterados con una apropiada descripción de todas las fases del proyecto.

El segundo objetivo es también importante en las grandes empresas. Para los altos niveles de gerencia, puede ser extremadamente desconcertante revisar un ciento de proyectos diferentes, donde cada uno de ellos es llevado de una manera diferente.

El tercer objetivo del ciclo de vida del proyecto, permite a los gerentes controlar el proyecto; esto es, en proyectos triviales se revisa solamente al final del proyecto con una pregunta ¿Fue finalizado el proyecto a tiempo y dentro de lo especificado? o aún más simple ¿Fue finalizado del todo? y ¿Cumple con los requerimientos del usuario?. Sin embargo, para grandes proyectos, los gerentes y encargados del proyecto podrían tener puntos intermedios de revisión, los cuales permitan determinar cuanto del proyecto está cumpliendo con lo planeado, y los recursos necesarios para alcanzarlo. Además, un usuario podría revisar algunos estados o fases del proyecto para determinar si se cumple con lo que el quiere, y así continuar con el avance del proyecto.

De lo anterior, podemos ver que el ciclo de vida de un proyecto definitivamente no es una carga del proyecto mismo. Y que éste no libera al gerente del proyecto de la difícil responsabilidad de tomar decisiones, revisar alternativas, enfrentar batallas de políticas, negociar con usuarios, impulsar la moral de los programadores, o cualquiera de los elementos relacionados con el proyecto. El ciclo de vida del proyecto solamente ayudará a organizar las actividades del gerente.

II.3.1. TECNICA BOTTOM-UP

Existe el ciclo de vida convencional o clásico del proyecto de sistemas, el cual se muestra en la figura II.3.1.1. Todos los proyectos van a través de una clase de análisis de sistemas, diseño e implementación, aún si este no se hace exactamente como se muestra en el diagrama.

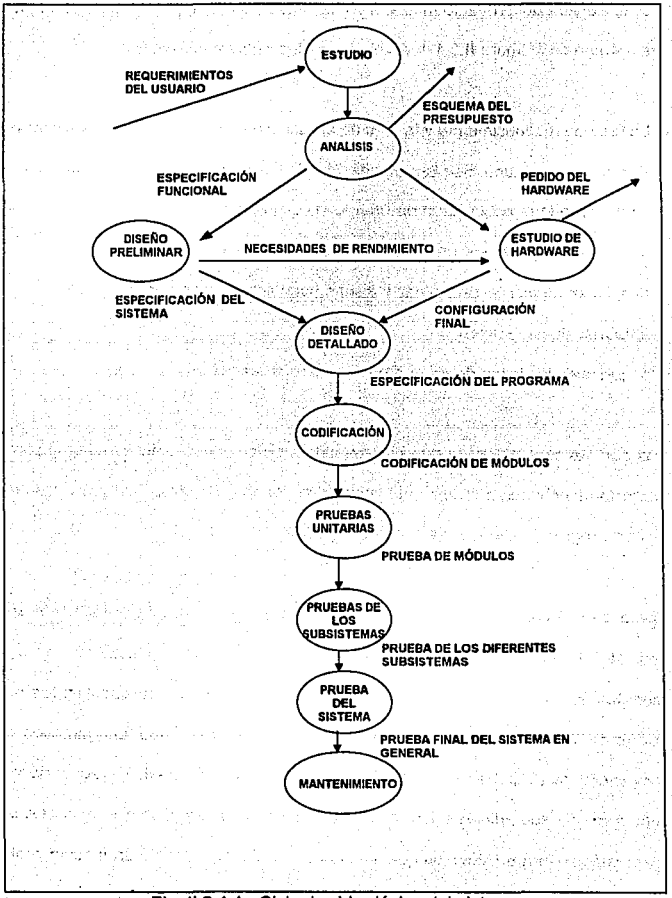


Fig. II.3.1.1 Ciclo de vida clásico del sistema

El ciclo de vida del proyecto utilizado en las organizaciones, puede diferir de alguna u otra manera de la figura II.3.1.1. en alguna de las siguientes formas:

- La fase de reconocimiento y la fase de análisis pueden ser unidas en una sola fase.
- Puede no existir una fase llamada estudio de hardware, si en el proyecto no se está considerando la adquisición de un nuevo equipo, sino al contrario, la utilización del mismo, pero sin causar un impacto operacional mayor.
- Las fases de diseño preliminar y diseño detallado pueden ser consideradas en una sola fase llamada diseño.
- Algunas de las fases de prueba pueden ser agrupadas en una sola.

Dos son las características que determinan un ciclo de vida clásico del proyecto: una fuerte tendencia a seguir una implementación del sistema bottom-up y una insistencia en una progresión lineal entre una fase y la otra.

Como su nombre implica, la técnica **bottom-up** (Metodología de abajo hacia arriba), consiste en la construcción, integración y prueba de programas, comenzando con los módulos más bajos llamados módulos atómicos (Esto es, los módulos en los niveles inferiores en la estructura de un programa). Debido a que los módulos están integrados de abajo hacia arriba, el procesamiento de información, requerido por los módulos que subordinan un nivel dado, siempre está disponible y la necesidad de crear módulos **stub** (Módulos que simulan el comportamiento de un módulo real, esto

es, que simplemente envían los datos que el módulo del siguiente nivel requiere) es eliminada.

Una estrategia bottom-up puede ser implementada con los siguientes pasos:

- Los módulos de más bajo nivel son combinados en **clusters** (grupos) que realizan una sub-función específica de software.
- Un **driver** (programa de control para pruebas) es escrito para coordinar las pruebas.
- El cluster es probado.
- Los drivers son eliminados y los cluster son combinados con los siguientes niveles en la estructura del programa.

Las pruebas de integración siguen el patrón mostrado en la figura II.3.1.2. Los módulos son combinados para formar los cluster 1, 2 y 3. Cada uno de los cluster es probado utilizando un driver (Mostrado como un bloque punteado). Los módulos en los cluster 1 y 2 son subordinados al módulo Ma. Los drivers D1 y D2 son eliminados y los cluster son interconectados directamente a Ma. De manera similar, el driver D3 para el cluster 3 es removido para la integración con el módulo Mb. Tanto Ma y Mb a su vez serán integrados con el módulo Mc, y así sucesivamente. Diferentes categorías de drivers son ilustrados en la figura II.3.1.3.

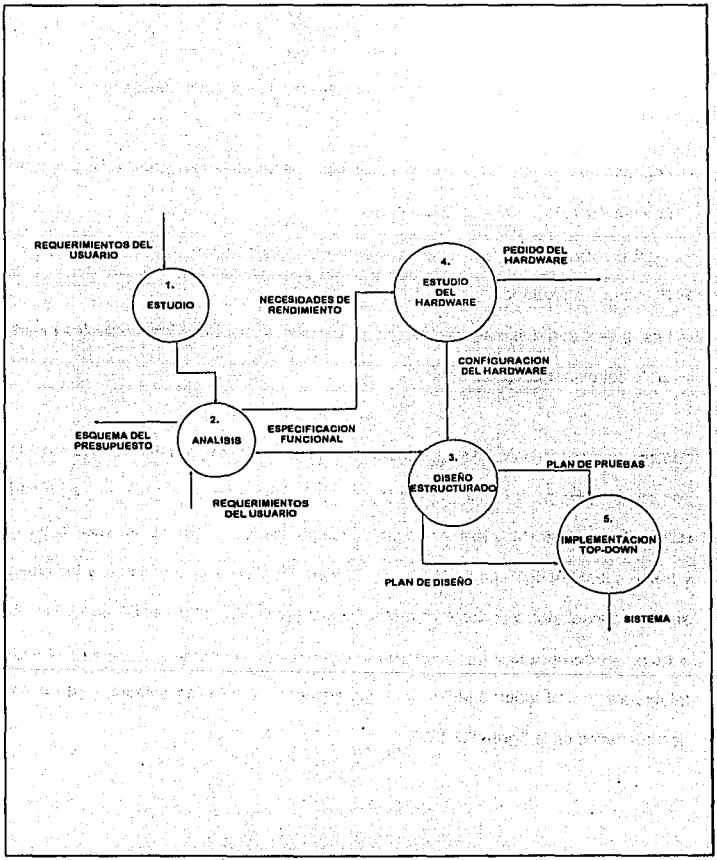


Figura II.3.1.2 Integración bottom-up.

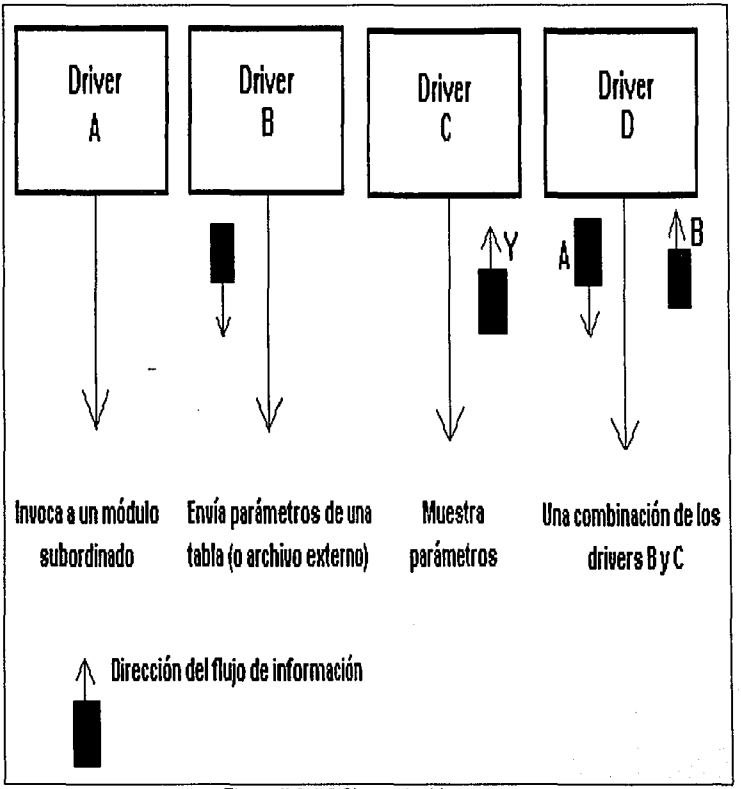


Figura II.3.1.3 Clases de drivers.

Como la integración se mueve hacia arriba, la necesidad de drivers de prueba separados desaparece. De hecho, si los dos niveles más altos de la estructura de un programa son analizados por medio de un método **top-down** (Metodología de arriba hacia abajo), el número de drivers puede ser reducido substancialmente y la integración de clusters es simplificado de una manera muy significativa.

El uso de una implementación bottom-up es una de las principales debilidades en el ciclo de vida clásico del proyecto. Como se pudo ver en la figura II.3.1.1, los programadores tienen que esperar a probar todos sus módulos primero, siguiendo la prueba del subsistema, y finalmente la prueba del sistema. Este procedimiento es también conocido en la industria de cómputo como el "ciclo de vida de cascada", que se muestra en la figura II.3.1.4

No es claro donde surgió este procedimiento, pero pudo haber surgido de las líneas de ensamble de la industria manufacturera. La implementación bottom-up es una buena técnica en una línea de ensamble de automóviles, pero solamente después de que el modelo prototipo ha sido completamente depurado. Desafortunadamente, muchas organizaciones que desarrollan sistemas, están tranquilamente produciendo algún sistema de esta categoría.

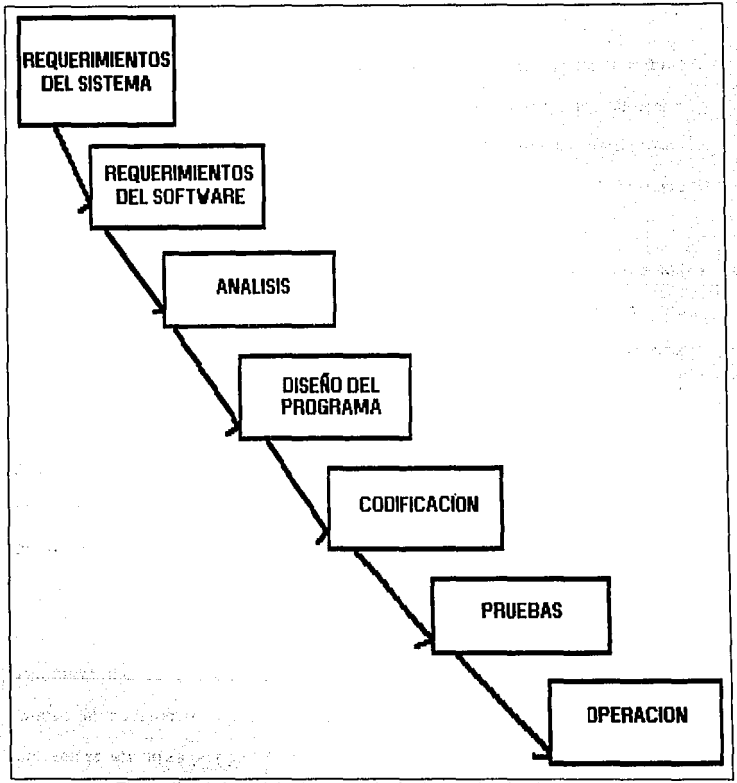


Figura II.3.1.4 Modelo de cascada de desarrollo de sistemas.

El método bottom-up tiene un número de dificultades serias:

- Nada está terminado hasta que todo está hecho. Esto es, si el proyecto falla justo en medio de una prueba del sistema, no habrá nada que mostrar al usuario, excepto una enorme pila de listados de programas que no tienen ningún valor para el usuario.

- Los errores más triviales son encontrados al comienzo del período de prueba, y los errores más serios son encontrados al final. Las pruebas de módulos no cubren errores lógicos relativamente simples dentro de los módulos individuales; la prueba del sistema, por otro lado, no cubre la mayoría de los errores de interface entre los subsistemas. El punto es, que esos errores principales de interface, no desea el programador encontrarlos al final del desarrollo del proyecto; tales errores pueden significar el recorrido de un gran número de módulos, y puede tener un devastador impacto en el plan de trabajo justo en el momento en que todos se encuentran cansados después de haber trabajado duramente durante meses.

- El **debugging** (nombre que se le da a la acción de buscar y encontrara un error dado) tiende a ser extremadamente difícil durante las últimas pruebas del sistema. Note que se distingue entre pruebas y debugging. Debugging es el arte de descubrir donde se encuentra el error (y la subsecuente determinación de como corregirlo)

por su parte el proceso de testing determinará que existe un error. Cuando un error es encontrado durante la fase de prueba del sistema del proyecto bottom-up, este es extremadamente difícil de decir en que módulo ha ocurrido el error; este podría estar en cualquiera de los cientos (o miles) de módulos que han sido combinados por primera vez. La búsqueda se puede representar como la búsqueda de una aguja en un pajar.

- Los requerimientos de tiempo de cómputo para las pruebas, usualmente crecen exponencialmente durante los estados finales de la prueba. Más específicamente, el gerente del proyecto frecuentemente encuentra que necesitará grandes cantidades de espacios continuos de tiempo de cómputo para la prueba del sistema, estas podrían ser 12 horas ininterrumpidas de tiempo de cómputo por día.

II.3.2. TECNICA TOP-DOWN

A finales de los 70´s y principios de los 80´s ha habido un gran reconocimiento a las técnicas como el diseño estructurado, la programación estructurada, y la implementación top-down que podría ser reconocida oficialmente como parte del ciclo de vida del proyecto. Este reconocimiento ha llevado al diagrama de un nuevo ciclo de vida del proyecto, representado en la figura II.3.2.1; el cual muestra dos capacidades obvias que no se presentaban en el ciclo de vida clásico:

- La secuencia de codificación bottom-up, la prueba de módulos, y la prueba del sistema son reemplazados por una implementación top-down, una técnica donde los módulos de más alto nivel son codificados y probados primero, seguidos por los módulos del siguiente nivel inferior, módulos de detalle. Existe también una fuerte tendencia a utilizar la programación estructurada como el método actual de codificación del sistema.

- El diseño clásico es reemplazado por un diseño estructurado.

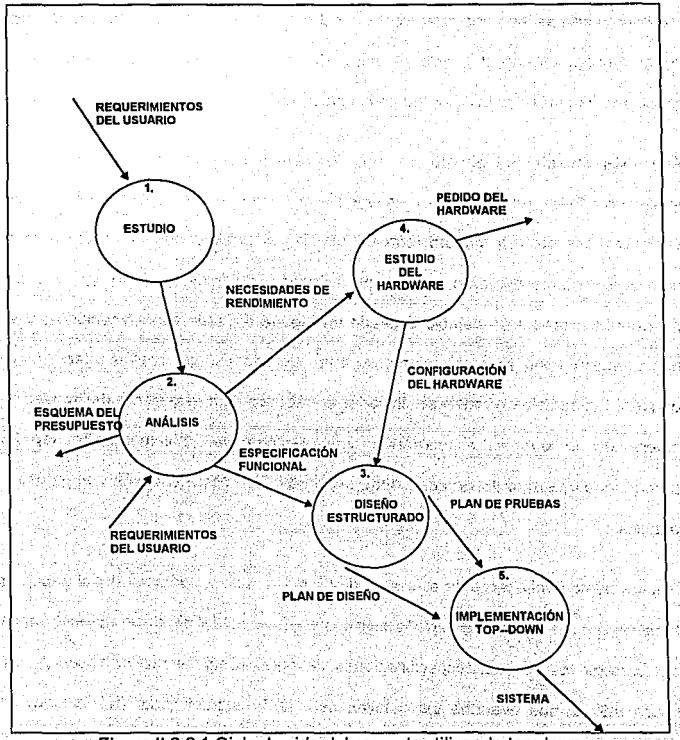


Figura II.3.2.1 Ciclo de vida del proyecto utilizando top-down

Además de esas diferencias obvias, hay algunos puntos a considerar de este ciclo de vida modificado. Considérese, por ejemplo, que la implementación top-down permite que alguna codificación y pruebas toman lugar de manera paralela. Ese punto

representa una de las principales ventajas sobre las fases secuencias que observamos en el modelo clásico del ciclo de vida. En particular, esto puede significar una retroalimentación entre la actividad de codificar, probar y efectuar debugging.

Aún más importante, el uso de una implementación top-down, permite a los implementadores, hablar con los usuarios aún y cuando las especificaciones hayan sido definidas de una manera formal. Esto es, es posible que el usuario describa errores en su entendimiento de las especificaciones iniciales; sin embargo, el usuario puede aún expresar un cambio deseado en alguna de esas especificaciones, y si la conversación toma lugar directamente entre alguno de los implementadores y el usuario, los cambios pueden ser llevados a cabo sin que el gerente del proyecto se entere de lo que está ocurriendo. En resumen, la implementación top-down proporciona una retroalimentación entre el proceso de implementación y el proceso de análisis.

Hay un punto final acerca de este ciclo de vida: Una parte significativa del trabajo que toma lugar bajo el título de "diseño estructurado", es actualmente un esfuerzo manual por corregir las pésimas especificaciones descritas. Esto se puede observar, en la figura II.3.2.2 que describe los detalles del diseño estructurado. (Nótese que esta figura consiste de los detalles del proceso 3 de la figura II.3.2.1).

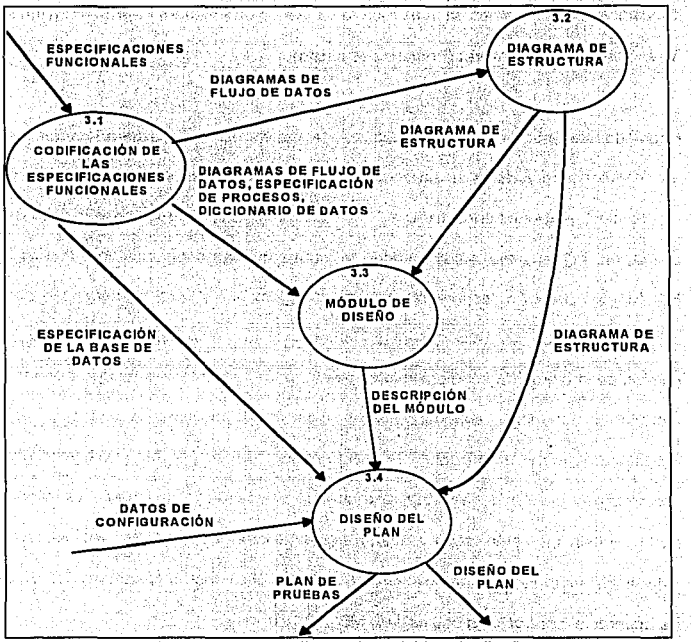


Figura II.3.2.2 Detalles de la actividad de diseño

La integración top-down, es una propuesta incremental para la construcción de la estructura de programas. Los diferentes módulos se van integrando en un movimiento hacia abajo a través de un control jerárquico, comenzando con el módulo de control principal (programa principal). Los módulos subordinados al módulo de control

principal son incorporados en la estructura ya sea de una manera **depth-first** (Primero hacia abajo) o **breadth-first** (Primero a lo ancho).

Haciendo referencia a la figura II.3.2.3, la integración depth-first deberá integrar todos los módulos involucrados en un camino de control de la estructura. La selección de ese camino, es de manera arbitraria y depende de las características específicas de la aplicación. Por ejemplo, seleccionando la parte izquierda como camino, los módulos M1, M2, M5 deberán ser integrados primero. Después M8 o (si es necesario por la propia función de M2) M6 deberá ser integrado. Entonces el camino central y el camino de la izquierda son construidos. Por su parte, breadth-first incorpora todos los módulos directamente subordinados en cada nivel, moviéndose a través de la estructura de manera horizontal. De la figura, los módulos M2, M3 y M4 (este último reemplazado, para poder iniciar, por un módulo S4. Este tipo de módulo se llamará a partir de este momento, **stub** y nos proporciona información que es recibida por el módulo del nivel superior al que reporta) deberán ser integrados primero. Le siguen, de acuerdo al control los módulos, M5, M6, etc.

El proceso de integración es realizado en una serie de cinco pasos:

- El módulo principal de control es utilizado como un driver de prueba, donde los resultados generados en los diferentes niveles son sustituidos en todos los módulos directamente subordinados al módulo principal de control.

- Dependiendo de la propuesta de integración seleccionada (depth-first o breadth-first) los resultados del módulo inferior son reemplazados uno a la vez en los módulos del nivel actual.

- Las pruebas son conducidas como cada módulo se vaya integrando. Esto es, cada vez que un módulo opera correctamente permitirá pasar al siguiente nivel, para continuar con las pruebas en éste.

- Al completar un conjunto de pruebas, cada módulo stub es reemplazado por su módulo real.

- Pruebas de regresión (Esto es, realizar nuevamente todas o algunas de las pruebas previas) pueden ser efectuadas para asegurar que no se están introduciendo nuevos errores.

El proceso continua del segundo punto en adelante, hasta que el programa por entero está construido. La figura II.3.2.3 ilustra ese proceso.

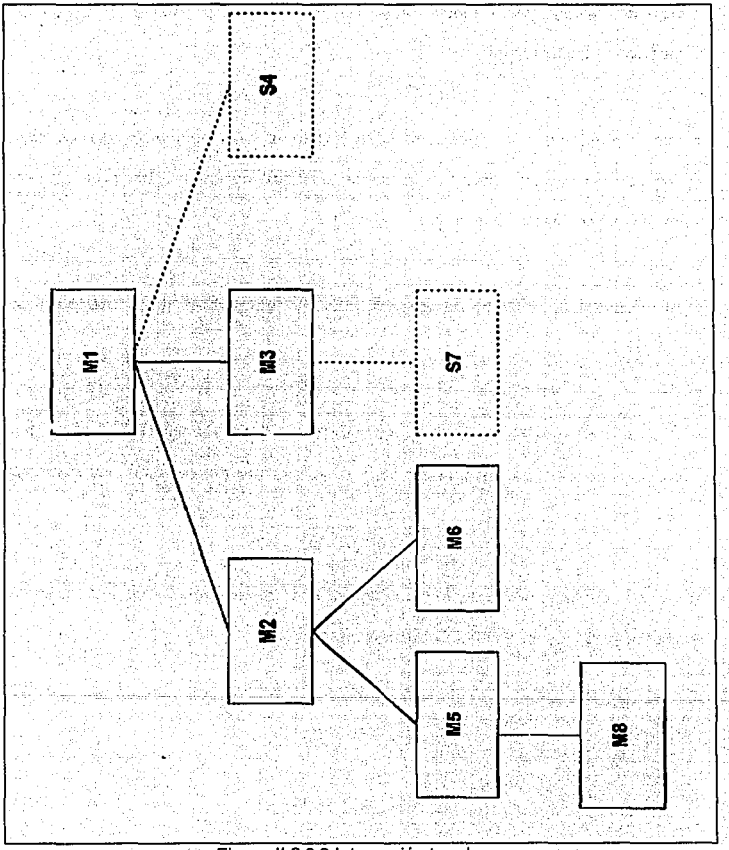


Figura II.3.2.3 Integración top-down

Asumiendo una propuesta depth-first y una estructura parcialmente completa, con el módulo stub S7, el cual será reemplazado con el módulo M7. El cual, a su vez, posteriormente estará conformado por módulos stub, y éstas serán sustituidos por sus correspondientes módulos. Es importante notar que en cada reemplazo de módulos stub por los módulos reales, las pruebas serán nuevamente realizadas a fin de verificar la interface.

La estrategia de integración top-down verifica los puntos de mayor control o decisión desde el principio del proceso de prueba. En un programa con una estructura bien realizada, la toma de decisiones se lleva a cabo en los niveles más altos de la jerarquía y por lo tanto son encontrados fácilmente. Si la integración depth-first es seleccionada, una función del software puede ser implementada y demostrada fácilmente. Por ejemplo, considérese una transacción clásica en que una compleja serie de entradas interactivas son requeridas, adquiridas y validadas vía un determinado camino. Ese camino deberá ser integrado de alguna manera con la técnica top-down. El procesamiento de todas la entradas puede ser demostrado antes que los demás elementos de la estructura hayan sido integrados.

La estrategia top-down suena relativamente sencilla, pero en la práctica, se presentan problemas en la logística a seguir. El más común de esos problemas ocurre cuando el resultado del proceso de los niveles más bajos en la jerarquía se requiere para una adecuada prueba de los niveles superiores. Los módulos stub reemplazan a los

módulos de más bajo nivel al comienzo de la prueba top-down; por lo tanto no significan datos que puedan fluir hacia la parte superior de la estructura del programa.

El encargado de realizar las pruebas se enfrenta a tres opciones:

- Retrasar muchas pruebas hasta que los módulos stub son reemplazados por los módulos reales.
- Desarrollar módulos stub que realicen funciones limitadas, las cuales simulan al módulo real.
- Integrar el software desde la parte inferior de la jerarquía hacia arriba. (Técnica bottom-up).

La figura II.3.2.4, ilustra las clases típicas de los módulos stub, abarcando desde el más simple (stub A), hasta el más complejo (stub D).

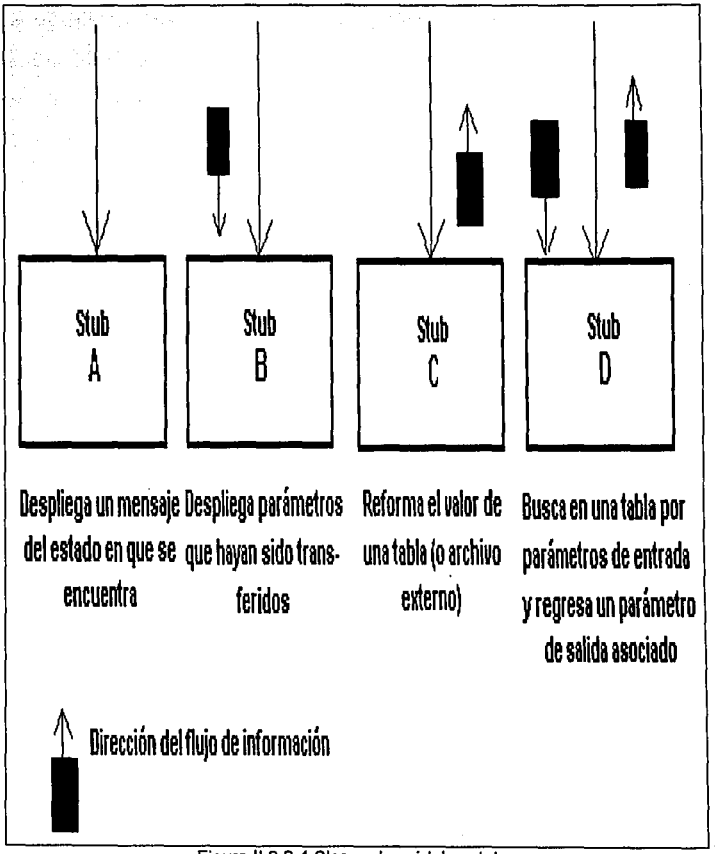


Figura II.3.2.4 Clases de módulos stub.

La primer alternativa (retrasar las pruebas hasta reemplazar los módulos stub con los módulos reales) provoca la pérdida de control en la correspondencia entre pruebas específicas y la incorporación de los módulos reales específicos. Esto puede generar una dificultad en la determinación de la causa del error y tiende a violar la naturaleza propia de la técnica top-down; al estar integrando un grupo considerable de módulos antes de hacer las pruebas por separado. La segunda alternativa se puede llevar a cabo, pero puede significar el desarrollo de módulos stub cada vez más complejos. Por último, la tercer alternativa se refiere a utilizar la técnica bottom-up discutida con anterioridad.

Como hemos apreciado, existen muchas discusiones relativas a las ventajas y desventajas en las pruebas de integración top-down y bottom-up. En general las ventajas de una estrategia tienden a resultar en desventajas para la otra estrategia. La mayor desventaja del top-down es la necesidad de los módulos stub y las dificultades que se generan asociadas a los mismos. Los problemas generados por los stub, pueden ser compensados por la ventaja de realizar pruebas desde un principio. Por su parte, la mayor desventaja de bottom-up es que, el programa como una entidad no existe hasta que el último módulo es integrado. Como ventaja tiene el hecho de ser probado en su diseño de una manera muy sencilla y el no necesitar de módulos stub.

II.3.3 TECNICA YOURDON

En las secciones anteriores hemos visto en detalle herramientas de modelado que se utilizan en el análisis de sistemas. Cada una de ellas nos permite enfocarnos selectivamente a los aspectos individuales de un sistema cuyas características es importante entender, como lo son: las funciones que el sistema debe desempeñar, los datos que debe manejar y su comportamiento en el tiempo. Tales herramientas nos permitirán modelar (o describir o imaginarse) prácticamente cualquier tipo de sistema: de negocios, biológicos, manufactura, políticos, de flujo de materiales, etc. Vivimos en un mundo de sistemas, y la mayor parte de nuestra vida cotidiana se emplea en comprenderlos e interactuar con ellos.

Esta sección presenta la técnica de análisis de sistemas de Edward Yourdon, la cual, se dirige a la construcción de sistemas de información automatizados. En ella se involucra el desarrollo de diversos tipos de modelos.

EL MODELO ESENCIAL

Cuando se tienen distintas herramientas de modelado, pueden plantearse preguntas como las siguientes: ¿Que tipo de modelo se debe construir?, ¿Se debe construir el modelo actual de implementación del sistema?, ¿Se debe construir el modelo de la

implantación que ahora se desea?, ¿Se construye un modelo independiente de la tecnología de implantación?. ¿Se construye todo lo anterior?.

En la *técnica Yourdon* se recomienda evitar modelar el sistema actual, si no es necesario. Y tan pronto como sea posible, comenzar a desarrollar un modelo del nuevo sistema que se desea. Este nuevo sistema se conoce como **Modelo Esencial** del sistema (otros autores lo nombran nuevo sistema lógico).

Ocasionalmente existirá alguna situación que amerite construir el sistema actual del usuario; esto sucede, por ejemplo, cuando se necesita profundizar en procesos esenciales que no son del todo entendibles y que son indispensables en el análisis actual.

El modelo esencial del sistema es un modelo de **lo que** el sistema debe hacer para satisfacer los requerimientos del usuario, indicando lo mínimo posible (de preferencia nada) acerca de **cómo** se implantará. El modelo del sistema supone que se tiene disponible una tecnología capaz y que se puede obtener fácilmente y sin costos elevados.

Concretamente, cuando se trata de identificar los requerimientos del sistema, se debe evitar describir las implantaciones específicas de los procesos (círculos en un DFD), también, no se debe mostrar las funciones realizadas por humanos o por otros

sistemas existentes. La figura II.3.3.1 muestra un modelo esencial que es apropiado para describir *lo que* el sistema debe realizar sin importar como es su implantación final.

Lo mismo sucede para los *flujos de datos* y *almacenes de datos*: el modelo esencial debe describir el contenido de los flujos o almacenes de datos, sin describir el medio (por ejemplo, disco o cinta) u organización física de los datos.

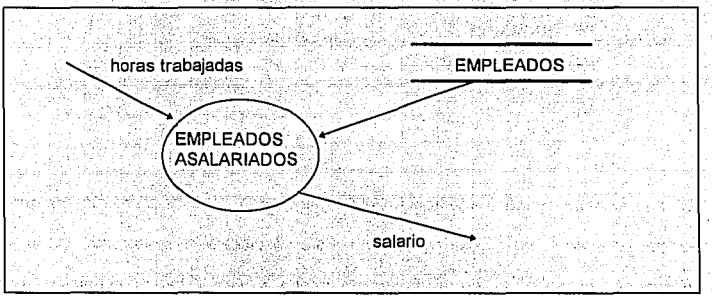


Figura II.3.3.1 Un modelo de **cuál es** la función del sistema

DIFICULTADES EN LA CONSTRUCCION DE UN MODELO ESENCIAL

Aunque las reglas antes mencionadas parecen simples y obvias, a menudo resulta muy difícil eliminar completamente todos los detalles de la implementación en el modelo esencial. Los errores más comunes en un modelo esencial son:

- Secuenciado arbitrario de las actividades en un DFD.
- Archivos innecesarios.
- Revisión de errores y validación innecesarias de datos y procesos dentro del sistema.
- Datos redundantes o derivados.

COMPONENTES DEL MODELO ESENCIAL

El modelo esencial consiste en dos componentes principales:

- Modelo ambiental.
- Modelo de comportamiento.

El *modelo ambiental* define la frontera entre el sistema y el resto del mundo, es decir, el ambiente en el cual existe el sistema. Este se discute más adelante. Como se observará, consiste de un diagrama de contexto, una lista de eventos y una descripción breve del propósito del sistema.

El *modelo de comportamiento* describe el comportamiento que se requiere del sistema para que interactúe de manera exitosa con el ambiente. Consiste en diagramas de flujo de datos, de entidad-relación, diccionarios de datos y especificaciones de procesos.

EL MODELO AMBIENTAL

Dentro del análisis de un sistema es necesario determinar qué es parte del sistema y qué no. Cualquier sistema que se desarrolle, será parte de un sistema aún mayor.

Así, el primer modelo importante de un sistema que se debe desarrollar es uno que no haga más que definir las **interfases** entre éste y el resto del universo, es decir, el **ambiente**. Este se conoce como **modelo ambiental**, el cual, modela el comportamiento exterior del sistema.

Además de determinar qué está en el interior y qué en el exterior (lo que se logra definiendo la **frontera** entre el sistema y el ambiente), también se necesita saber que información entra al sistema desde el exterior y cual es la que produce como salida.

Las entradas y salidas siempre se presentan en forma racional. Tienen un propósito específico como evento derivado de algún acontecimiento o estímulo en el ambiente. Por ello, otro aspecto crítico del modelo ambiental es identificar los **eventos que ocurren en el ambiente al cual debe responder el sistema**. Sólo se deben de considerar eventos que (1) ocurren en el ambiente exterior y (2) requieren una respuesta del sistema.

La frontera entre un sistema y su ambiente es **arbitraria**, como se observa en la figura II.3.3.2. Sin embargo, puede considerarse alguna política administrativa, negociaciones políticas u otras estrategias para definir una frontera.

Generalmente se tiene una buena idea de la frontera **general** entre el sistema y el ambiente, pero como se muestra en la figura II.3.3.3, a menudo existe un "área gris" que esta abierta a consideraciones. Se trata de una área sobre la cual (1) no hay certidumbre o, (2) no se ha pensado o, (3) se tienen ideas preconcebidas que se tienen que reflexionar.

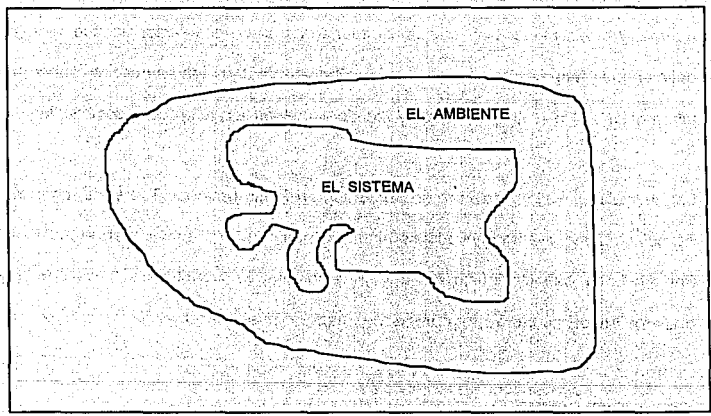


Figura II.3.3.2 La frontera entre el sistema y el ambiente.

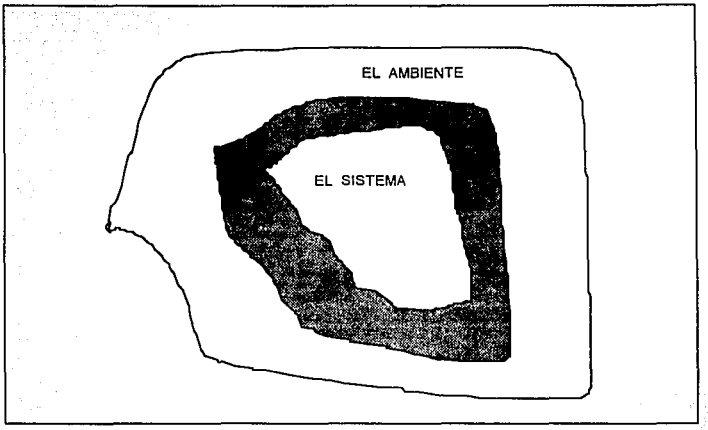


Figura II.3.3.3 El área gris entre el sistema y el ambiente.

El área dentro de la frontera del sistema suele conocerse también como el **dominio de cambios**. Todo lo que se encuentre en ella esta sujeto a cambios (por ejemplo, reorganización y/o automatización), mientras que todo lo que esta fuera se queda en su forma actual y no es investigado.

COMPONENTES USADOS PARA DEFINIR EL AMBIENTE

El modelo del ambiente consta de tres componentes:

1. Declaración de propósitos

2. Diagrama de contexto.

3. Lista de eventos.

1. LA DECLARACION DE PROPOSITOS

La declaración de propósitos del modelo ambiental es una declaración textual breve y concisa del **propósito** del sistema, dirigida al nivel administrativo superior, la administración de los usuarios, y otros que no estén directamente involucrados con la construcción del sistema.

El siguiente es un ejemplo de una declaración de propósitos:

"*El propósito del presente trabajo es diseñar e implementar una base de datos que administre el equipo, usuarios, servidores, estadísticas y respaldos, así como optimizar los recursos de software y hardware que deberán operar en un ambiente visual amigable en el Centro de Cálculo del ITESM campus Ciudad de México.*"

La declaración de propósitos puede constar de una o varias frases. Sin embargo, no debe exceder más de un párrafo, ya que la intención no es proporcionar una descripción completa y detallada del sistema. Como resultado de ello, la declaración de propósitos será intencionalmente vaga en cuanto a muchos detalles, los cuales, son aclarados en el modelo de comportamiento.

También en la declaración de propósitos es conveniente resumir los beneficios tangibles y cuantificables que el sistema puede proporcionar.

2. EL DIAGRAMA DE CONTEXTO

El diagrama de contexto es un caso especial de un diagrama de flujo de datos, en donde **un sólo** proceso (representado por un círculo) representa todo el sistema.

A partir de un diagrama de contexto es posible enfatizar varias características importantes del sistema, ellas son:

- Las personas, organizaciones y sistemas con los que se comunica el sistema. Estos se conocen como **terminales**.
- Los datos que el sistema recibe del mundo exterior y que deben procesarse de alguna forma.
- Los datos que el sistema produce y que se envían al mundo exterior.
- Los almacenes de datos que el sistema comparte con los terminales. Tales almacenes de datos se crean fuera del sistema para su uso, o bien, son creados en él y usados en el exterior.

La figura II.3.3.4 presenta un diagrama de contexto para un sistema de pedido de libros. Este sistema es utilizado como ejemplo de la técnica Yourdon para análisis de sistemas al final de la presente sección. Las técnicas para la construcción de diagramas de contexto se discuten más adelante.

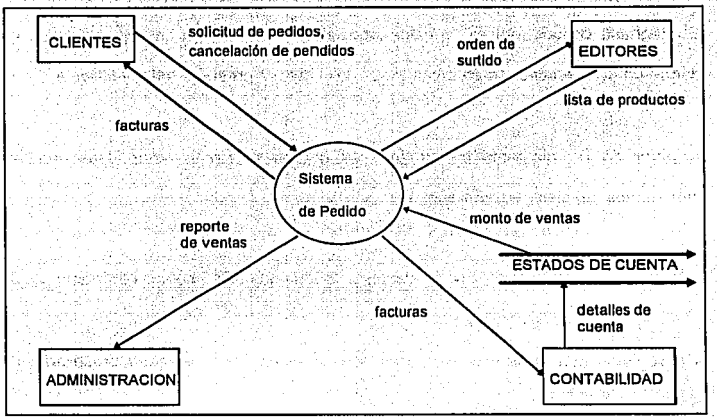


Figura II.3.3.4 Diagrama de contexto de un Sistema de Pedido de Libros.

3. LA LISTA DE EVENTOS

La lista de eventos es una lista narrativa de los eventos que ocurren en el mundo exterior a los cuales el sistema debe responder.

A continuación se muestra una posible lista de eventos para el sistema de apartado de libros de la figura II.3.3.4:

1. Un cliente hace un pedido **(F)**
2. Un cliente cancela un pedido **(F)**
3. La administración pide un reporte de ventas **(T)**
4. Llega un pedido de reimpresión de un libro a la bodega **(C)**

Se observa que cada evento se etiqueta como **F**, **T** o **C**. Con ello se muestra si es un evento de tipo **flujo**, **temporal**, o de **control**.

Un *evento de flujo* es aquel que se asocia con un flujo de datos; es decir, donde el sistema se entera que ha llegado algún dato (o posiblemente varios). Es evidente que esto corresponde al flujo de datos en el diagrama de contexto.

Sin embargo, no todos los flujos de datos en un diagrama de contexto necesariamente son eventos de tipo flujo. Considérese el diagrama de contexto parcial que se muestra en la figura II.3.3.5. En el diagrama puede deducirse que los flujos de datos *A*, *B* y *C* son todos indicadores de eventos separados y discretos. Pero quizás sólo el flujo de datos *A* resulte estar asociado con un evento (por ejemplo, al flujo de datos lo inicia un terminal). Para procesar un evento, el sistema **explícitamente** podría pedir entradas a otros terminales a lo largo de los flujos de datos *B* y *C* para otorgar alguna respuesta.

Así que no necesariamente existe una correspondencia uno a uno entre los flujos de datos del diagrama de contexto y los eventos de la lista de eventos. En general, cada flujo de datos es un evento (o, más precisamente, la indicación de que un evento ha ocurrido), o bien, es una requisición adicional del sistema para poder procesar un evento.

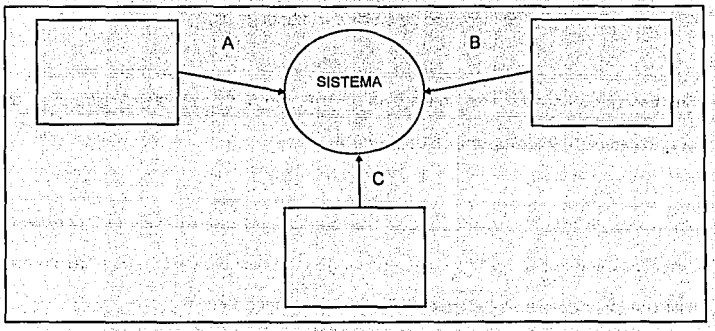


Figura II.3.3.5 Diagrama parcial de contexto

Un sistema también puede tener *eventos temporales.* Como su nombre lo indica, los eventos temporales inician en un momento determinado de tiempo.

Algunos ejemplos de eventos temporales pueden ser:

- A las 9:00 A.M se requiere un reporte diario de todos los pedidos de libro.
- Las facturas deben generarse a las 7:00 P.M.
- Se deben generar reportes administrativos una vez por hora, etc.

Debe observarse que los eventos temporales no se **inician** con flujos de datos de entrada; puede imaginarse que el sistema tiene un reloj interno con el cual puede determinar el paso del tiempo. Sin embargo, debe tenerse en mente también que un evento temporal podría requerir que el sistema solicite entradas de uno o más terminales. Por ello, podrían asociarse uno o más flujos de datos con un evento temporal, aunque los flujos de datos, en sí, no representan el evento mismo.

Los *eventos de control* pueden considerarse como un caso especial del evento temporal: un evento externo que ocurre en algún momento impredecible. A diferencia de un evento temporal, el evento de control no se asocia con el paso regular del tiempo, por lo que el sistema no puede anticiparlo utilizando el reloj interno. Y a diferencia de un flujo normal, el de control no indica su presencia con el arribo de datos. Como lo muestra la figura II.3.3.6, un evento de control se asocia con un flujo de control en el diagrama de contexto.

El flujo de control puede considerarse como un flujo de datos binario: está encendido o apagado, y puede cambiar de estado en cualquier momento, señalando así al sistema que se necesita tomar alguna acción **inmediata**. Los flujos de control son bastante comunes en sistemas de tiempo real. Pero en otro tipo de sistemas no suelen tener flujos de control en sus diagramas de contexto.

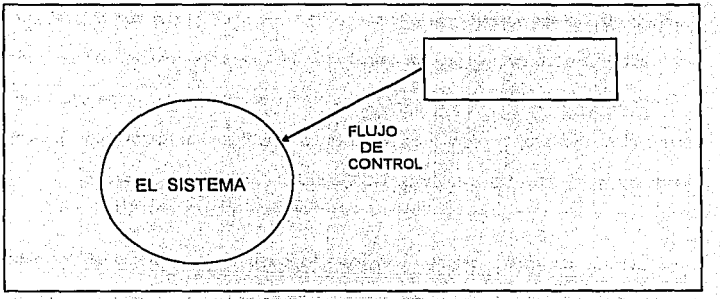


Figura II.3.3.6 Flujo de control asociado con un evento de control.

COMPONENTES ADICIONALES DEL MODELO AMBIENTAL

Pueden ser útiles dos componentes adicionales para el modelo ambiental dependiendo de la naturaleza o complejidad del sistema:

- El diccionario de datos inicial, que define todos los flujos y almacenes de datos externos.
- El diagrama entidad-relación de las fuentes de datos externas.

Estos componentes son una información adicional para tener en claro las interacciones entre los terminales y el sistema mismo.

CONSTRUCCION DEL MODELO AMBIENTAL

Usualmente, como otras técnicas de modelado, el modelo ambiental se construye con una serie de refinamientos iterativos; en cada una de ellos se agregan datos adicionales y/o corregidos para una nueva revisión. La construcción de los elementos del modelo ambiental se tratan en detalle a continuación.

CONSTRUCCION DEL DIAGRAMA DE CONTEXTO

La parte más difícil de un diagrama de contexto es el proceso; que consiste de un círculo en un DFD. El nombre de tal proceso suele ser el nombre completo del sistema o un acrónimo convenido. En un caso extremo, el sistema puede representar una organización completa; el nombre del proceso sería el de la organización misma.

Los terminales, como hemos visto se representan con rectángulos en el diagrama de contexto. Se comunican con el sistema a través de flujos de datos o de control. Estos no deben de comunicarse directamente entre si, esto es incorrecto dentro de un diagrama de contexto. Aclaraciones adicionales sobre los terminales son:

- Algunos tienen un amplio número de entradas y salidas. Para evitar un diagrama innecesariamente saturado, conviene dibujar el terminal más de una vez y ser marcado por un asterisco o una diagonal.

- Cuando el terminal es una persona individual, generalmente es preferible indicar el roll que desempeña, más que su identidad.
- Es importante distinguir entre almacenes de datos y *manejadores de base de datos*. Se entiende por almacenes de datos a *bases de datos y archivos*; mientras un manejador es un mecanismo, dispositivo o medio físico usado para transportar datos hacia dentro o fuera del sistema.

Los flujos que aparecen en el diagrama de contexto se incluyen para:

- Detectar un acontecimiento en el ambiente en que deba responder el sistema, o si se ocupan (como datos) para producir una respuesta.
- Ilustrar datos que son transportados entre los terminales y el sistema.

El diagrama de contexto de un modelo esencial debe evitar (hasta donde sea posible), mostrar los manejadores cercanos a la implementación que introducen y sacan datos del sistema. Además, tampoco deberá mostrar los mensajes y medios específicos de coordinación que el sistema y los terminadores pasan entre sí para indicar que están listos para las entradas o salidas.

En lugar de lo anterior, es conveniente dibujar el diagrama de contexto bajo el supuesto de que las entradas son causadas e **iniciadas** por los terminales y que las salidas son causadas e iniciadas por el sistema.

Aún así, habrá ocasiones en que el terminal no inicie las entradas pues, aún con la tecnología capaz, este no sabe que el sistema requiere sus entradas. Similarmente, hay ocasiones en que el sistema no inicia la generación de salidas, debido a que no sabe que el terminal las necesita o desea. En ambos casos, el mensaje es una parte esencial del sistema. A veces resulta conveniente mostrar el mensaje y el correspondiente flujo de entrada o salida con un flujo de diálogo (una flecha de dos cabezas).

CONSTRUCCION DE LA LISTA DE EVENTOS

Al crear la lista de eventos debe asegurarse de distinguir entre un evento y un flujo relacionado con un evento.

En la mayor parte de los casos, la manera más fácil de identificar los eventos relevantes para un sistema es visualizarlo en acción: examinar cada terminal y preguntar que efecto puede tener sus acciones sobre el sistema.

CONSTRUCCION PRELIMINAR DEL MODELO DE COMPORTAMIENTO

De acuerdo al modelo ambiental, hasta el momento se debe tener listo un diagrama de contexto, una lista de eventos y una declaración de propósitos. Además, se ha

comenzado a construir el diccionario de datos, contando por lo menos con la definición de los datos que representan las interfaces entre los terminales externos y el sistema.

A continuación se debe iniciar la construcción del **modelo de comportamiento** del sistema. Esto involucrará el desarrollo de un diagrama de flujo de datos y un diagrama entidad-relación preliminares, además de la elaboración de las entradas iniciales del diccionario de datos.

Básicamente este enfoque implica dibujar el borrador del diagrama de flujo de datos, con un proceso (círculo) para la respuesta del sistema ante **cada evento** que se indicó en la lista de eventos (identificación de respuestas a eventos). A continuación se dibujan almacenes de datos en el borrador del DFD para modelar los datos que deben recordarse entre eventos no sincronizados. Finalmente, se conectan los flujos de entrada y salida apropiados a los procesos (círculos) y se compara el conjunto de diagramas de flujo de datos contra el diagrama de contexto para asegurar la consistencia.

Una vez hecho esto se procede a un proceso de limpieza, descrito más adelante, para producir un modelo bien organizado del proceso y un modelos de datos final.

IDENTIFICACION DE RESPUESTAS A EVENTOS

El enfoque de partición por eventos incluye los siguientes cuatro pasos:

1. Dibujar un círculo, o proceso, para cada evento de la lista.
2. Nombrar al círculo de manera que describa la respuesta que el sistema debe dar al evento asociado.
3. Dibujar las entradas y salidas apropiadas de tal forma que el círculo pueda dar la respuesta requerida, y dibujar los almacenes de datos, como sea apropiado, para la comunicación entre círculos.
4. El borrador de DFD **preliminar** que resulta se compara con el diagrama de contexto y la lista de eventos para asegurar que esté completo y sea consistente.

En el desarrollo de los puntos anteriores, existen dos casos: (1) eventos únicos que causan múltiples respuestas y, (2) eventos múltiples que causan la misma respuesta. El primer caso se resuelve dibujando un proceso para cada respuesta; pero esto solo es apropiado si todos las respuestas usan el mismo flujo de entrada y si todas las repuestas son independientes entre sí. Esto se ilustra en la figura II.3.3.7.

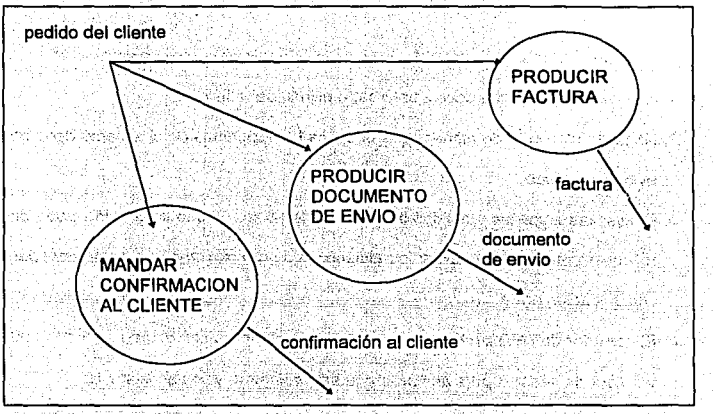


Figura II.3.3.7 Múltiples respuestas de un evento.

En el segundo caso, se tiene que un proceso se asocia con más de un evento. Tal situación es valida y apropiada si la respuesta del proceso es **idéntica** para los diversos acontecimientos, y sólo si los datos de entrada y salida son idénticos para las diversas respuestas a eventos. La figura II.3.3.8 ilustra el caso.

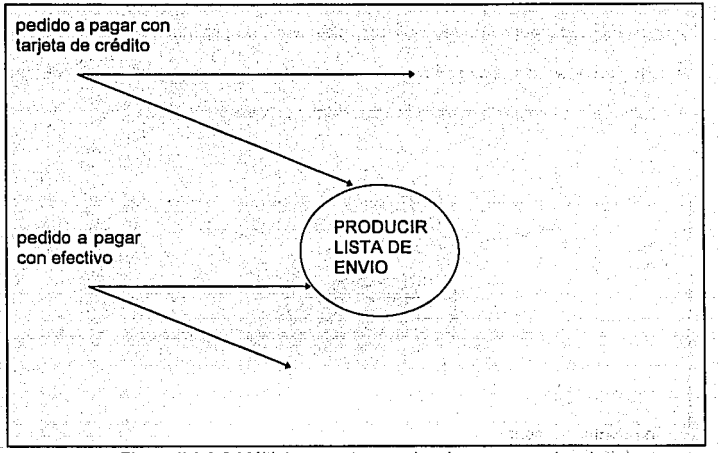


Figura II.3.3.8 Múltiples eventos con la misma respuesta

CONEXION DE LAS RESPUESTAS A EVENTOS

Las dos figuras anteriores nos muestran que **los procesos no se comunican entre sí. Estos deben comunicarse a través de otros almacenes de datos.**

Lo anterior se debe a que los procesos en un DFD preliminar representan respuestas a eventos, y los eventos que ocurren en el ambiente externo son, en el caso general, no sincronizados. La única forma de sincronizar múltiples acontecimientos

interdependientes es mediante una fuente de datos. En este caso, se trata de almacenes de datos esenciales, que son necesarios, no por retrasos asociados a una tecnología imperfecta, sino por consideraciones de tiempo en el ambiente. La figura II.3.3.9 muestra el modelo de comunicación entre procesos.

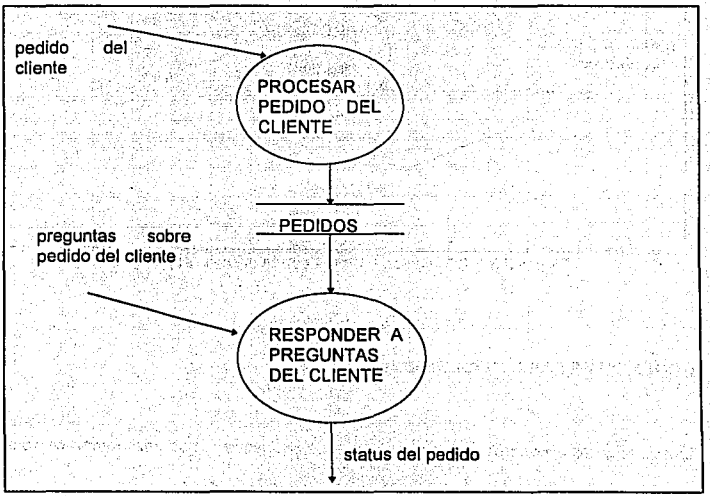


Figura II.3.3.9 Modelo de comunicación entre procesos

DESARROLLO DEL MODELO INICIAL DE DATOS

En esta etapa de análisis, el diagrama entidad-relación y el DFD se desarrollan en paralelo y en forma **independiente**, y esto es una ventaja para que se revisen entre sí. Los almacenes de datos que se definen tentativamente en el DFD preliminar pueden usarse para sugerir ***objetos*** en el diagrama entidad-relación preliminar; y los objetos que se identificaron tentativamente en el diagrama entidad-relación pueden usarse para ayudar a escoger almacenes de datos apropiados en el DFD preliminar. Ningún modelo debe prevalecer sobre otro.

Así mismo, la lista de eventos es útil para la creación del diagrama entidad-relación inicial como para crear el DFD inicial. Por ejemplo, si un evento es "Cliente hace pedido", se puede deducir que el diagrama entidad-relación puede tener las entidades CLIENTE y PEDIDO. De manera similar las entidades del diagrama entidad-relación deberán estar presentes en las declaraciones hechas en la lista de eventos.

CONCLUSION DEL MODELO DE COMPORTAMIENTO

La conclusión del modelo de comportamiento consiste en los siguientes puntos:

- Nivelar hacia arriba un DFD inicial.
- Ocultar los almacenes de datos.
- Partir los procesos iniciales del DFD hacia abajo.

- Completar el diccionario de datos.
- Completar las especificaciones del proceso.
- Completar el modelo de datos.

Estos puntos se tratan a continuación.

NIVELACION DEL DFD

El DFD preliminar consta de un sólo nivel, con demasiados procesos. Por ello es necesario una nivelación **ascendente** del DFD preliminar. Esto significa que se deben agrupar procesos **relacionados** en agregados con significado, cada uno de los cuales representará un proceso (círculo) de un diagrama de nivel superior. Esto se ilustra en la figura II.3.3.10.

Existen tres reglas que se deben seguir para nivelar un DFD:

1. Cada agrupación de procesos debe involucrar respuestas estrechamente relacionadas (recordar que cada proceso del DFD preliminar se nombra de acuerdo con la respuesta a un evento de la lista). Usualmente esto significa que los procesos manejan datos cercanamente relacionados.

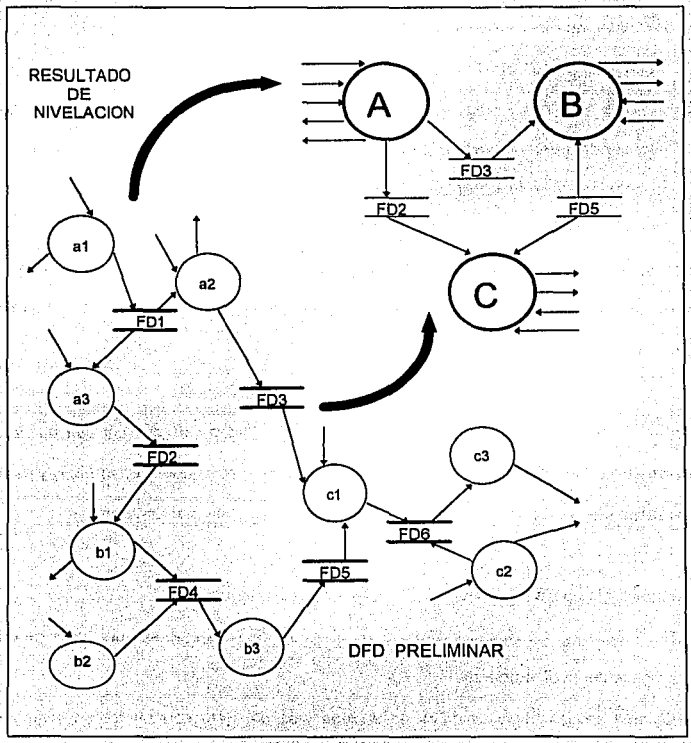


Figura II.3.3.10 Nivelación ascendente del DFD.

2. Ocultar los almacenes de datos. Si un grupo de procesos en los DFD preliminares se refieren a un *almacén de datos común*, y no hay otros procesos en el DFD

preliminar que se refieran a ésta, entonces puede crearse un proceso de nivel superior para ocultarla. Esto se ilustra en la figura II.3.3.11.

3. No crear un gran número de agregados o grupos del DFD preliminar. Se recomienda que sean aproximadamente 7 o más o menos 2 bloques de información, donde un proceso (y sus flujos relacionados) se consideren como un bloque.

Son necesarios varios intentos de nivelación ascendente. Las dos primeras reglas mencionadas deben ser el parámetro principal para la nivelación ascendente, y no hay ninguna regla aritmética necesariamente.

Podría requerirse de una **nivelación descendente**. Esto puede presentarse cuando los procesos identificados en el DFD deben analizarse o revisarse nuevamente, y esto se realiza en particiones descendentes, o sea, en los DFD de nivel inferior.

Algunas reglas para llevar a cabo la nivelación descendente son:

- Realizar una descomposición funcional, es decir, identificar las subfunciones de un proceso de nivel superior.
- Guiarse por los flujos de datos de entrada y salida. Esto se muestra en la figura II.3.3.12, donde se muestra la nivelación descendente de un proceso complejo "ABC".

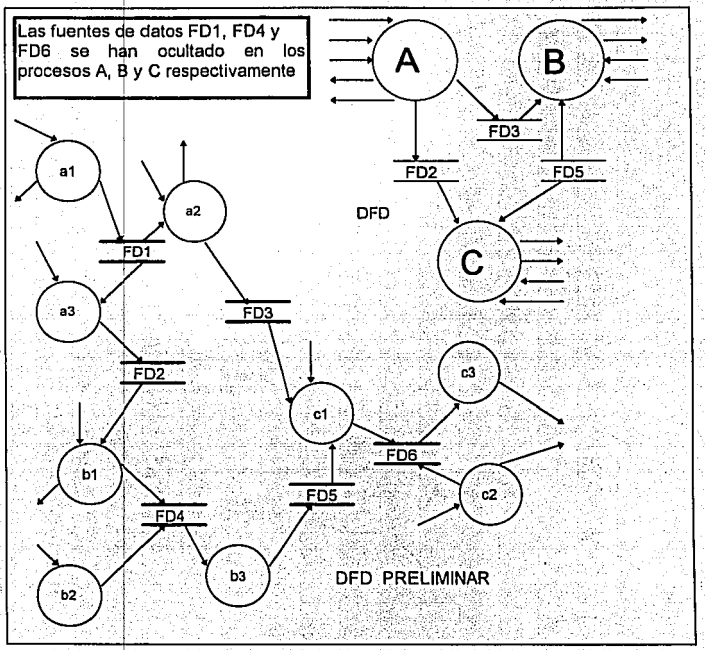


Figura II.3.3.11 Como ocultar un fuente de datos local en un nivel superior.

En la nivelación ascendente y descendente el balanceo es importante. Es decir, se debe asegurar que las entradas y salidas de un proceso que se muestran en el nivel superior correspondan en el nivel inferior

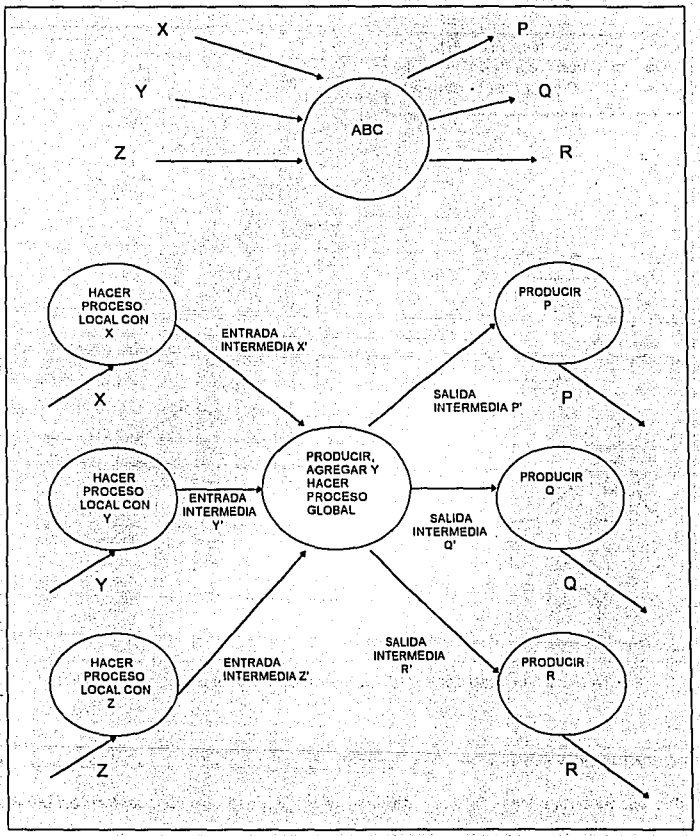


Figura II.3.3.12 Nivelación descendente del proceso de nivel superior "ABC"

COMO COMPLETAR EL DICCIONARIO DE DATOS

Es común desarrollar el diccionario de datos cuando se esta elaborando el diagrama de contexto. Sin embargo no estará completo en ese momento. Aún es necesario llenar la descripción del **significado** de cada dato; también es apropiado dividir los datos complejos en elementos menores por claridad.

Al completar el diccionario de datos se debe verificar que estos: (1) sean consistentes, (2) estén balanceados con el diagrama de flujo de datos por niveles, (3) estén balanceados con el diagrama entidad-relación y (4) estén balanceados con las especificaciones del proceso.

COMO COMPLETAR LAS ESPECIFICACIONES DEL PROCESO

Cuando se desarrolla un DFD preliminar, es probable que no se escriban especificaciones del proceso. De hecho suele ser una mala idea dedicar tiempo a la escritura de las especificaciones del proceso antes de terminar el DFD preliminar, porque este aún esta sujeto a cambios y correcciones, incluso tal vez desaparecer posteriormente.

Por tanto, es conveniente esperar una estabilización del DFD preliminar, y cuando pase la prueba de nivelación ascendente (es decir, si la actividad no descubre fallas

importantes en el modelo), entonces pueden comenzarse a escribir las especificaciones del proceso.

TERMINADO DEL MODELO DE DATOS

El modelo esencial final debe contener lo siguiente:

- Declaración de propósitos.
- Diagrama de contexto.
- Lista de eventos.
- Diagrama entidad-relación completo y terminado.
- Conjunto completo de diagramas de flujos de datos por niveles.
- Diccionario de datos completo (para la fase de análisis del proyecto).

EJEMPLO DE LA TECNICA YOURDON

Tomando el Sistema de Pedidos de Libros que se mostró en la figura II.3.3.4, a continuación se muestran los resultados que es posible obtener utilizando la técnica Yourdon para análisis de sistemas. La intención del siguiente ejemplo es *ilustrar* lo que se ha mencionado en esta sección, por lo que se presentan *parcialmente* los resultados que se logran en algunos pasos, como lo son aquellos que se refieren a la

lista de eventos completa, todos los *diagramas de flujo de datos por niveles, diccionario de datos* completo y todas las *especificaciones de procesos de nivel inferior.*

Declaración de propósitos:

"El propósito del Sistema de Pedidos de Libros es controlar los detalles de los pedidos de libros que realizan los clientes, además de la facturación y cobros retroactivos de las ventas. La información acerca de los pedidos de libros debe estar disponible para otros sistemas, tales como mercadeo, ventas y contabilidad."

Diagrama de contexto:

Se muestra en la figura II.3.3.4. En éste, el elemento terminal *clientes* realiza la solicitud de compra de un libro, el cual en caso de existir, puede disponer de más de un ejemplar y pagar periódicamente hasta que cubra el costo total y posteriormente recibir una factura; en caso contrario, puede esperar a que el ejemplar sea surtido por el elemento terminal *editores* y dejar su solicitud de pedido. Los *editores* se encargan además de enviar listas de nuevos títulos al sistema.

Las facturas que son generadas por la venta de libros son enviadas al elemento terminal *contabilidad*, que es el departamento responsable de la generación y envío de los movimientos contables al almacén de datos *estados de cuentas*.

Cuando el elemento terminal *administración* solicita un reporte de ventas, las cifras de éstas son tomadas del almacén de datos *estados de cuentas.*

Lista de eventos:

1. Un cliente hace un pedido (F).
2. Un cliente cancela un pedido (F).
3. Se reciben pagos, parciales o totales, de los clientes (F).
4. Se emiten facturas para los clientes (F).
5. Se envían facturas al departamento de contabilidad (cierre de caja 7:00 p.m.) (T).
6. La administración pide un reporte de ventas (F).
7. Se envía ordenes de surtido a los editores (F).
8. Se recibe lista de títulos de los editores (F).

Diagrama Entidad-Relación:

Se muestra en la figura II.3.3.13. De acuerdo con el diagrama de contexto y la lista de eventos, se definen las entidades CLIENTE, LIBRO, FACTURA, EDITOR y ESTADOS DE CUENTA.

La relación entre CLIENTE y LIBRO es PEDIDO, en donde *un* CLIENTE puede pedir *N* libros y *un* LIBRO puede ser pedido por *N* clientes; así mismo, *un* pedido que ha sido pagado totalmente genera *una* FACTURA.

La relación MOVIMIENTO CONTABLE entre las entidades FACTURA y ESTADOS DE CUENTA indica que *N* FACTURA(s) generan *un* MOVIMIENTO CONTABLE.

Para la existencia y surtidos de libros se tiene la relación SURTIDO entre las entidades LIBRO y EDITOR, en donde *un* LIBRO es surtido por *un* EDITOR.

En el diagrama cada entidad y relación tiene como atributo su llave primaria. El mapeo de entidades y relaciones a tablas no incluye la relación MOVIMIENTO CONTABLE, ya que esta sólo se utiliza para modelar un evento entre las entidades que involucra.

Diccionario de datos:

Con el mapeo de entidades y relaciones se definen las tablas CLIENTE, PEDIDO, LIBRO, SURTIDO, EDITOR, FACTURA y ESTADOS DE CUENTA. Aquí se presenta el diccionario de datos que corresponde a la tabla CLIENTE.

Tabla: CLIENTE			
Descripción: Contiene la información sobre los clientes.			
Columna	**Tipo**	**Tamaño**	**Descripción**
*rfc_cliente	caracter	15	R.F.C. del cliente
*nom_cliente	caracter	40	Nombre del cliente
dir_cliente	caracter	50	Dirección del cliente
pob_cliente	caracter	30	Población del cliente
tel_cliente	caracter	10	Número de teléfono del cliente
fax_cliente	caracter	10	Número de fax del cliente

(*) Llave primaria4

Diagrama de flujo de datos de los eventos:

Se muestra en la figura II.3.3.14. De acuerdo con la lista de eventos y el diagrama entidad-relación, se observan *ocho* procesos y los almacenes de datos CLIENTES, PEDIDOS, LIBROS, FACTURAS, SURTIDOS, ESTADOS DE CUENTA Y EDITORES; estos últimos permiten la conexión entre los procesos.

En cada proceso se tiene como entrada un enunciado que corresponde a la lista de eventos, y en algunos casos, un flujo de datos adicional proveniente de un almacén de

datos que es necesario para la ejecución del proceso. Todos los procesos generan salidas, que como se indicó en el diagrama de contexto, van dirigidas a personas, almacenes de datos y departamentos del sistema.

Diagrama de flujo de datos nivelado:

Se muestra en la figura II.3.3.15. Para nivelar el DFD preliminar los procesos se agrupan de la siguiente forma:

- *Control de Pedidos.* Agrupa a los siguientes procesos: (1) Realiza solicitud de pedido, (2) Realiza cancelación de pedido, (3) Recibe pago de cliente y (4) Emite factura. Se oculta el almacén de datos PEDIDOS.

- *Control de Surtidos.* Agrupa a los siguientes procesos: (1) Anota título nuevo y (2) Realiza alta de surtido. Se oculta el almacén de datos SURTIDOS.

- *Control de Cuentas.* Agrupa a los siguientes procesos: (1) Realiza movimiento contable y (2) Realiza reporte de ventas. Se oculta el almacén de datos ESTADOS DE CUENTAS.

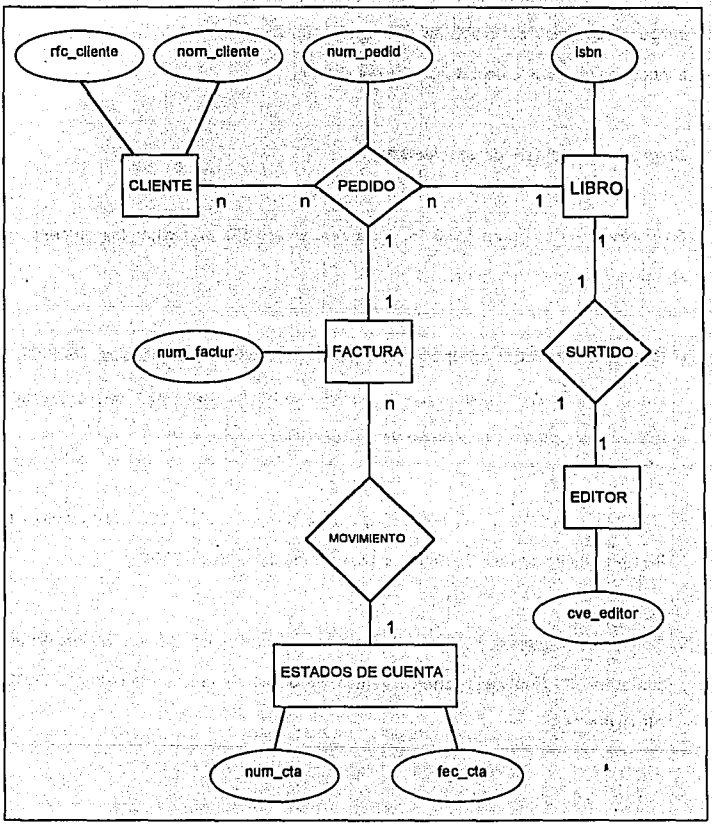


Figura II.3.3.13 Diagrama entidad - relación del sistema de pedido de libros.

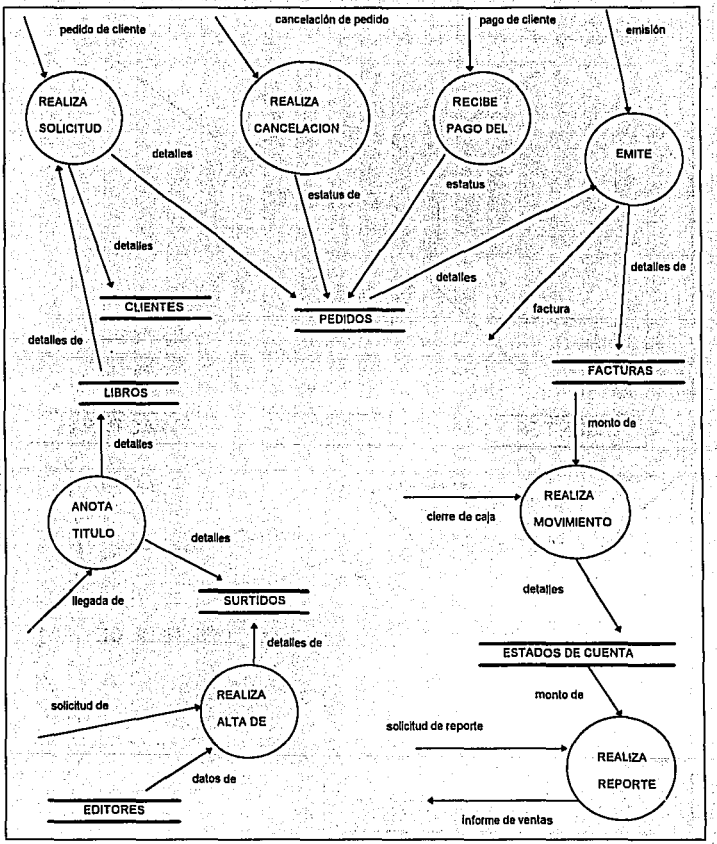


Figura II.3.3.14 DFD preliminar del Sistema de Pedido de Libros.

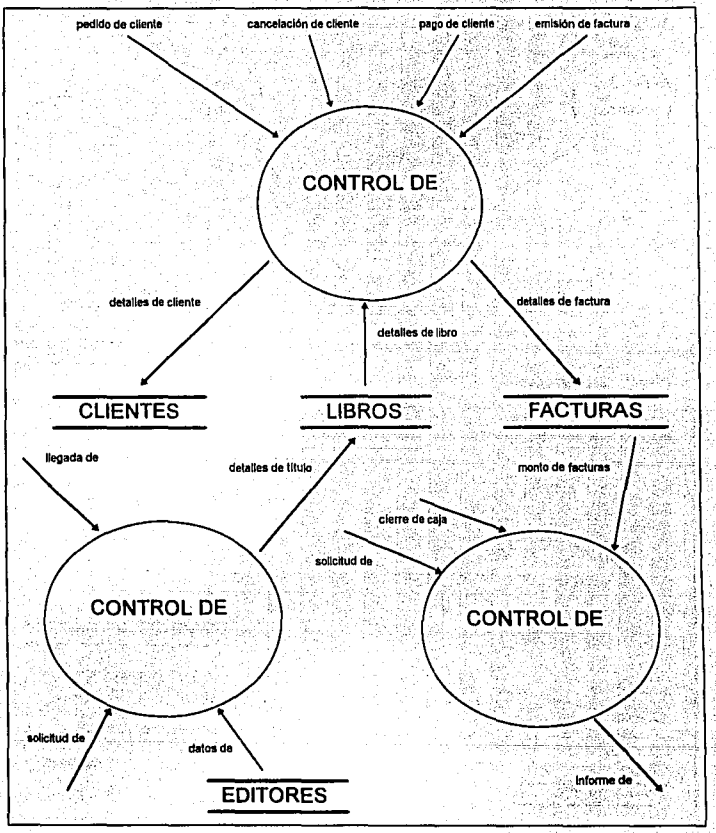


Figura II.3.3.15 DFD balanceado del Sistema de Pedido de Libros.

II.3.4 TECNICA GANE - SARSON

El enfoque para análisis de sistemas de Gane-Sarson fue planteado a finales de los setentas. Este también utiliza las herramientas de modelado antes descritas.

En el enfoque Gane-Sarson también se desarrolla un diagrama de contexto. En donde a partir del único proceso que se tiene, se procederá directamente a un DFD de nivel superior (conocido como figura 0), en donde cada proceso representa un subsistema principal.

Cada proceso de la figura 0 se parte a continuación en figuras de nivel inferior, y cada proceso de las figuras de nivel inferior se parte aún más, etc., hasta haber alcanzado el nivel de un proceso "atómico" que no requiere de mayor descomposición. Esto se ilustra en la figura II.3.4.1

Esta metodología implica construir un sistema top-down por refinamientos sucesivos.

Primero se produce un diagrama de flujo de datos de todo el sistema, entonces se desarrollan otros diagramas de flujos de datos en forma más detallada, a continuación se definen las estructuras de datos y los procesos lógicos, creándose de esta forma, un diseño de estructura modular.

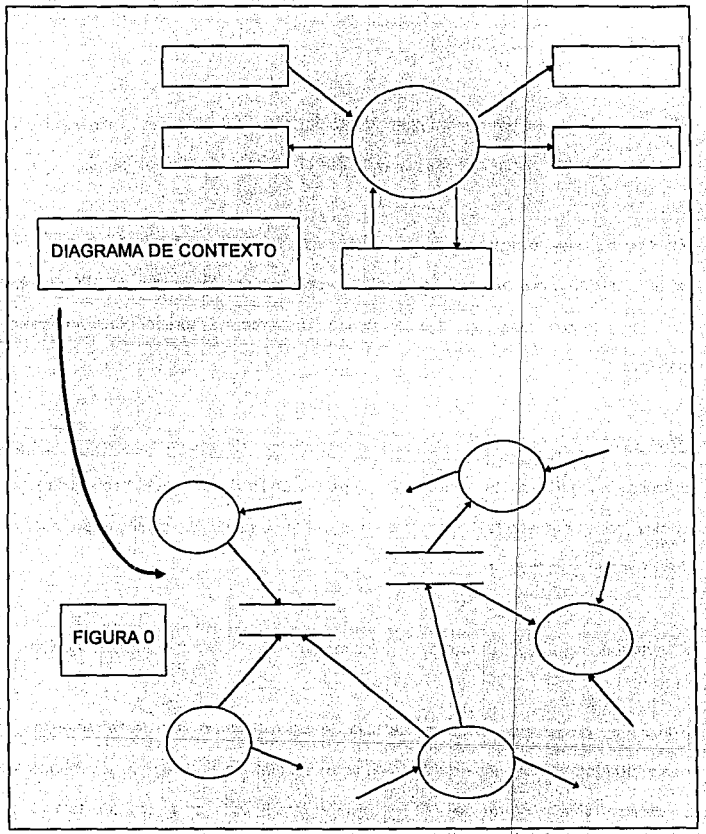


Figura II.3.4.1 Desarrollo descendente del modelo de comportamiento.

Con la estructura anterior, el sistema se analiza en forma top-down, se diseña en forma top-down, se desarrolla en forma top-down y se prueba en forma top-down.

Además, debe tomarse en cuenta que un buen desarrollo involucra iteraciones. Esto significa que se debe refinar el modelo lógico y el modelo físico del sistema en cada versión del diseño.

Los retos más comunes que se tiene en el enfoque Gane-Sarson son:

- Pueden existir ambigüedades para plantear la figura 0. Esto se presenta cuando el sistema es grande y complejo, o cuando se tiene más de un analista viendo el diagrama de contexto.

- La partición de figuras suele ser arbitraria y no ajustarse a los parámetros marcados por los requerimientos de la lista de eventos.

II.4 TURBO PASCAL 7.0 VENTAJAS Y DESVENTAJAS.

El lenguaje de programación PASCAL fue desarrollado al final de los años 60 por el Profesor Niklaus Wirth en el Eidgenössische Technishe Hochshule, de Zürich, Suiza (Y recibe el nombre del filósofo del Siglo XVII, Blaise Pascal). Su objetivo fue producir un lenguaje que incluyese un número pequeño de conceptos fundamentales de programación, que fuera apropiado para enseñar la programación como disciplina lógica y sistemática, y también que pudiera ser implantado en forma eficiente en la mayoría de las computadoras. Su éxito en conseguir esta meta puede ser medido por el rápido y vasto incremento en el uso de PASCAL, como un lenguaje para la enseñanza de los principios de programación de computadoras y como un lenguaje práctico para escribir programas de sistemas y aplicaciones.

Antes del Pascal, a la mayoría de los estudiantes se les enseñaba FORTRAN, lenguaje de computadora mucho más antiguo y no estructurado. El profesor Wirth creyó que muchos de los errores más comunes en la programación podrían evitarse si se utilizaba un lenguaje estructurado por bloques que tuviera un flexible reconocedor de tipos. Aunque Pascal fue inicialmente desarrollado como un lenguaje educativo, su facilidad de uso y su fuerte estructuración le permitió ganar muchos conversos que habían programado en otros lenguajes. Como Pascal empezó a ser usado crecientemente en el desarrollo de las aplicaciones comerciales, se necesitaban entonces sucesivas ampliaciones. Antes del Turbo Pascal, cada compilador soportaba

el desarrollo de un ligero conjunto de ampliaciones, las cuales dificultaban la transportabilidad. Sin embargo, debido a que el Turbo Pascal había surgido como el estándar, muchas de las ampliaciones podían usarse libremente.

Con la presentación de Turbo Pascal en 1983, Borland International revolucionó la forma en que los programadores de microcomputadoras se relacionaban con su mundo. En un paquete sorprendentemente pequeño y barato, Turbo Pascal proporcionó a los programadores una poderosa plataforma de desarrollo de software que combinaba un editor, un compilador y un enlazador en una sola unidad integrada (ver figura II.4.1). Los educadores estuvieron entre los que inicialmente se convirtieron al Turbo Pascal. Atraídos por el cómodo y amigable ambiente que la microcomputadora llevó a los estudiantes, éstos estaban deleitados de encontrar un producto de software que fuera tan poderoso como su contraparte de mainframe. En efecto, Turbo Pascal rápidamente se convirtió en un estándar entre los educadores para utilizarlo en cursos de introducción.

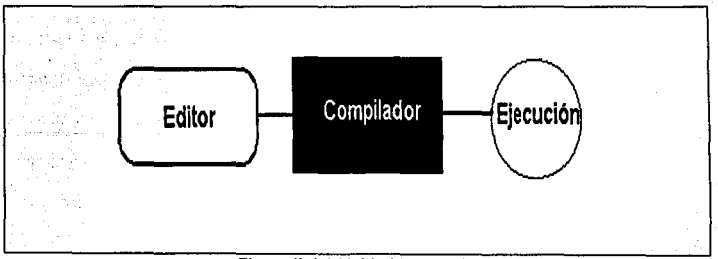


Figura II.4.1 Unidad Integrada

Durante su relativamente breve existencia, Turbo Pascal ha pasado por varias encarnaciones. Desde la perspectiva del educador, muchas de las adiciones al Turbo Pascal original, como el soporte a gráficas (versión 3.0), compilación separada (versión 4.0), debugger integrado (versión 5.0) y extensiones para la orientación a objetos (versión 5.5) han sido bienvenidas. Al mismo tiempo, el ambiente de programación actual (ambiente de desarrollo integrado o IDE, en terminología Borland) ha evolucionado.

El ambiente original era fácil de utilizar, pero limitado. El programador podía crear y correr un programa en Pascal, pero poco más. Turbo Pascal 4.0 añadió un ambiente basado en menús que proporcionaban muchas más opciones. Mientras la mayoría de las adiciones eran útiles, el ambiente basado en menús era difícil en el deseo individual de aprender como utilizar el sistema. Esto era especialmente cierto cuando ese individuo era un programador principiante, con poca o ninguna experiencia en computación.

El Pascal, que fue inventado como un lenguaje de aprendizaje, se ha convertido en un lenguaje de propósito general gracias a Borland. Las ampliaciones que hizo Borland se agrupan en dos categorías: sentencias del lenguaje (palabra clave) y procedimientos predefinidos. El Turbo Pascal 6.0 introduce el soporte del ratón y una forma expandida del ambiente basado en menús que caracterizaron a las versiones de la 4 a la 5.5. Todavía están presentes las mismas características, pero algunas han sido

modificadas o expandidas; algunas de las nuevas características pueden solamente ser accesadas mediante el ratón.

Cada nueva versión de Turbo Pascal también ha colocado requerimientos adicionales en el ambiente. Turbo Pascal 1.0 se distribuyó en un solo disquete de 360K y requería solamente 39K de memoria. Turbo Pascal 6.0 se distribuye en forma comprimida en cuatro disquetes de 3 pulgadas de alta densidad que, cuando se desempacan ocupan cerca de 3 megabytes. Aunque mucho del material incluido es opcional, y se presenta para ilustrar detalles y técnicas especiales, el propio compilador anda sobre los 323,000 bytes, y su archivo de ayuda sobre los 641,000 bytes. Esta claro que es casi obligatorio tener un disco duro o un ambiente de red para utilizar la versión 6.0.

EL PASCAL COMO LENGUAJE ESTRUCTURADO

El Pascal es un lenguaje estructurado con algunas similitudes con el Algol y C. La característica que distingue a un lenguaje estructurado es su capacidad para *compartir programas y datos*. Esto significa que puede separar y almacenar del resto del programa toda la información e instrucciones necesarias para realizar una tarea específica. Generalmente, la compartición se realiza mediante subrutinas, llamadas frecuentemente *subprogramas*, con *variables locales*, las cuales son temporales.

En este sentido, es posible escribir subrutinas tales que lo que sucede en ellas no causará efectos en otras partes del programa. El excesivo uso de variables globales

(variables conocidas por la totalidad del programa) puede dar lugar a arrastrar errores en un programa al no poderse detectar por sus efectos locales. En Pascal, todos los subprogramas son funciones concretas o procedimientos. Las funciones y procedimientos son los bloques que constituyen el Pascal. Una tarea específica en un programa se puede definir y programar separadamente en funciones y procedimientos. En Pascal, usando bloques de programa, se crea un programa estructurado.

OBJETIVOS DE LA PROGRAMACION

En la preparación de un programa un programador puede tener que escoger entre las soluciones alternativas en muchos puntos. Cada alternativa debe ser hecha para satisfacer los objetivos y restricciones de la tarea de programación particular.

El uso de una computadora para una labor particular implica tres pasos esenciales:

(a) especificar la labor que el computador realizará, en términos de los datos de entrada que serán suministrados y los datos de salida o resultados a ser producidos;

(b) planear un *algoritmo* o secuencia de pasos, por los cuales la computadora pueda producir la salida requerida a partir de la entrada disponible;

(c) expresar este algoritmo como un programa de computadora en un lenguaje de programación tal como PASCAL.

El paso (a), la especificación, no es normalmente considerado como parte del proceso de programación pero una especificación precisa es un prerrequisito para un programa exitoso.

Ha sido práctica común en el pasado separar los pasos (b) y (c), definiendo primero el algoritmo en una notación conveniente para su diseño, y luego trasladando o codificando este diseño en el lenguaje de programación escogido. Sin embargo, PASCAL proporciona una notación que puede ser usada tanto para el diseño como para la codificación final del programa requerido. Con PASCAL, por tanto, los pasos (b) y (c) no son usualmente separados, sino unidos como un proceso continuo de diseño/programación.

En principio, una vez que el programa de computadora ha sido escrito, la labor de programador está completa, ya que la ejecución de este programa por la computadora podría producir los resultados requeridos. En la práctica, puesto que la tarea a ser llevada a cabo por la computadora es compleja, y la capacidad humana del programador es limitada, el primer programa escrito puede no producir los resultados requeridos. El programador, por tanto, entra en un ciclo de revisión y corrección de su programa hasta que esté satisfecho de cumplir con su especificación completamente. Este proceso de detección y corrección de errores en un programa es conocido como limpieza. La limpieza es comúnmente realizada corriendo el programa en el computadora con datos de prueba adecuados.

En un lenguaje de "alto nivel" tal como PASCAL el programa es expresado como una secuencia de pasos elementales que son convenientes para el programador. Igualmente el programa es preparado en una forma que es cómoda de generar por el programador - como una pieza de texto escrito o impreso en papel, perforado en una secuencia de tarjetas, o digitado en un teclado terminal del computador.

Sin embargo, el programa que el procesador del computador ejecuta debe ser expresado como una secuencia de las operaciones de "bajo nivel" mucho más simples disponibles en el procesador, y debe ser guardado en la memoria de la computadora como una secuencia cada una de las cuales es inmediatamente ejecutada por el procesador. La preparación de un programa en esta forma es una tarea extremadamente tediosa y de mucho cuidado para un programador humano.

Afortunadamente, sin embargo, la traslación de un texto de programa expresado en un lenguaje de alto nivel a una secuencia equivalente de instrucciones ejecutables por el procesador dentro de la memoria de la computadora es ella misma una labor de rutina que puede ser llevada a cabo por un programa de computadora. Tal programa es suministrado por cada lenguaje de alto nivel que pueda ser usado en una computadora, y es conocido como el *compilador* para ese lenguaje.

Así un programa escrito en un lenguaje de alto nivel en forma de texto es primero entrado a la computadora como datos para una ejecución del programa compilador de

ese lenguaje. El compilador produce un programa ejecutable equivalente en la memoria del computador, el cual puede luego ser ejecutado o corrido para producir el efecto deseado. La figura II.4.2 indica este proceso en dos etapas en forma esquemática.

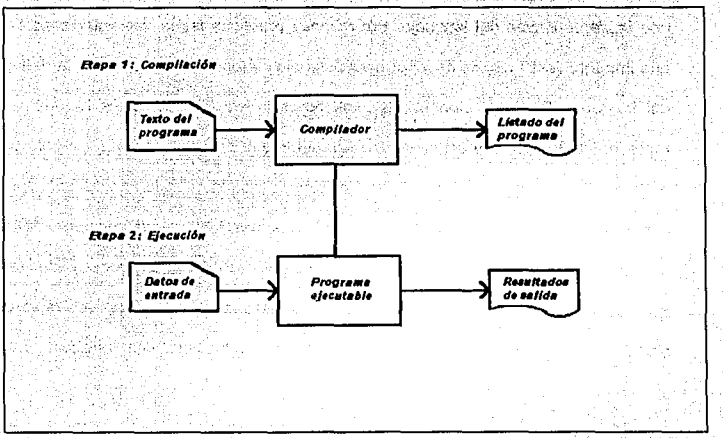


Fig. II.4.2 Etapas de la corrida de un programa.

Una ventaja de usar un lenguaje de programación de alto nivel, tal como PASCAL es que un programa escrito en el lenguaje puede ser usado en cualquier computadora para el cual haya sido provisto un compilador del lenguaje. En principio el lenguaje, y

por tanto los programas escritos en él, son independientes de la computadora presente - se dice que son *independientes de la máquina*.

En la práctica están disponibles gran número de computadoras muy diferentes. La provisión de un compilador para un lenguaje dado y una computadora dado es llamada una *implementación* del lenguaje. Por razones prácticas las implementaciones tienen que imponer restricciones de lenguaje adicionales a las especificadas por la definición del lenguaje. De igual manera las implementaciones algunas veces requieren una representación del texto del programa ligeramente diferente de aquella descrita por la definición del lenguaje. El estándar PASCAL identifica tales aspectos del lenguaje como *siendo definidos por la implementación* o *dependientes de la implementación*. Al preparar ó usar un programa para una implementación dada, un programador tendrá que asegurarse de que el programa se ajuste a cualesquiera requerimientos particulares de esa implementación, así como a las reglas misma del lenguaje.

Además de la capacidad para compilar un programa, una implementación puede proporcionar otras ayudas para el programador. Por ejemplo, la detección de un error de corrida o lógico en un programa no necesariamente identifica su causa. Las implementaciones pueden suministrar ayudas específicas para auxiliar al programador en tal identificación.

II.5. VISUAL BASIC 3.0 VENTAJAS Y DESVENTAJAS

Los primeros lenguajes de programación fueron diseñados en los años 50 y se crearon, fundamentalmente, para resolver complejos problemas matemáticos. Eran bastante complejos para la gente normal, pero ello no representaba un grave problema porque entonces los ordenadores sólo se encontraban en las principales instituciones investigadoras. Sin embargo, la gente se dio cuenta de que la tecnología informática podía ser útil para muchas más cosas que las matemáticas, y los ordenadores empezaron a hacerse habituales en las empresas y en las universidades. A medida de que más gente comenzaba a utilizar los ordenadores, los esotéricos y complicados lenguajes de programación llegaron a suponer algo más que un obstáculo.

Como solución, a principios de los años 60 se desarrolló un lenguaje denominado BASIC en el Dartmounth College. La versión original del BASIC (acrónimo de *Beginner's All-purpose Simbolic Instruction Code*) era un lenguaje muy simple, diseñado especialmente para hacer que resultase sencillo aprender a programar. Una generación entera de programadores dio sus primeros pasos con el BASIC y los uso para escribir una impresionante variedad de programas.

La simplicidad del BASIC lo hizo pequeño, y el tamaño era importante cuando los ordenadores también comenzaban a quedarse pequeños. El MITS Altair, en plena

revolución de los microordenadores, apareció en 1975. Bill Gates y Paul Allen, cofundadores de Microsoft, aceptaron el reto de desarrollar una versión del BASIC para el Altair que funcionase en los 4 kilobytes de RAM de los que disponía ese ordenador. Esa versión del BASIC ha llegado a convertirse en el producto más usado de la industria de los ordenadores personales.

Con el transcurso de los años, este lenguaje de programación se ha mejorado y desarrollado. Cuando los primeros micros le abrían el camino al PC de IBM, el GWBASIC de Microsoft estableció el estándar. Más adelante, la demanda de un software más rápido, pequeño y fácil de usar condujo al desarrollo del Microsoft QuikBasic. QuickBasic llevó al BASIC a la primera línea de la tecnología de los lenguajes de programación de los años 80, pero hubo un cambio aún más importante en el horizonte: el interface gráfico de usuario (GUI).

Con el advenimiento de Microsoft Windows, los usuarios de PC pudieron trabajar en un entorno intuitivo y gráficamente rico. Un interface gráfico de usuario permite que las aplicaciones sean fáciles de aprender y de usar. En lugar de aprender a escribir largos comandos, el usuario, simplemente, selecciona una opción (o comando) de un menú con un "clic" de un botón del ratón. Diversas ventanas en la pantalla permiten que el usuario pueda ejecutar más de un programa a la vez. Cuando un programa necesita información o que el usuario tome decisiones aparecen los cuadros de diálogo.

Aunque este entorno es maravilloso para el usuario, la vida se volvió de repente muy complicada para los programadores. Ahora tenían que crear ventanas, menús, fuentes, cuadros de diálogo y una multitud de elementos, incluso para el programa más simple. Por ello, cuando se presentó Microsoft Windows, los programadores se sintieron a la vez excitados y deprimidos - excitados porque Windows les proporcionaba una plataforma para escribir aplicaciones gráficas y agradables para el usuario; deprimidos porque su trabajo se hacía mucho más complicado.

Un sencillo programa que mostrase un mensaje en la pantalla podía escribirlo en cuatro líneas un programador que trabajase con MSDOS. Un programa similar para Windows requería dos o tres páginas de código e implicaba aprender a controlar fuentes, menús, ventanas, memoria y otros recursos del sistema. Pero las ventajas de Windows para el usuario final eran incuestionables, y la gente empezó a comprar programas escritos para Windows en grandes cantidades. Así que a los programadores profesionales no les quedó más remedio que empezar a escribir páginas y más páginas de códigos.

Muchos creyeron que Windows representaba el fin de la programación aficionada. En el mundo de MSDOS, los profesionales de áreas no relacionadas con la informática podían (en líneas generales) aprender lo suficiente de programación como para escribir aplicaciones sencillas que les ayudasen en su trabajo, les ahorrasen tediosos cálculos u organizasen los datos rápidamente. Pero, ¿podía alguien hacer eso mismo

en Windows, donde los requerimientos de programación eran tan complicados, incluso para las aplicaciones más sencillas?

La respuesta surgió en 1991 cuando Microsoft presentó Visual Basic. El sistema de programación Visual Basic sortea la complejidad de Windows de una manera realmente espectacular. Combinando las probadas posibilidades del lenguaje Basic con herramientas de diseño visual proporciona simplicidad y facilidad de uso, sin sacrificar prestaciones o las características gráficas que hacen de Windows un entorno tan apetecible en el cual trabajar. Los menús, fuentes, cuadros de diálogo, campos de texto con desplazamiento y todo lo demás se diseñan con facilidad, y esas características no requieren más de unas pocas líneas de programa para controlarlas. Ver Figura II.5.1

Visual Basic es también uno de los primeros lenguajes de programación que admite la programación llamada "orientada a eventos", un estilo de programación especialmente adaptado a los interfaces gráficos de usuarios. Tradicionalmente, la programación ha estado muy orientada al proceso, al paso a paso, de manera muy similar al de una receta: batir los huevos, añadir la leche, mezclar con el azúcar, hornear durante 20 minutos. Uno de los inconvenientes de este estilo consiste en que la persona que escribió la receta (el programa) es la que determina qué hay que hacer en cada momento. Eso puede ser aceptable para cocinar galletas, pero en los modernos

programas de ordenador el objetivo es que sea el usuario el que determine en cada momento qué es lo que quiere hacer.

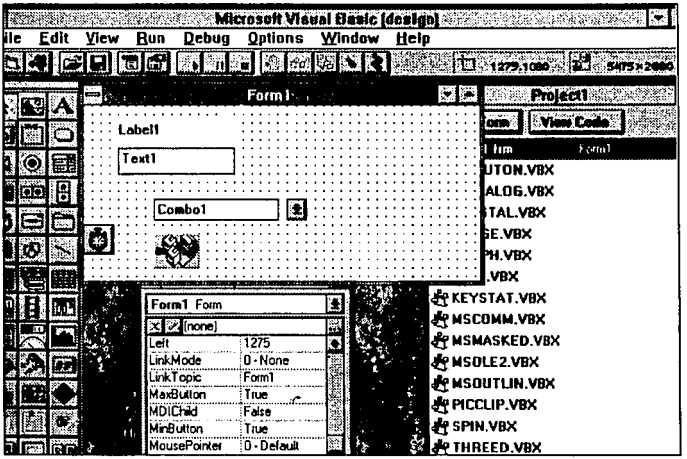


Figura II.5.1 Visión del ambiente de desarrollo de Visual Basic.

Y eso es exactamente lo que proporciona la programación orientada a eventos. En lugar de escribir un programa, el programador escribe un programa que responde a las acciones del usuario: elegir un comando, hacer clic en una ventana, mover el ratón. En vez de escribir un gran programa, el programador crea una aplicación que es realmente una colección de microprogramas que cooperan entre ellos y que se ejecutan a raíz de eventos iniciados por el usuario. Y, con Visual Basic, una aplicación así se puede escribir con una rapidez y facilidad sin precedentes. Ver figura II.5.2

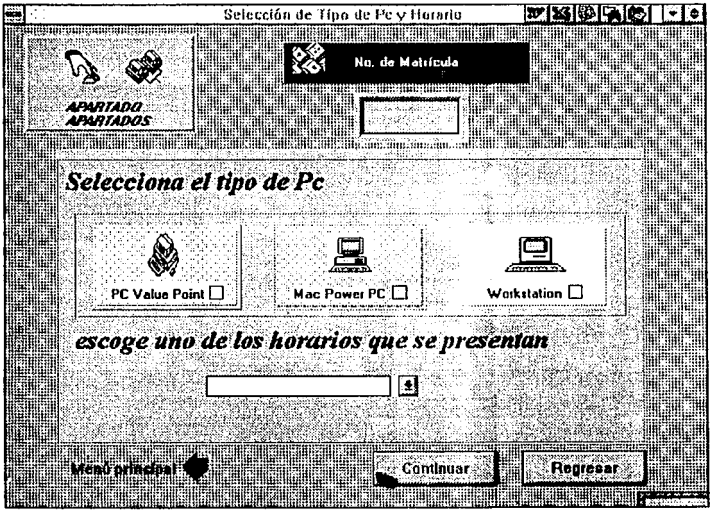


Figura II.5.2 Ventana desarrollada con las herramientas de Visual Basic.

El lanzamiento inicial de Visual Basic se convirtió en un éxito espectacular, vendiéndose decenas de miles de copias y ganando premios en la mayoría de la principales revistas de informática. A finales de 1992 se lanzó la versión 2 de Visual Basic, la cual ofrecía nuevas características y posibilidades.

Visual Basic es un poderoso sistema de programación gráfica que le permite crear aplicaciones Windows reales con código BASIC. Este sistema es un tremendo avance

para el desarrollo de programas, por que combina la sintaxis simplificada de BASIC y GWBASIC con la estructura de programación de QBASIC y QuickBasic. Visual Basic proporciona las herramientas necesarias para crear con facilidad los mismos elementos gráficos que son comunes en la mayor parte de las aplicaciones Windows. La Figura II.5.3 muestra un ejemplo de una aplicación Windows creada con Visual Basic.

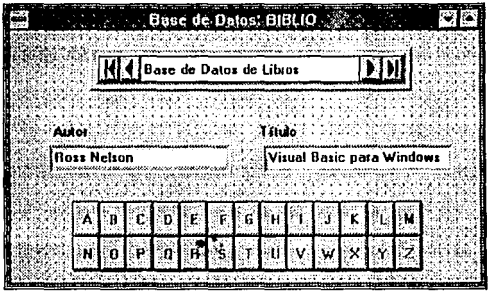


Figura II.5.3 Ventana creada con Visual Basic.

El sistema de programación Visual Basic le permite crear *objetos*, establecer y cambiar sus propiedades y, posteriormente, asignarles un código BASIC funcional. La filosofía de programación Visual Basic consiste, primero, en crear objetos como ventanas, iconos y menús y después elaborar procedimientos que sean llamados por cada uno de estos objetos. Esto difiere del método tradicional de elaboración de un programa, en el cual existen estructuras para controlar el flujo del programa de un procedimientos a otro de manera lógica hasta que el programa termina.

Cualquier programador experimentado se sorprenderá de lo sencillo que es escribir aplicaciones Windows *visualmente*, esto es, crear objetos y después escribir los procedimientos que los activan en BASIC. La programación con objetos es un método flexible y conveniente de escribir programas Windows. Por ejemplo se puede elaborar el código para un objeto ya creado y después hacer copias del mismo con el código asignado; no se tiene que escribir el código otra vez.

Visual Basic se puede definir como un *sistema de programación orientada a objetos*, en donde se crean objetos, llamados *formas y controles*, que hacen que su aplicación funcione.

Con el lanzamiento de Visual Basic 3, el producto se ha convertido en un sistema de programación maduro, al que se le han añadido muchas herramientas de programación de gran potencia. La versión 3 incluye las características nuevas que se muestran a continuación:

- Prestaciones mejoradas.
- Una herramienta de creación de bases de datos.
- Acceso visual a datos mediante el Control Data, de manera que se puedan crear aplicaciones que examinen los datos sin tener que escribir el código.
- Un nuevo control OLE (enlace e incrustación de objetos) que permite la edición "in situ".

- Un conjunto de cuadros de diálogo estándar que se ocupa de las tareas más habituales de interfaces con el usuario.
- Las posibilidad de crear menús emergentes (popup) en cualquier posición de la aplicación.

Una de las ventajas de utilizar Visual Basic es que se pueden desarrollar aplicaciones complejas y poderosas que los usuarios pueden correr como programas independientes en Windows. La mayor ventaja de utilizar ambiente Windows es que proporciona una interfaz consistente y manejable para desplegar información en muchas aplicaciones diferentes que comparten controles similares. La filosofía de la interfaz Windows se basa en la utilización de metáforas visuales para realizar acciones y tareas. Al utilizar controles Windows como menús, iconos, barras de desplazamiento y cajas de diálogo, la información es presentada a los usuarios en un formato dinámico y visualmente interesante.

II.6 BTRIEVE VENTAJAS Y DESVENTAJAS

ACCESO A LA BASE DE DATOS: PANORAMA GENERAL

Localizar un elemento de información específico en la Base de Datos y presentarlo al usuario requiere varios niveles de programas para acceso a datos. Los detalles de esos niveles varían en los distintos sistemas y lo mismo la terminología, pero los principios son bastantes generales y pueden explicarse a grandes rasgos de la siguiente manera:

1. En primer término, el Sistema de Administración de Base de Datos, (DBMS; Data Base Manager System), decide cual registro almacenado se necesita y pide al administrador de registros que lo extraiga.

2. El manejador de registros decide cual página contiene el registro deseado y pide al manejador de disco que lea esa página. La página es la unidad de Entrada/Salida, es decir, la cantidad de datos transferidos entre el disco y la memoria principal.

3. Por último, el manejador de disco, determina la localización física de la página deseada en el disco y realiza la operación de Entrada/Salida necesaria.

En términos generales, pues, el DBMS percibe la Base de Datos como un conjunto de registros almacenados, el manejador de registros apoya esa percepción; el manejador de registros a su vez, percibe la Base de Datos como un conjunto de páginas y el manejador de disco apoya esa percepción; y el manejador de disco percibe al disco "como es en realidad".

MANEJADOR DE REGISTROS

Utiliza los recursos del manejador del disco de manera tal que el usuario puede percibir el disco como un conjunto de archivos almacenados, un archivo almacenado es el conjunto de todas las ocurrencias de un tipo de registro almacenado. Cada conjunto de páginas contendrá uno o más archivos almacenados.

Cada archivo almacenado se identifica mediante un nombre de archivo o identificador de archivo único y cada registro almacenado, a su vez se identifica mediante un número de registro o identificador de registro único.

Entre las operaciones que puede realizar el manejador de registros con los archivos almacenados están las siguientes:

- Leer el registro almacenado **r** del archivo almacenado **a**
- Reemplazar el registro almacenado **r** dentro del archivo almacenado **a**

- Añadir al archivo almacenado **a**, un nuevo registro y devolver el nuevo identificador de registro **r**
- Eliminar el registro almacenado **r** del archivo almacenado **a**
- Crear un nuevo archivo almacenado **a**
- Destruir el archivo almacenado y **a**

Estas operaciones primitivas de manejo de archivo permiten construir y manipular las estructuras de almacenamiento.

BTRIEVE

El administrador de registros Btrieve fue desarrollado originalmente por Sofcraft, una compañía contratada por Novell a mediados de los ochentas.

Btrieve es un sistema de administración de registros completo, que permite manipular un archivo indexado. Está diseñado para optimizar el manejo de datos, mejorando la programación; permite recuperar aplicaciones, insertar, actualizar o borrar cualquier registro por medio de valores claves o por métodos de acceso secuencial o aleatorio.

CARACTERISTICAS DE BTRIEVE

INDEXACION

Btrieve permite insertar, actualizar y borrar registros manteniendo automáticamente los archivos indexados.

Un índice es un tipo especial de archivo almacenado, en el cual cada entrada (registro), se compone de sólo dos valores; un dato y un apuntador, el dato es un valor de algún campo del archivo indexado, y el apuntador identifica un registro de ese archivo que tiene ese valor en ese campo; dicho campo se llama campo indexado.

La ventaja primordial de los índices es la agilización de la obtención de datos. Sin embargo existe una desventaja, hacen más lenta la actualización de los archivos.

Los índices se pueden utilizar de dos maneras distintas; en primer término, pueden servir para tener acceso secuencial al archivo indexado, donde secuencial significa en el orden definido por los valores del campo indexado, en segundo término pueden servir también para tener acceso directo a los registros individuales del archivo indexado con base en un valor dado del campo indexado.

Además de la conservación automática del índice, Btrieve soporta las siguientes características de indexado:

- Más de 119 segmentos claves por archivo.
- Agregar o borrar cualquier índice después de que un archivo ha sido creado.
- Numerosos tipos de datos para valores claves: entero, punto flotante, fecha, tiempo, decimal, moneda, lógico, numérico, cadena, y otros.
- Numerosos atributos clave; duplicado/no-duplicado, modificable/nomodificable, modo ascendente/descendente, etc.

ESPECIFICACIONES DE ARCHIVO

- Tamaño de archivo arriba de 4 billones de bytes (4 G)
- Numerosos registros limitados sólo por el tamaño límite del archivo.
- Definición consistente de archivos y administración de rutas independiente del ambiente operante.
- Estructuras de archivo consistente.

ADMINISTRACION DE MEMORIA

El caché es área de memoria que el Btrieve reserva para almacenar temporalmente las páginas que esta leyendo, una página es la cantidad de datos transferidos entre el disco y la memoria principal.

Cuando una aplicación requiere un registro, Btrieve primero verifica el caché para determinar si la página que contiene el registro deseado está lista en memoria, si es así, se transfiere el registro desde el caché hasta el buffer de datos de la aplicación.

Si la página no esta en el caché, Btrieve lee la página desde el disco hacia el interior del caché, transfiriendo el registro requerido a la aplicación.

Si cada buffer caché esta lleno cuando Btrieve necesita transferir una nueva página en la memoria, un algoritmo LRU (Least Recently Used; la de menor uso reciente), determina cual página se puede sobrescribir. El algoritmo LRU reduce el tiempo de procesos manteniendo la referencia de las páginas más recientes en memoria.

Cuando se inserta o actualiza un registro, Btrieve primero modifica la correspondiente página en el caché y después la escribe en el disco. La página modificada permanece en el caché hasta que el algoritmo LRU determina que esa página se puede sobrescribir.

Generalmente un caché grande, mejora el funcionamiento, porque permite más páginas en memoria dispuestas al mismo tiempo. Btrieve permite especificar la cantidad de memoria reservada para la Entrada/Salida de los buffers caché.

SEGURIDAD

Se refiere a la protección de los datos contra una relevación alteración o destrucción; implica asegurar que los usuarios están autorizados para llevar a cabo lo que tratan de hacer. Btrieve mejora la seguridad de los datos asignando nombres propios a los archivos y especificando ciframiento y desciframiento de datos. El ciframiento consiste en que los datos pueden almacenarse físicamente en el disco o transmitirse a través de las líneas de comunicación, en forma codificada o cifrada, y cualquier persona que intente obtener acceso a ella a través de canales distintos de los oficiales, sólo detectará una mezcla ininteligible de bits.

INTEGRIDAD

La mayor parte de los DBMS son sistemas para múltiples usuarios; es decir, son sistemas en los cuales se permite a cualquier cantidad de transacciones tener acceso a la misma Base de Datos al mismo tiempo. En estos sistemas, es necesario algún tipo de mecanismo de control de concurrencia. Btrieve usa las siguientes técnicas para soportar accesos concurrentes y asegurar la integridad de los archivos; la integridad

se refiere a la exactitud o validez de los datos, implica asegurar que lo que se trata de hacer con los datos es correcto.

Uno de los mecanismos de control de concurrencia es el bloqueo; el efecto del bloqueo es "bloquear el acceso de otras transacciones" al objeto y en particular evitar que lo modifiquen. Así, la primera transacción puede realizar su procesamiento con toda confianza, pues el objeto en cuestión permanecerá en un estado estable mientras esa transacción lo deseé. Btrieve utiliza simples y múltiples bloqueos en dos niveles, el primero para registro y el segundo para archivo.

Uno de los problemas que puede causar el bloqueo es el llamado bloqueo mutuo, es una situación el la cual dos o más transacciones están en un estado de espera simultáneo, y cada una espera la liberación de un bloqueo por parte de la otra para poder continuar. En un ambiente de servidor, Btrieve detecta el bloqueo mutuo.

Otras técnicas que utiliza Btrieve para asegurar la integridad de los datos son las siguientes: Crear archivos pre-imagen para almacenar imágenes de páginas de archivos antes de insertar, actualizar o borrar registros. Utiliza procesos de transacción que permiten mantener la consistencia durante actualizaciones múltiples de archivos.

SERVIDORES Y ESTACIONES DE TRABAJO.

Btrieve puede ejecutarse en un servidor o en cualquier estación de trabajo. La versión basada en estación de trabajo ejecuta todo el proceso en ésta, accesa todos los archivos por medio de llamadas al Sistema Operativo. Estas llamadas al Sistema son ejecutadas localmente (por archivos locales) o redireccionadas al servidor (por archivos al servidor). Puede ejecutarse sobre varios sistemas operativos como: DOS, OS/2, Windows, Unix Ware.

En la versión basada en servidor, Btrieve se ejecuta en un servidor y un programa de petición Btrieve, corre en las estaciones de trabajo. Dicho programa puede ejecutarse en DOS, OS/2 y Windows, maneja la Entrada/Salida de datos entre las estaciones de trabajo y el servidor de la Red. El manejo de Entrada/Salida en el servidor se realiza con el archivo del Sistema.

Las aplicaciones escritas para la versión basada en Estación de Trabajo pueden correr en la versión para servidor y viceversa.

Además de proveer un sistema de administración de registros para aplicaciones de Estación de Trabajo, Btrieve también acepta llamadas desde otras aplicaciones de servidores.

Btrieve puede trabajar en un multiambiente de Red, como la figura II.6.1 lo muestra; cada uno de los tres diferentes ambientes de Estación de Trabajo pueden accesar a cualquier servidor de la Red. Sus respectivos programas de petición Btrieve las comunican con Btrieve NetWare cargado en el servidor deseado.

PROGRAMAS DEL SERVIDOR

El manejador de registros Btrieve NetWare utiliza dos programas de servidor; BTRIEVE.NLM y BSPXCOM.NLM. el primero se debe cargar en cada servidor que almacena archivos Btrieve, el segundo debe cargarse para accesar a Btrieve desde una Estación de Trabajo.

BTRIEVE.NLM es el programa manejador de registros que maneja las demandas Btrieve, el Shell del servidor y el módulo de comunicaciones de la Red.

BTRIEVE.NLM hace lo siguiente: Ejecuta todas las Entradas/Salidas de disco para los archivos Btrieve almacenados en el servidor donde residen. Distribuye y libera todos los bloqueos nivel registro y nivel archivo en el servidor donde residen. Mantiene un registro de todas la demandas Btrieve que resultan en cambios de un archivo.

BSPXCOM.NLM permite a las demandas Btrieve de diferentes ambientes operantes comunicarse con la versión servidor de Btrieve.

PROGRAMA DE PETICION BTRIEVE

El ***Programa de Petición*** debe ser cargado en cada estación de trabajo que haga demandas a Btrieve; es la vía de comunicación entre los programas de aplicación de las Estaciones de Trabajo y Btrieve. Tiene las siguientes funciones: Recibe las demandas Btrieve desde una aplicación y las despacha al servidor apropiado. Regresa los resultados de la demanda Btrieve a la aplicación.

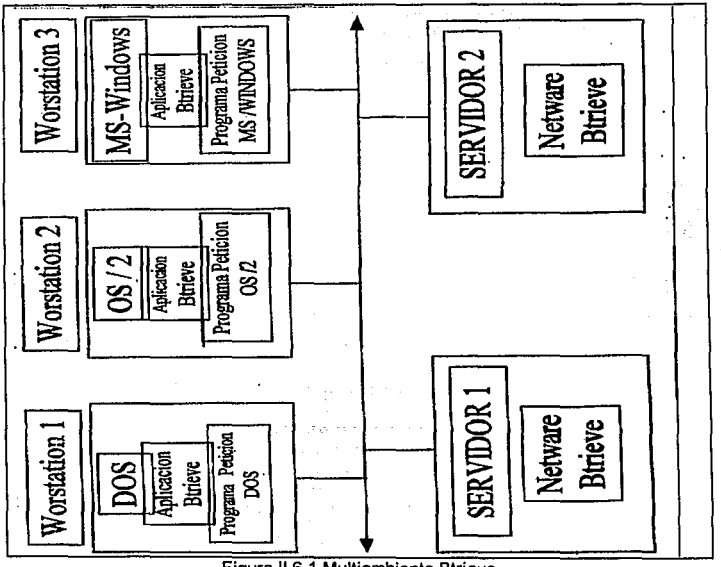


Figura II.6.1 Multiambiente Btrieve

ARQUITECTURA DE BTRIEVE

Los principales componentes de Btrieve para servidor son los siguientes:

- Administrador de registros basado en servidor
- Programa de comunicaciones
- Programa de petición para Estaciones de Trabajo
- Utilerías

Hay cuatro programas básicos en Btrieve NetWare:

- BSPXCOM.NLM
- BTRIEVE.NLM
- EL PROGRAMA DE PETICION
- BROUTER.NLM

BTRIEVE.NLM es una librería de funciones Btrieve que reside en el servidor, las aplicaciones de Estación de Trabajo y de servidor pueden accesar a Btrieve.

Las aplicaciones de Estación de Trabajo se comunican con el servidor a través del programa de petición y BSPXCOM.NLM.

BTRIEVE.NLM

Es una librería de funciones Btrieve, debe cargarse en cada servidor que almacene archivos Btrieve.

BSPXCOM hace llamadas a Btrieve en nombre de una estación de trabajo o un BROUTER remoto. Específicamente estas llamadas manejan las siguientes tareas:

- Manejo de todas las Entradas/Salida de disco para archivos Btrieve almacenados en un servidor.
- Distribución y liberación de todos los bloqueos nivel registro y nivel archivo.
- Registro de todas las demandas que resulten de cambios en un archivo.

BSPXCOM.NLM

Debe estar cargado en el archivo de acceso del servidor. Permite a las demandas originadas fuera del servidor, comunicarse con Btrieve.

Cuando Btrieve es accesado, la demanda puede originarse en cualquier servidor que tenga una copia de acceso a Btrieve, o desde una aplicación en una estación de trabajo.

Cuando la demanda es originada en un servidor, BSPXCOM recibe la demanda desde otra copia de BROUTER y hace la llamada de función de Btrieve apropiada para procesar la demanda.

Si la demanda es de una estación de trabajo, BSPXCOM recibe la demanda del Programa de Petición de la estación de trabajo y hace la llamada a la función apropiada de Btrieve para procesar la demanda.

Después de que la demanda ha sido procesada, BSPXCOM la empaqueta para transmitirla, ya sea a una copia de Programa de Petición en una estación de trabajo, o a una copia de BROUTER en cualquier servidor.

PROGRAMA DE PETICION BTRIEVE

El Programa de Petición debe estar cargado en cada estación de trabajo que haga demandas a Btrieve. Las aplicaciones de las estaciones de trabajo, se comunican con Btrieve mediante el Programa de Petición, el cual tiene las siguientes funciones:

- Recibe demandas Btrieve desde una aplicación y las despacha al Btrieve ejecutándose en el servidor apropiado por medio del BSPXCOM.
- Recibe el resultado de la demanda Btrieve desde BSPXCOM y las regresa a la aplicación.

BROUTER.NLM

Se carga en el archivo del servidor. Es una aplicación basada en servidor que permite a otras aplicaciones de este tipo cargarse en el servidor para comunicarse con Btrieve a través de BSPXCOM. BROUTER maneja las siguientes tareas:

- Proporciona acceso convencional basado en SPX a archivos Btrieve que residen en otro servidor en la Red.
- Mantiene un único sistema de código de identificación para cada aplicación por lo tanto los bloqueos, transacciones y otros mecanismos de control de acceso pueden trabajar sin conflicto a través de la Red entera.

UTILERIAS

Btrieve proporciona dos programas de servicio; el primero BCONSOLE que realiza el monitoreo de la actividad de cualquier archivo Btrieve en la Red. El segundo BROLLFWD es una utilería para restaurar archivos.

ACCESO A BTRIEVE

Los programas de Btrieve NetWare funcionan como si fueran una subrutina de una aplicación. Btrieve NetWare soporta los siguientes métodos de acceso:

- Una aplicación de estación de trabajo puede accesar a Btrieve mediante el Programa de Petición.
- Una aplicación basada en servidor puede accesar a Btrieve directamente en el mismo servidor por llamadas al punto de entrada de exportación.
- Una aplicación basada en servidor puede accesar a Btrieve en cualquier otro servidor por medio de BROUTER.

APLICACIONES DE ESTACION DE TRABAJO

Las aplicaciones que se ejecutan en una estación de trabajo se comunican con Btrieve por medio del Programa de Petición. Los siguientes pasos ilustran el flujo de control cuando una aplicación de estación de trabajo accesa a Btrieve:

1. La aplicación distribuye una demanda Btrieve usando una función de llamada.
2. Un código de interface proporcionado por Novell que se liga con la aplicación, hace la llamada al Programa de Petición.
3. El programa de Petición empaqueta la demanda en un mensaje de red, determina cual servidor la recibirá y enruta el mensaje al BSPXCOM que reside en el servidor.
4. BSPXCOM recibe el mensaje, valida los parámetros y entonces ejecuta la demanda haciendo llamadas a funciones Btrieve. Dependiendo de la naturaleza de la demanda, puede involucrar operaciones sólo de memoria u operaciones de Entrada/Salida de un dispositivo de almacenamiento.

5. BSPXCOM regresa el resultado de la operación al Programa de Petición de la Estación de Trabajo.
6. El Programa de Petición regresa los datos y el código de status apropiado a los parámetros de las variables de la aplicación en memoria y retorna el control a la aplicación.

Si se están usando múltiples servidores o una red interconectada, no todos los servidores tienen que estar en línea cuando empieza a trabajar el Programa de Petición de la Estación de Trabajo.

APLICACIONES BASADAS EN SERVIDOR

Una aplicación creada en ambiente NetWare recibe el nombre de Módulo cargable NetWare (NLM, NetWare Loadable Module). Un NLM puede hacer dos tipos de demandas Btrieve, local o remota. Una demanda local es procesada solamente en un servidor local, esto es, el servidor donde la demanda Btrieve es originada. Una demanda remota es procesada en alguna parte de un servidor remoto, esto es, un servidor diferente al servidor donde se originó la demanda.

Una aplicación puede usar ambos tipos, local o remoto. Ya sea que una demanda sea ejecutada localmente o remotamente, el proceso es transparente a la aplicación. Para que una demanda local tenga éxito, BTRIEVE.NLM debe estar cargado en el servidor

local. Si BROUTER.NLM esta cargado en el servidor local cuando la demanda se hace, la llamada a la función Btrieve puede ser despachada a través del BROUTER.

De otro modo, las llamadas al Btrieve pueden ser hechas directamente al punto de entrada de exportación de Btrieve. Para el éxito de una demanda remota, BTRIEVE.NLM y BROUTER.NLM deben estar cargados en el servidor local. BSPXCOM.NLM y BTRIEVE.NLM deben cargarse en el servidor remoto.

II.7 CONCEPTOS DE REDES

Una red de computadoras en su forma global se define como un grupo de computadoras interconectadas a través de uno o varios caminos o medios de transmisión, con el fin de intercambiar la información almacenada en cada una de ellas y permitir la utilización de los recursos computacionales de diferentes computadoras.

En forma general, los principales objetivos de las redes pueden ser enumeradas como sigue:

- Eliminar el desplazamiento de los individuos en la búsqueda de información y en el acceso a equipos de alto costo.
- Ofrecer transparencia al usuario por medio de compatibilidades técnicas en las terminales.
- Aumentar la capacidad de procesamiento y almacenamiento disponible por cada uno de los usuarios en un momento determinado.
- Proponer alternativas de enrutamiento para el transporte de la información en caso de fallas en los medios de transmisión.
- Ofrecer acceso a servicios universales de datos.

Existen muchos servicios ofrecidos por las redes, aunque los que sobresalen por su mayor uso son el correo electrónico (e-mail), la transferencia de archivos, el acceso a bancos de datos y los Servicios de Boletines Electrónicos (BBSs).

Las redes de computadoras son clasificadas en base a varios criterios, siendo el más importante la cobertura geográfica, por lo que se clasifican en: Redes de Area Local (LAN's), Redes de Area Metropolitana (MAN) y Redes de Area Amplia o Extendida (WAN). Por su velocidad de transmisión las redes pueden ser de banda de voz (hasta 19.2 Kbps), de banda estrecha (desde 56 Kbps hasta 1.5 Mbps), de banda ancha (desde 1.5 Mbps hasta 45 Mbps) pudiendo ser terrestres o sátelitales. Por su topología las redes locales se clasifican en: bus, estrella, física/bus lógico, anillo, estrella física/anillo lógico y malla.

REDES DE AREA LOCAL

Una red local se define como una interconexión de computadoras o máquinas mediante un medio de transmisión dentro de una distancia que no supere una decena de kilómetros. Son utilizadas en edificios de oficinas, plantas de ensamblado, universidades, centros de investigación, hospitales, etc. La información intercambiada es principalmente de datos aunque empiezan a aparecer redes locales para la transmisión de vídeo y redes soportando aplicaciones multimedia.

En el mundo existen más de un millón de redes locales , una etapa consecuente de este crecimiento será la posibilidad de interconexión de todas esas redes locales, poniendo al alcance de los usuarios una capacidad de comunicación multiempresarial.

Por ejemplo, el Tecnológico de Monterrey cuenta con más de 63 redes locales, todas ellas interconectadas. El servicio de más éxito es el servicio de correo electrónico y transferencia de archivos. Cada alumno y profesor del tecnológico, dentro de cada uno de los 26 Campus del Sistema ITESM en toda la República Mexicana tiene acceso a bibliotecas de Estados Unidos y Europa, así como acceso a la Red Mexicana de Universidades (REDMEX) que conecta una veintena de instituciones de alto nivel académico en el país.

REDES DE AREA METROPOLITANA

Una red metropolitana es esencialmente una red local muy grande que cubre una ciudad entera, suministrando el transporte de datos a grandes velocidades (del orden de 100 Mbps) utilizando fibra óptica. Típicamente una MAN conectará LANs de más baja velocidad a través de una ciudad o región, solucionando las limitaciones de ancho de banda. Para salvaguardar todos los datos transmitidos, las redes metropolitanas emplean mecanismos de autorecuperación para asegurar el grado más alto de disponibilidad y confiabilidad de la red. Las MANs son diseñadas de manera que el transporte es fácilmente compartido por muchos clientes.

Las aplicaciones más sobresalientes de las redes metropolitanas son: interconexión de LANs, interconexión de Conmutadores Privados de Voz (PBX), interconexión de computadoras de aplicaciones CAD/CAM y transmisión de vídeo.

El uso de sistemas de fibra óptica para realizar la transmisión asegura virtualmente que las MANs no lleguen a su máxima capacidad en un futuro cercano. Las redes metropolitanas pueden ser públicas o privadas. Un ejemplo de una red privada sería que una corporación conecte sus edificios que se encuentran a lo largo de una ciudad, para el intercambio de información de voz, datos e imagen y en la cual es propietaria de todos los medios que utiliza para su conectividad. Una MAN pública es una red hecha para ofrecer facilidades que son compartidas por muchas organizaciones.

Para la implementación de las MANs existen tres propuestas a nivel internacional: IEEE 802.6/6/DQDB, SMDS Y FDDI.

REDES DE AREA EXTENDIDA

Con el fin de interconectar computadoras en áreas geográficamente muy alejadas, o redes LAN y MAN, se requiere contar con mecanismos de acceso especiales que caen dentro del concepto "internetworking". La intercomunicación remota requiere emplear enlaces de microondas, fibra óptica, cable submarino y satélites. La redes WAN emplean conmutación de paquetes, inicialmente a baja velocidad (hasta 19,200 bps)

con el protocolo X.25. Sin embargo, el advenimiento de tecnologías más rápidas y los requerimientos de las nuevas aplicaciones (multimedia, por ejemplo), impulsan el empleo de nuevos protocolos como frame-relay y cellrelay, con los que se alcanzan velocidades de transmisión en el orden de Megabits/segundo.

Las aplicaciones de las WAN se pueden sintetizar en:

- Acceso a programas remotos
- Acceso a bases de datos remotos
- Facilidades de comunicación de valor agregado

Como ejemplos de redes WAN tenemos, TELEPAC, TRANSPAC, ARPANET, USENET, CSNET, BITNET, TYMNET.

Las redes locales son las herramientas de comunicación por excelencia que permiten intercambiar información y compartir recursos. La topología y su protocolo de acceso son los dos conceptos básicos que definen su funcionamiento. Las más difundidas en la actualidad son Ethernet y Token Ring, y su protocolo de acceso está diseñado para la transmisión de datos en ráfagas. Las redes del futuro serán aquellas capaces de transmitir tráfico multimedia.

Cuatro características básicas distinguen a las redes locales de las subredes de comunicación de área amplia. Primero, la extensión limitada de la subred (entre unos cuantos metros y algunos kilómetros). Segundo, las altas velocidades de transmisión empleadas (típicamente 10 Mbps). Tercero, la baja probabilidad de error durante la transmisión (entre 10^{-4} y 10^{-11}) y cuarto, el carácter privado de la subred.

COMPONENTES DE UNA RED LOCAL

EL CABLEADO

Según la forma geométrica resultante de conectar las computadoras o recursos físicos a la red se define la topología.

LOS METODOS DE ACCESO

Debido a que todas las computadoras están conectadas a un único medio físico (el cable), debe definirse el método por el cual se garantice que todos los usuarios conectados a la red tengan las mismas posibilidades de comunicarse a través de ella.

EL SISTEMA OPERATIVO DE LA RED

Define los conceptos lógicos que se manejan en la red, por ejemplo, la definición de los usuarios, grupos de usuarios, métodos de seguridad para el acceso a la información, etc.

TOPOLOGIA

La topología de una red se refiere a la forma en la que se conectan físicamente las computadoras (u otros dispositivos) a la red. Por razones económicas las redes locales utilizan topologías simples, a diferencia de las redes de área amplia que generalmente utilizan una topología en malla. Las tres topologías básicas de las redes locales son: bus, anillo y estrella.

TOPOLOGIA EN BUS

Consiste en un hilo único (bus) del cual se "cuelgan" cada una de las estaciones de la red. El bus debe pasar cerca de todas y cada una de las estaciones.

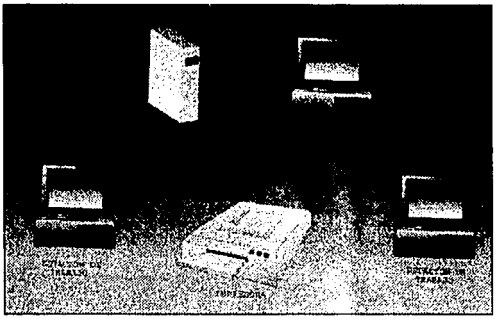


Figura II.7.1 Topología física en bus. Todas las estaciones "cuelgan" de un hilo que pasa cerca de ellas

VENTAJAS

- No existen elementos centrales de los que dependa toda la red, cuyos fallos dejarían inoperantes a todas las estaciones.
- El cableado es de bajo costo, tanto por los materiales que se emplean como por su reducida complejidad de instalación.
- El momento de conexión/desconexión de las estaciones no afecta al funcionamiento de la red.
- El envío de información entre estaciones es sencillo.

DESVENTAJAS

- Si se deteriora el cable se inutiliza la red por completo.
- Sólo se puede utilizar un medio de transmisión.

En la actualidad es la topología más utilizada. Cuando la red es de grandes dimensiones se suele tener una estructura de múltiples buses interconectados.

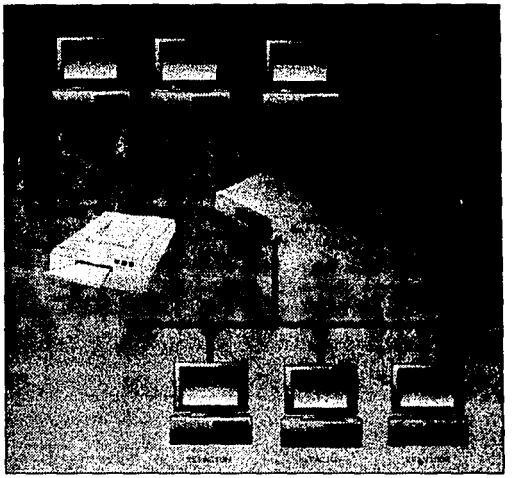


Figura II.7.2 Red de grandes dimensiones con estructura de múltiples buses.

TOPOLOGIA DE ANILLO

Se construye un anillo físico tendiendo un conductor, generalmente de pares de hilos, desde cada estación a la siguiente. La información suele circular en un sólo sentido

del anillo. Para que la información llegue a un nodo concreto debe pasar por todos los nodos anteriores, por lo que el envío de información a todas las estaciones resulta sencillo. A esta topología también se le conoce como "bucle". Para transmitir la información de un nodo a otro ésta se divide en paquetes que contienen la dirección del nodo que debe recibir la información.

VENTAJAS

- No existe dependencia de un nodo central.
- Es posible utilizar distintos medios de transmisión en diferentes sectores del anillo.
- El envío de información a todos los nodos es sencillo.

DESVENTAJAS

- El cableado es caro, tanto por los materiales que se emplean como por la complejidad de su instalación.
- Una anomalía en el cable provoca la caída de toda la red.
- La fiabilidad de la red depende de todas y cada una de las estaciones.
- Para añadir o retirar estaciones de la red es necesario detener la misma

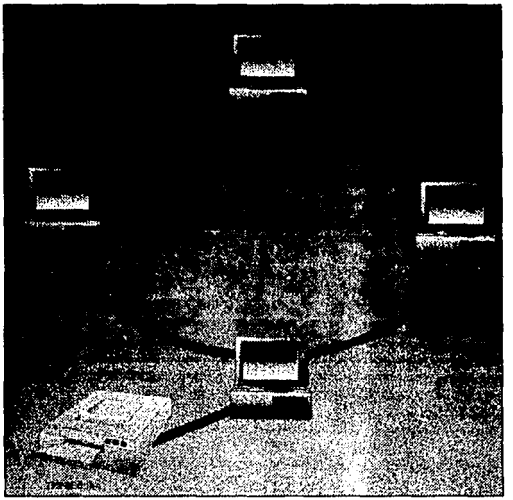


Figura II.7.3 Topología en anillo. Cada una de las estaciones se conecta con la siguiente, resultando la forma geométrica de un anillo.

TOPOLOGIA EN ESTRELLA

Todas las estaciones de la red se unen mediante cables, generalmente. La unidad de control, da turnos a las estaciones para utilizar la red. Este método se denomina polling. La unidad de control no tiene por que ser el servidor de ficheros, puede ser solamente un servidor de red, es decir, la unidad encargada de gestionar el tráfico de información a través de la red.

VENTAJAS

- El protocolo de comunicación reside en la unidad central, por lo que se reducen las tareas de las estaciones y por tanto su costo.
- Las estaciones pueden tener diferentes velocidades de transmisión, medios y protocolos.
- Las averías son fáciles de localizar y es muy sencillo añadir o eliminar estaciones

DESVENTAJAS

- La unidad de control central es un punto crítico. Si éste cae, toda la red cae.
- Para la instalación se requieren grandes cantidades de cable, ya que se debe unir cada una de las estaciones con la unidad central por lo que el costo es elevado.
- El controlador central limita la relación entre el número de estaciones y las velocidades de éstas.

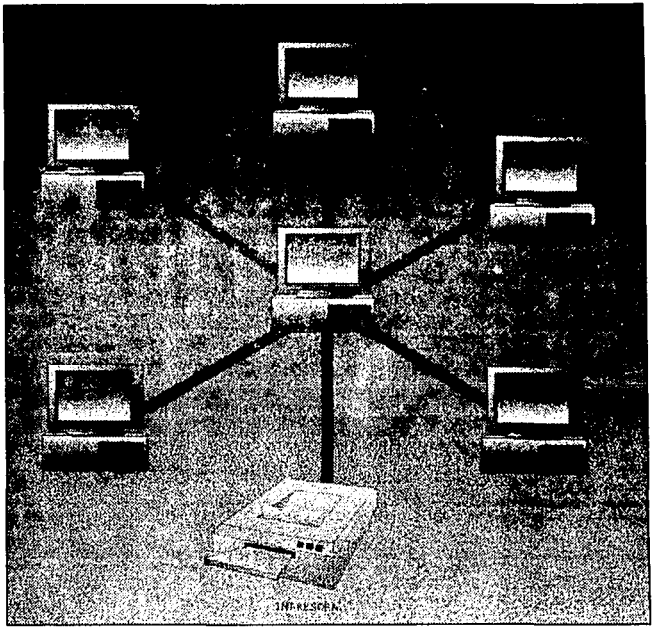


Figura II.7.4 Topología en estrella. Todas las estaciones de la red se conectan a una unidad de control central.

En la implantación real se utilizan estrellas jerarquizadas, es decir, cada rama se ramifica de nuevo.

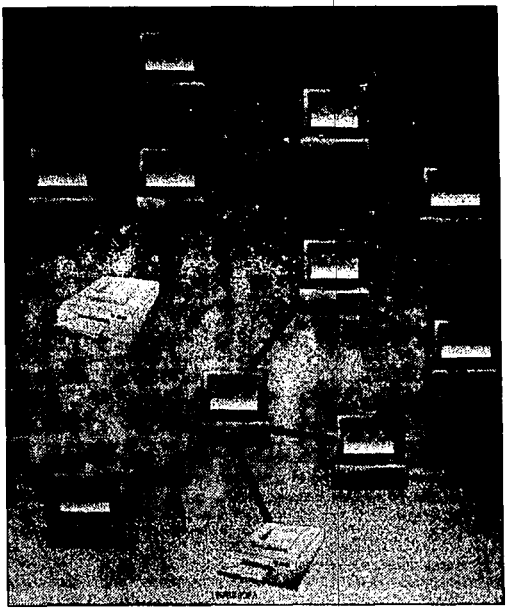


Figura II.7.5 En las estrellas jerarquizadas cada rama se ramifica de nuevo y forma una nueva red

TECNICAS DE TRANSMISION

Actualmente las técnicas de transmisión se dividen en dos grupos:

TRANSMISION DE BANDA BASE

La señal procedente de la estación que desea comunicarse se entrega a la red en forma digital, sin modulación y ocupando todo el ancho de banda disponible en el medio. Es decir, empleando toda la capacidad del cable para transmitir la información.

TRANSMISION DE BANDA ANCHA

La señal se modula y el ancho de banda disponible en el cable se divide en canales, por lo que el resto de los canales se pueden utilizar para otras comunicaciones de la red o para comunicaciones externas. Se podría decir que es como dividir el interior del cable en varios cables de menor sección y utilizar cada uno para un propósito. El inconveniente de utilizar banda ancha es que el hardware que se utiliza para esta técnica de transmisión tiene un costo muy elevado. Sin embargo, tiene la ventaja de que permite compartir los medios de transmisión con otros sistemas de comunicación, lo que la hace ideal cuando se tiene que instalar cables para sistemas de voz, vídeo y otros sistemas de datos.

MEDIOS DE TRANSMISION

Por último en lo al aspecto físico se refiere, hay que destacar los tipos de cables que se utilizan en la instalación de redes locales.

CABLES DE PARES TRENZADOS

Es el medio más barato. Tiene la gran ventaja de que en muchos de los edificios modernos ya está instalado para sus comunicaciones telefónicas, y se puede utilizar como medio de la red local.

El gran inconveniente que presenta es que son altamente sensibles a perturbaciones eléctricas del medio ambiente, por tanto la tasa de errores es alta y se deben reenviar los bloques de información, lo que se traduce en velocidades de transmisión muy limitadas.

CABLE COAXIAL

Es más caro que el anterior, pero tiene mejores características de transmisión de alta frecuencia, por lo que tiene la ventaja de ser menos sensible al ruido eléctrico, permitiendo velocidades de transmisión más elevadas.

En el cable coaxial un hilo central transporta la señal. El hilo está protegido con un aislante y una camisa de hilos conductores en forma de malla que actúan como un escudo contra el ruido eléctrico. Este tipo de cable es el que se emplea para la instalación de red local Ethernet.

FIBRA OPTICA

Se caracteriza por su altísima velocidad de transmisión, así como por su no menos elevadísimo costo, tanto de los materiales como de instalación. El estándar de las redes locales por fibra óptica es el llamado FDDI (Interface de datos distribuidos por fibra óptica) y esta normalizado por ANSI (Instituto nacional americano de normalización). Entre sus características más importantes destacan su velocidad de transmisión de 100 Mbps y topología de doble anillo redundante (que envía datos simultáneamente en dos direcciones), así como los 2 Kms de distancia máxima entre las estaciones y un máximo de 500 estaciones.

El medio es solamente la vía por la cual se transmite la información, por tanto se pueden construir redes locales basándose en radio, satélites y otros medios de comunicación.

METODOS DE ACCESO

Todas las estaciones están conectadas a un medio de comunicación único, dicho medio sólo puede ser utilizado por una estación emisora simultáneamente, por ésta razón, si en una red local una computadora transmite mientras otra se encuentra transmitiendo, la interferencia presente en el medio físico puede causar que una o ambas transmisiones resulten dañadas. Es por esto que las computadoras que se

conecten a una red local deben seguir un protocolo de acceso que controle el orden en el cual se realizan las transmisiones.

Dado que el tráfico generado por las diferentes computadoras conectadas a una red local no es continuo sino por ráfagas, el protocolo de acceso debe asignar de manera dinámica el uso de la red local a las computadoras. El protocolo de acceso permite a las computadoras transmitir información sobre la red local durante intervalos de tiempo limitados. Esta característica de los protocolos de acceso evita que una computadora monopolice el medio de transmisión durante periodos largos de tiempo y por lo tanto que otras computadoras sufran largas esperas antes de poder transmitir.

Los mensajes transmitidos por una computadora en una red local tienen un formato predefinido (diferente para cada protocolo de acceso) en el cual se incluyen las direcciones de las computadoras origen y destino del mensaje.

Los protocolos más populares son los siguientes:

CSMA/CD

Acceso múltiple con escucha de portadora y detección de colisiones. La transmisión a través de este protocolo se realiza pasando por las siguientes fases:

1. La estación que desea transmitir lee la posible información que circula por el cable para saber si alguna otra estación lo esta utilizando. En caso de ser afirmativo, debe esperar hasta que el medio quede libre y volver a intentar la comunicación.

2. Mientras la estación emisora está emitiendo su mensaje debe leer la información que circula por el cable con el fin de detectar si ha coincidido con alguna otra estación que intentaba comunicarse en ese instante. En caso afirmativo, el resultado de ambas comunicaciones sería ruido y se produce lo que se llama una colisión.

3. Proceso de interferencia. Cuando la estación emisora detecta que se ha producido una colisión transmite una pequeña interferencia. De esta forma se asegura que todas las estaciones conectadas a la red son informadas del estado de colisión, y que la información recibida es errónea, siendo necesario enviar de nuevo el mensaje.

4. Tiempos de espera. Cuando se produce una colisión en la red, en cada una de las estaciones emisoras que lo han provocado se pone en marcha un temporizador interno de distintos tiempos, de forma que se garantice que el reintento de comunicación no vuelva a coincidir. El algoritmo por el cual se determina el tiempo que debe esperar cada estación es generalmente, un generador de números aleatorios.

VENTAJAS E INCONVENIENTES

El método CSMA/CD es idóneo para situaciones de bajo tráfico, ya que cuando una estación desea comunicarse puede hacerlo instantáneamente. Sin embargo, si el tráfico de la red aumenta, el número de colisiones es mayor y por tanto también aumentan los tiempos de espera. Es el protocolo más empleado en redes con topología en bus. La velocidad real de transmisión es aleatoria, ya que depende de los tiempos de espera ocasionados por colisiones, resultando imposible determinar el tiempo necesario para que una estación envíe la información eficazmente.

PASO DE TESTIGO (TOKEN RING)

Suele utilizarse en topologías tipo bus (Token bus) y anillo (Token Ring). Se basa en enviar un paquete de información, llamado testigo, que circula de estación en estación. Cuando una de las estaciones desea establecer comunicación espera a que dicho testigo circule por ella y lo retiene. Este tiempo de espera depende del número de estaciones conectadas a la red y de las que desean transmitir en ese momento, pero esta limitado, ya que la estación emisora sólo puede disponer del testigo durante un intervalo de tiempo prefijado. De esta forma, si pasado este periodo desea seguir transmitiendo debe esperar a que el testigo haga el recorrido completo por la red y le vuelva a corresponder su turno. La estación poseedora del paquete toma el control de la red y puede establecer comunicación con cualquier otra estación conectada a la red.

VENTAJAS E INCONVENIENTES

La cantidad de información que se envía por el medio es siempre menor que la velocidad máxima de transmisión soportada por éste, ya que cada estación antes de comunicarse debe esperar a que llegue el testigo. Generalmente el tiempo medio de espera será la mitad del tiempo necesario para que el testigo circule por todas y cada una de las estaciones. Este protocolo es el más adecuado cuando se utilizan aplicaciones en tiempo real.

INTERROGACION (POLLING)

Consiste en que una unidad central va preguntando a cada una de las estaciones si desea transmitir. Cuando una de las estaciones desea transmitir debe esperar a que la unidad central se lo pregunte y solicitar permiso. Si se lo concede, la estación emisora debe enviar la información a la central para que esta la reenvíe a la estación de destino. Durante el proceso de interrogación la estación puede recibir la información de la red que estuviera dirigida a ella.

VENTAJAS E INCONVENIENTES

Es un método muy sencillo de aplicar. Además, permite que a las estaciones que se les ha definido un orden de prioridad mayor sean interrogadas más veces. Sin

embargo, tiene el gran inconveniente de que si la computadora central cae, toda la red cae con él. Otro inconveniente es que en la computadora central se produce un embudo de información que puede reducir los tiempos de transmisión de las estaciones.

REDES IEEE 802

Dada la gran variedad de protocolos de acceso totalmente incompatibles (no interoperables) que pueden diseñarse, el Instituto de Ingenieros Electricistas y Electrónicos (IEEE) de los Estados Unidos decidió normalizar en 1980 un único protocolo de acceso, en su proyecto 802. Esta decisión se tomo para que la existencia de este único protocolo normalizado impulsara a los fabricantes a construir interfaces de red en grandes volúmenes y que de esta manera los precios disminuyeran, y el mercado de las redes locales aumentara considerablemente.

Varias propuestas fueron presentadas al IEEE por diferentes instituciones y al tratar de evaluarlas se encontró que no se tenían criterios que definieran qué propuesta era técnicamente la mejor. Eso llevó a normalizar no uno sino tres protocolos de acceso: CSMA/CD (apoyado por Xerox, Intel y DEC), Token bus (apoyado por General Motors) y Token Ring (apoyado por IBM)

Para conservar una misma estructura en estos tres protocolos, se definió el modelo de referencia IEEE 802 que contempla la funcionalidad de los dos primeros niveles de la arquitectura OSI.

CAPA DE APLICACION	Contiene una variedad de protocolos que se necesitan frecuentemente
CAPA DE PRESENTACION	Se ocupa de los aspectos de sintaxis y semántica de la información que se transmite
CAPA DE SESION	Gestiona el control de dialogo
CAPA DE TRANSPORTE	Acepta los datos de la capa de sesión, los pasa a la capa de red, asegurándose que lleguen correctamente al otro extremo
CAPA DE RED	Se ocupa del control de la operación de la subred
CAPA DE ENLACE	Transforma un medio común y corriente en una línea sin errores de transmisión para la capa de red
CAPA FISICA	Se ocupa de la transmisión de bits a lo largo de un canal de comunicación

Figura II.7.6 Modelo de referencia OSI

La capa de control de acceso al medio (MACA) es la parte central del modelo y define el protocolo de acceso a la red local. La capa física define el medio de transmisión, la

velocidad (entre 1 y 16 Mbps) y el modo de transmisión (banda ancha o banda base), y la codificación (o modulación) utilizada. La capa LLC (Control Lógico del Enlace) puede ofrecer servicios de transmisión de datos orientados a conexión (confiables) o sin conexión entre dos usuarios de la red, y es similar a la capa 2 del modelo OSI.

El modelo IEEE 802 engloba también las normas relativas a la red FDDI (Interface de datos distribuidos por fibra óptica).

LOS DOS ESTANDARES POPULARES

A pesar del hecho de que existen gran número de tipos de red para atender a las distintas necesidades de los usuarios, en la actualidad prácticamente todas las redes se construyen atendiendo a dos estándares: Ethernet y Token Ring.

EL PROTOCOLO ETHERNET

Las redes Ethernet están basadas en el protocolo de acceso al medio CSMA/CD. Por tanto el acceso al medio y la transmisión de la información se realiza como se describió anteriormente. Sin embargo, la recepción se realiza de forma más sencilla. Cada una de las estaciones conectadas a la red está siempre "escuchando" lo que transita por el cable, de forma que si detecta algún mensaje lo descifra y analiza a quien va dirigido. Si la dirección de destino del mensaje coincide con la dirección

propia de la tarjeta, ésta almacena el mensaje en su memoria e informa a la computadora en la cual está instalada. Una vez que la computadora es informada, la tarjeta Ethernet espera a que ésta acepte el mensaje. Si por el contrario la dirección de destino no es la de la tarjeta simplemente lo omite.

La velocidad máxima de transmisión en una red Ethernet, si no hay colisiones, es de 10 Megabits por segundo.

EL PROTOCOLO TOKEN RING

Atiende a las características del protocolo de paso de testigo. El medio de transmisión que se emplea es en anillo. Cuando una estación recibe el testigo por su lado "izquierdo", procedente de la estación anterior, debe pasarlo a la de su lado "derecho". Si la estación que recibe el testigo no tiene nada que transmitir, sencillamente, lo deja pasar. Cuando una estación emite un mensaje el testigo lleva la dirección de la estación de destino, que hasta llegar a esta última deberá pasar por todas las anteriores. Si a la estación transmisora le llega el testigo con la dirección que ella había indicado, significa que la estación receptora está desconectada de la red.

Si una de las estaciones de la red se avería puede o bien desaparecer el testigo o bien enviar múltiples testigos en el anillo. Para resolver este problema una de las estaciones, la maestra, se encarga de volver a poner el testigo en marcha por la red o

retira los testigos sobrantes de la red. La velocidad de transmisión es de 4 Mbps. Esta no es la velocidad efectiva, ya que es necesario que el testigo pase por todas las estaciones.

EL CABLEADO ETHERNET

El medio que se emplea en este tipo de redes es un bus de cable coaxial. Existen dos tipos de cable Ethernet: el cable estándar o thick-Ethernet (cable grueso) y el cheapernet o thin-Ethernet (cable fino). Cuando el cable utilizado es el estándar, la conexión de las estaciones a la red se realiza mediante unos elementos llamados transceptores, además de los cables, llamados drop, que unen los tranceptores con las estaciones. La longitud máxima del cable coaxial es de 500 metros.

Sin embargo, cuando se utiliza cable Cheapernet las tarjetas de red llevan incorporado su propio transceptor, lo cual las permite conectarse directamente al cable coaxial mediante un conector especial en forma de "T". Este se conecta a la tarjeta en su conector de la parte inferior, utilizándose los otros dos conectores para entrada y salida del cable coaxial.

La distancia máxima cuando se emplea cable Cheapernet es de 185 metros. Si es necesario sobrepasar la distancia máxima, es necesario emplear repetidores. El

inconveniente es que si uno de los tramos del cable sufre alguna anomalía toda la red cae.

EL CABLEADO TOKEN RING

Aunque el tipo de topología física es un anillo, las técnicas actuales han simplificado enormemente la forma de conectar las estaciones, utilizando unas unidades de interconexión llamadas MAU, que simplemente actúan como puentes. Aunque el aspecto físico cuando se instalan las MAUs es el de una estrella, gracias a los puentes de su interior sigue siendo un anillo. Interconectando MAUs entre sí, se puede hacer crecer el anillo para conectar un gran número de estaciones.

COMPARACION DE LOS DOS PROTOCOLOS

El inconveniente de las redes Ethernet es la probabilidad de colisión, que aumenta con el número de estaciones conectadas a la red. No obstante, pueden ser rentables con hasta 100 estaciones de trabajo, un número aceptable para la mayoría de las oficinas. Cuando el número de estaciones es reducido la velocidad de transmisión es cercana a los 10 Mbps, lo que la hace realmente efectiva.

En el caso de las redes Token Ring, cuando el número de estaciones es bajo, la velocidad máxima es de 4 Mbps, lo que está muy por debajo de la red Ethernet. Esta

velocidad se reduce aún más cuando el número de estaciones conectadas es mayor. Sin embargo, en condiciones de tráfico elevado no existen retransmisiones simplemente, cada estación debe esperar el tiempo máximo predeterminado para recibir el testigo. Por tanto, en una red de gran tráfico, el tiempo de espera tenderá a ser un valor constante, mientras que en Ethernet aumentará con el número de estaciones conectadas.

COMPARACION ENTRE LOS CABLES ETHERNET Y TOKEN RING

El cableado Ethernet es mucho más sencillo de instalar que el cable empleado en Token Ring, ya que no requiere ni tantas cantidades de cable ni MAUs. Además, el cableado Ethernet estándar y el Cheapernet son compatibles entre sí, y este último es el más barato de todos ellos. El inconveniente principal de este tipo de cable es que si se avería, se cae toda la red. Sin embargo, en el caso de Token Ring bastará con desconectar la MAU correspondiente, permitiendo que el resto de las estaciones conectadas a otras MAUs puedan seguir trabajando. Otra ventaja del cable Ethernet es que algunas otras redes sólo soportan este tipo de cable (por ejemplo, TCP/IP). Sin embargo, el cable Token Ring permite conectar muchas más estaciones que el Ethernet. La característica común, es que ambos soportan cableado de pares trenzados no apantallados.

REDES MULTIMEDIA

Las redes locales fueron originalmente diseñadas para transmitir información en ráfagas entre computadoras. Sin embargo, actualmente se desea transmitir no sólo datos sino también otro tipo de tráfico, por ejemplo, voz. De esta manera las redes locales transmitirían voz y datos, eliminando la necesidad de comunicaciones separadas.

Para que una estación pueda transmitir voz en tiempo real es necesario asegurar que transmitirá un byte exactamente cada 250 microsegundos, algo que no pueden garantizar Ethernet, Token Bus o Token Ring. Esta limitante condujo al desarrollo de nuevas redes (por ejemplo Carthage) o a la adaptación de redes existentes (FDDI-II) para satisfacer las necesidades del tráfico isocrono (que no admite retrasos).

Estos nuevos protocolos de acceso funcionan sobre anillos o buses unidireccionales y tienen dos modos de transmisión multiplexados sobre la misma red. El primero de ellos se utiliza para las aplicaciones clásicas de transferencia de datos en ráfagas mientras que el segundo, basado en técnicas de multiplexaje en el tiempo, se utiliza para transmitir tráfico isocrono. Este segundo modo de transmisión está basado en la circulación sobre la red de ciclos transmitidos periódicamente (250 microsegundos) para una estación especial. Uno o más bytes en cada ciclo pueden ser reservados

(utilizando el primer modo de transmisión) por cada estación para la transmisión de tráfico isocrono.

Las nuevas redes locales y metropolitanas deben transmitir a velocidades elevadas (del orden de 100Mbps) y tener un protocolo de acceso eficiente para poder acomodar un número considerable de usuarios y soportar las velocidades de tráfico isocrono necesarias para transmitir, por ejemplo, vídeo en tiempo real.

LAS REDES INALAMBRICAS

Las redes de área local se han convertido en el medio natural para lograr la interconectividad entre sistemas mayores de cómputo (como minis y maiframes) y equipos menores, a fin de intercambiar y compartir información. Esto ha ocasionado que el número de nodos conectados a un servidor aumente, puesto que las ventajas de la interconectividad atraen a los usuarios que trabajan de manera aislada. Esta adición puede causar fuertes problemas al administrador.

Otra moda de nuestro tiempo es la relocalización de terminales y el establecimiento de grupos de trabajo temporales. Para dar atención a este tipo de solicitudes, el administrador de la red tiene que analizar la estrategia más adecuada para el redimensionamiento del sistema.

Cuando la instalación esta basada en un sistema de cableado estructurado, el proceso se reduce a la adquisición de la tarjeta de interface y a la asignación de un puerto en el centro del cableado. Si no se tiene la fortuna de tener dicho sistema de cableado, la solución se complica ya que, dependiendo del tipo o topología de red, puede ser necesaria hasta la interrupción total del servicio. Una opción es la instalación de redes inalámbricas.

Un sistema inalámbrico puede perfectamente acoplarse al sistema tradicional de cable, o ser totalmente inalámbrico. Lo más común es diseñar ambientes híbridos, en los que conviven ambas tecnologías y el servidor contiene dos tarjetas, una para cada tipo de red.

Básicamente, las redes inalámbricas se basan en el uso de dos tecnologías. Cada una de ellas tiene pros y contras, específicamente en términos de la velocidad de transmisión, compatibilidad y medio en el que se instala. Las dos tecnologías son: ondas de radio o espectro distribuido en el rango UHF y microondas y luz infrarroja.

Dado que cada una tiene variaciones en el desempeño, son precisamente esas características específicas del producto las que determinan cuál sistema es más apropiado para una aplicación en particular.

La habilidad de utilizar la energía eléctrica para transmitir una cierta cantidad de información, ya sea por medio de cables, ondas de radio o de luz, es un factor de la frecuencia o el número de transiciones de la señal por unidad de tiempo. La frecuencia (o el número de ciclos por segundo) de una señal es representada por un hertz, donde un hertz es igual a un ciclo por segundo. La tecnología electrónica permite la detección de las variaciones en la amplitud, frecuencia, fase y patrones o combinaciones de esas características.

Lógicamente, para una frecuencia dada, sólo una cierta cantidad de información puede ser transmitida por medio de la utilización de estas características, para permitir que la señal represente algún código específico de información. Si una señal de un hertz puede representar una determinada cantidad de información, a dos hertz puede representar dos veces más información dentro de la misma unidad de tiempo.

TOPOLOGIA INALAMBRICAS

Cuando se tiene un sistema cableado, la topología se define por la forma física en que se interconectarán las computadoras en red. En un sistema inalámbrico (ondas de radio o luz), esto se refiere a la comunicación o esquema lógico de transmisión.

Se emplean dos métodos fundamentales: en el primero, cada nodo se comunica con todos los demás. En el segundo, existe un dispositivo central, a través del cual se

conectan todos los módulos. Una ventaja asociada al uso de un controlador central es la de poder incorporar sistemas de administración y control de acceso.

VENTAJAS

Más del 85% de los problemas en los fallos de una red es ocasionado por el cableado. Si éste se elimina, el tiempo medio entre fallas aumentará en forma considerable.

REDES CON CABLE

VENTAJAS

- Tecnología madura
- Altas velocidades de transmisión
- Confiabilidad
- Cumple con varios estándares de la industria

LIMITACIONES

- Reparaciones costosas
- El tiempo medio entre fallas es menor
- El tiempo de reparación es mayor
- Dificultad para el tendido del cableado o la reutilización de éste
- Mayor tiempo de instalación

REDES INALAMBRICAS

VENTAJAS

- Buenas características de desempeño
- Resistencia a la interferencia externa
- Seguridad
- Bajos costos de operación
- Facilidad de instalación
- Facilidad en el mantenimiento y detección de fallas
- Útil en ciertas circunstancias geográficas
- Menor tiempo de instalación
- Buen nivel de integración con redes tradicionales existentes
- Mínima capacitación para la instalación

LIMITACIONES

- Potencia y distancia limitadas
- Velocidad de transmisión limitada
- Alto costo por unidad
- Es una tecnología relativamente nueva

INTERCONEXION DE REDES LOCALES

La primera generación de conectividad fue crear redes locales, tal vez la segunda generación de conectividad se refiera a la habilidad de poder conectar redes con redes. A esta habilidad se le conoce por su término en inglés "internetworking" o interconexión de redes.

Básicamente, existen cuatro tipos de productos para la interconexión de redes: repetidores, puentes, ruteadores y pasarelas (gateways).

Cada uno de ellos representa un nivel diferente de conectividad y funcionalidad correspondiente a los modelos de referencia IEEE 802 y OSI. Estos modelos se aplican a cualquier conjunto de productos para conectividad, desde módem hasta redes globales y redes X.25.

REPETIDORES

Al propagarse a través de un medio de transmisión (por ejemplo, par trenzado , cable coaxial o fibra óptica) las señales transmitidas sufren, gradualmente , una disminución en su amplitud y una distorsión en su forma. Por esta razón se fija un límite a la longitud máxima del medio de transmisión que asegure que la atenuación y distorsión no impidan la interpretación correcta de las señales recibidas . Si la longitud

del medio de transmisión excede este límite, deben insertarse repetidores a lo largo del medio de transmisión que restauren el nivel y la forma de las señales.

Los repetidores son el producto más sencillo para la interconexión de redes y operan al nivel más bajo del modelo OSI (la capa física). Los repetidores físicamente extienden el alcance de una red regenerando señales (bits) de un medio de transmisión y retransmitiéndolas a otro. Esto puede lograrse con un repetidor que se conecta directamente a los dos medios, o utilizando dos repetidores remotos conectados por un enlace infrarrojo o de fibra óptica.

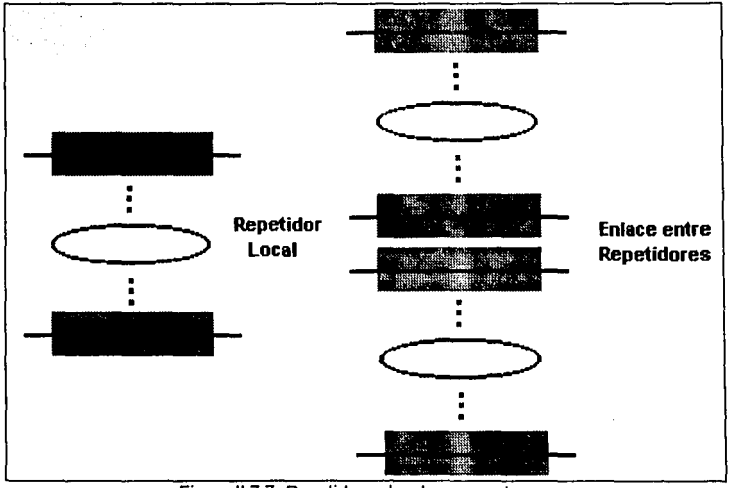


Figura II.7.7 Repetidores locales y remotos

Los medios conectados mediante un repetidor pueden ser de naturaleza distinta , por ejemplo coaxial grueso y coaxial delgado, o coaxial y par trenzado. Es posible también conectar varios segmentos entre si utilizando un solo repetidor multipuertos.

Los repetidores interconectan segmentos para constituir una sola red física. El número de repetidores que pueden conectarse en cascada para formar esta red está limitado por el protocolo de Control de Acceso al Medio (MAC) utilizado, ya que existe un retardo de propagación máximo que debe respetarse.

Los repetidores, como conectores en la capa física, pasan bits directamente de un medio a otro y no llevan a cabo ningún procesamiento de alto nivel (carecen de inteligencia); es por esto que tienen mayor rendimiento (en bps transmitidos) que los puentes, ruteadores y pasarelas. Además, su sencillez técnica conlleva a su relativo bajo costo y a su facilidad de instalación.

PUENTES (BRIDGES)

Cuando se utilizan repetidores, las tramas enviadas por una estación se propagan a todos los segmentos de la red sin importar la localización física de la estación receptora, generando tráfico inútil en algunos segmentos de la red. Para solucionar este problema pueden utilizarse para solucionar este problema pueden utilizarse puentes que permiten aislar el tráfico local de las diferentes segmentos de una red.

Con un grado de complejidad más elevado que los repetidores, los puentes conectan redes al nivel de la capa de enlace de datos del modelo OSI y más específicamente en la subcapa MAC del modelo IEEE 802.

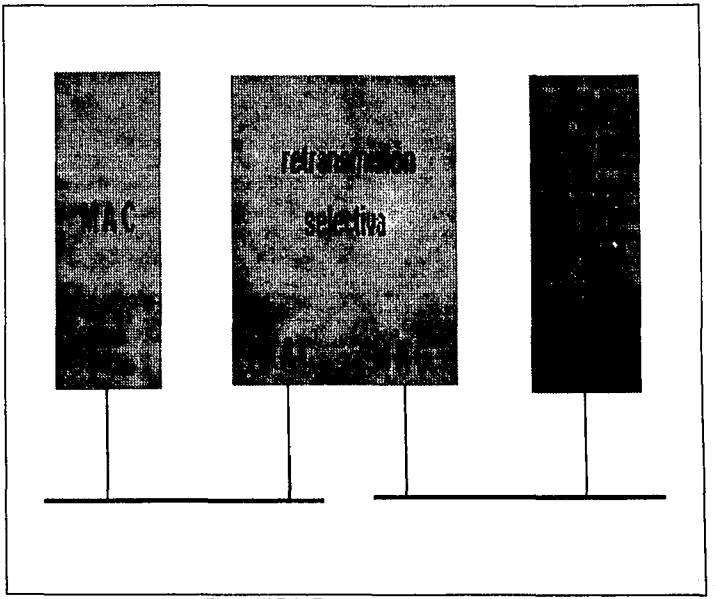


Figura II.7.8. Estructura de un puente

Los puentes permiten interconectar redes que utilicen el mismo o diferente protocolo MAC, extender el alcance de una red y aumentar el número de estaciones que pueden conectarse a ella más allá de los límites permitidos por el protocolo MAC en una red sin puentes y, debido al aislamiento de tráfico, aumentar el desempeño de la red en su conjunto y mejorar su disponibilidad.

Un puente se conecta a dos o más redes locales y conoce las direcciones MAC de las estaciones que pueden ser alcanzadas directa o indirectamente a través de cada uno de estos segmentos. De acuerdo a la norma IEEE 802.Id. un puente lee las direcciones origen y destino de todas las tramas que circulan por los segmentos a los cuales está conectado. Si la dirección destino indica una estación en el mismo segmento sobre el cual se recibió la trama, el puente descarta la trama para evitar un tráfico inútil en los otros segmentos (mecanismo de filtrado). Si la dirección destino indica una estación en otro segmento, entonces el puente envía la trama sólo a este segmento (mecanismo de reenvío) utilizando el protocolo MAC correspondiente. Si el puente no sabe en que segmento se encuentra la estación destino, envía la trama sobre todos los segmentos a los cuales este conectado (con excepción del segmento sobre el que recibió la trama) . Un puente puede conocer dinámicamente a través de qué segmento puede alcanzar una determinada estación examinando las direcciones origen de las tramas que recibe (mecanismo de aprendizaje). De esta manera, los puentes permiten crear una única red lógica a partir de grupos de redes dispersas.

Cuando una red esta formada por un conjunto de segmentos unidos por puentes es posible utilizar una topología en cascada o crear caminos redundantes entre los diferentes segmentos y tener una arquitectura que puede tolerar fallas En funcionamiento normal sólo existe un camino activo entre cada par de segmentos pero si una falla ocurre en algún camino activo es posible calcular dinámicamente uno nuevo.

Los puentes pueden utilizarse tanto en ambientes locales como remotos. En el primer ambiente un puente se configura con dos (o más) interfaces de red, mientras que en el segundo se utilizan dos puentes enlazados por un enlace remoto. Normalmente los segmentos interconectados por puentes remotos están separados por distancias cortas y la velocidad del enlace entre los puentes es del mismo orden de magnitud que la velocidad de los segmentos.

Para este tipo de enlaces de alta velocidad puede utilizarse, por ejemplo, fibra óptica para distancias (típicamente) hasta de 10 km. o un enlace de microondas para distancias de hasta 7 Km. Sin embargo, también es posible interconectar segmentos mediante puentes remotos que pueden cubrir prácticamente cualquier distancia utilizando líneas privadas telefónicas o líneas digitales tipo E-1. En este último caso el rendimiento de la interconexión esta limitado por la velocidad del enlace remoto.

Una característica importante de los puentes (así como de los repetidores) es que son transparentes a las estaciones de los usuarios y por lo tanto fáciles de instalar . Los puentes se conectan a la red y sin intervención del usuario funcionan automáticamente La información de enrutamiento necesaria para su operación la obtienen mediante el mecanismo de aprendizaje antes descrito. El usuario no tiene que preocuparse tampoco de la existencia de caminos alternos que perturben el proceso de aprendizaje y causen la transmisión cíclica continua de tramas entre puentes gracias al uso del algoritmo STP . La transparencia implica también que las estaciones no envían nunca tramas dirigidas directamente a los puentes, las tramas contienen sólo las direcciones MAC de las estaciones fuente y destino.

Como los puentes funcionan en la capa MAC, son independientes de los protocolos empleados en las capas superiores y permiten interconectar redes que utilizan protocolos diferentes, tales como TCP/IP, SPX/IPX y DECnet entre otros. En otras palabras, en las redes pueden coexistir diferentes tipos de protocolos de la capa 3 y superiores. En redes conectadas por puentes, las capas superiores del modelo OSI que residen en las estaciones de los usuarios eliminan cualquier incompatibilidad, lo que es muy importante para grandes organizaciones donde existen ambientes de cómputo y de comunicaciones variados, y desean un ambiente de red homogéneo y sencillo.

Un puente, a diferencia de un repetidor, almacena las tramas que recibe y verifica que no tengan errores antes de procesarlas. El almacenamiento y procesamiento de tramas realizado por los puentes introduce un retardo que no existe en un repetidor y disminuye por lo tanto su rendimiento.

RUTEADORES

Los ruteadores conectan redes a nivel de la capa 3 del modelo OSI y ofrecen conectividad con enrutamiento selectivo de paquetes de datos, siguiendo los métodos establecidos por el protocolo de la capa de red que utilizan . Los ruteadores pueden enviar paquetes sobre diferentes rutas en una red dependiendo de ciertos criterios, tales como la ruta con menor costo, la más rápida o la más segura. Los ruteadores, a diferencia de los puentes, aprovechan la existencia de rutas alternas en la red.

Los ruteadores pueden servir para interconectar redes locales a redes de área amplia o redes locales entre sí . Para interconectar redes locales que se encuentran físicamente cercanas un ruteador se conecta directamente a las redes que interconecta, mientras que para interconectar redes locales geográficamente dispersas los ruteadores se conectan a través de una red de área amplia.

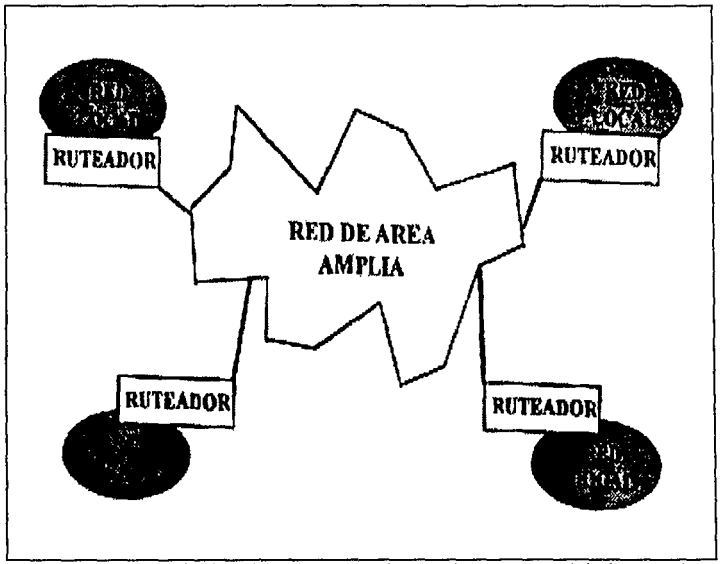


Figura II.7.9. Conexión de redes locales usando ruteadores y una red de área amplia.

Esta espina dorsal de área amplia puede utilizar , por ejemplo , una red X25 o, si se requiere mayor velocidad de acceso, un servicio de relevo de tramas ("frame relay") o un Servicio Conmutado (2da transmisión) de Datos a Multimegabits ("SMDS"). La utilización de una red de área amplia para la interconexión de redes locales presenta menor costo que la utilización de líneas privadas.

Los ruteadores utilizan un direccionamiento (lógico) de nivel 3 de tipo jerárquico(red, estación) para rutear los paquetes entre diferentes redes. Además, utilizan sólo la parte de red de la dirección para tomar sus decisiones de enrutamiento, lo que significa que sirven para interconectar redes separadas más que para formar una red lógicamente unificada como lo hacen los puentes . Esta característica facilita la administración de la interconexión de redes, sobre todo cuando el tamaño de la red es considerable.

Los ruteadores no son transparentes a las estaciones de los usuarios, deben ser direccionados directamente por éstas para transmitir un paquete de una red a otra. Cuando una estación en una red local quiere enviar un paquete a una estación que no se encuentra en la misma red , envía una trama (subcapa MAC) dirigida a un ruteador conteniendo el paquete (capa de red) que debe ser transmitido a la otra red. El ruteador utiliza la dirección de red de la estación destino contenida en el paquete para determinar si puede enviarlo directamente a su destino final o necesita pasar por otro ruteador.

Los ruteadores son capaces de determinar dinámicamente, en función del tráfico y la disponibilidad, la ruta que deben seguir los paquetes. Además, el protocolo de la capa de red permite a los ruteadores fragmentar los paquetes al pasar por redes con diferentes tamaños máximos permitidos y reensamblarlos al llegar a su destino final.

Por requerir procesamiento adicional para manipular paquetes de acuerdo al protocolo de la capa de red, los ruteadores son generalmente más costosos y tienen menor rendimiento que los puentes.

PASARELA (GATEWAY)

Son los dispositivos de interconexión más complejos ya que permiten la comunicación entre redes que utilizan pilas de protocolos totalmente diferentes. Para lograrlo, las pasarelas realizan la conversión completa de una arquitectura a otra sin modificar los datos transmitidos, de modo que los protocolos utilizados en la red fuente puedan ser entendidos en la red destino. Al nivel mas alto, las pasarelas permiten que ciertas aplicaciones se comuniquen entre si. Las pasarelas con generalmente mas costosas y lentas que los puentes o ruteadores ya que efectúan mas procesamiento para llevar a cabo la conversión de protocolos.

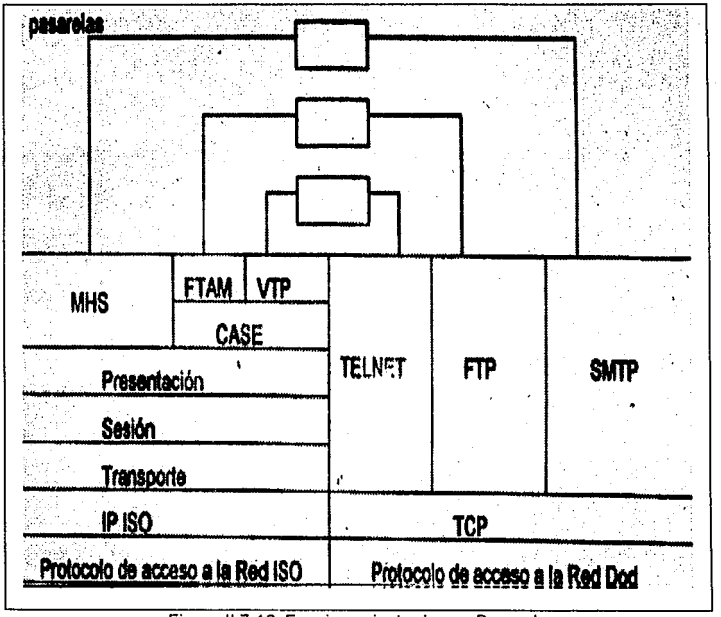


Figura II.7.10 Funcionamiento de una Pasarela

Capítulo III

Planteamiento del Problema y Propuesta de Solución

III.1. ESTUDIO DE LA SITUACION ACTUAL

Uno de los objetivos del Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Ciudad de México es que su comunidad tenga acceso a equipo de cómputo como una herramienta en el desarrollo de sus prácticas en las más diversas áreas de conocimiento profesional, figura III.1.1.

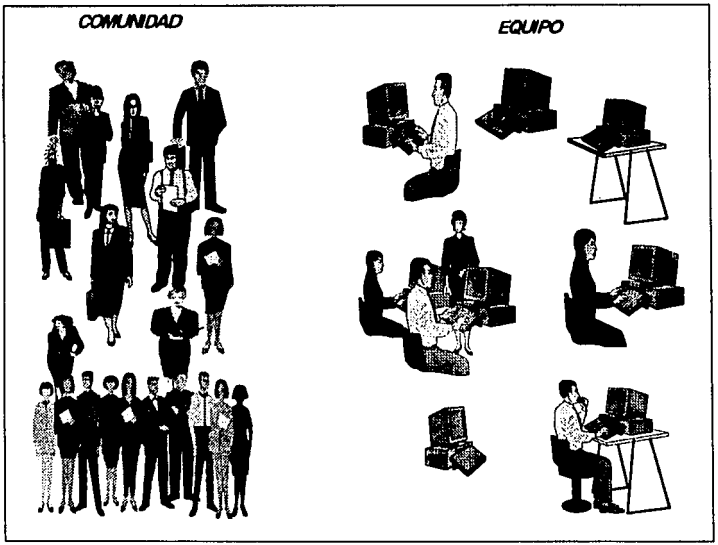


Figura III.1.1. Comunidad y equipo.

Conscientes de que el tiempo es valioso requiere proporcionar un servicio eficiente para tener acceso al mismo, en forma ágil con solicitudes que se asignen oportunamente y con equipo en óptimas condiciones y siempre disponible.

Todos los equipos de cómputo comparten recursos tanto de hardware como de software entre los que podemos mencionar el uso de diferentes impresoras de alta velocidad y al acceso a servicios de información al exterior a través del uso de la red INTERNET, red de redes, figura III.1.2.

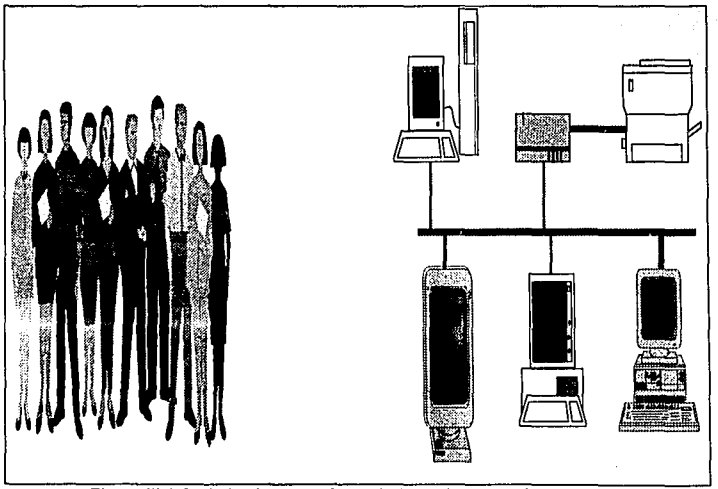


Figura III.1.2. Todos los usuarios trabajan bajo un ambiente de red.

Actualmente el Centro de Cálculo del Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Ciudad de México no cuenta con un sistema de administración de apartados de equipo de cómputo para su comunidad. Un apartado es la acción que realiza un miembro de la comunidad al asegurar el uso de un tipo de computadora durante un día y una hora al día. En la Figura III.1.3. se muestran los aspectos a controlar:

- El reglamento
- Los apartados por tipo de máquina, día y hora
- Los accesos al centro de cálculo
- Las cancelaciones
- Las reasignaciones de apartado por equipo dañado y que no esté en uso
- Directorio de alumnos
- Inventarios de equipo
- Estadísticas sobre oferta y demanda de equipo en cada hora para cada tipo de máquina, como apoyo a la toma de decisiones gerenciales.

Figura III.1.3. Aspectos a controlar en la administración del equipo de cómputo.

USUARIOS DE LOS SERVICIOS

Son usuarios de los servicios del Centro Electrónico de Cálculo del ITESM:

- Alumnos
- Personal Académico
- Personal Administrativo
- Exalumnos (con previa identificación)

EQUIPO EN SERVICIO

64 PS Value Point IBM con procesador 486 DX a 33 Mhz.

48 POWER MAC 7100 con procesador de 32 bits y velocidad de 66 Mhz

64 Workstation RS/6000 de IBM

Los tipos de usuarios y equipo se muestran en la figura III.1.4.

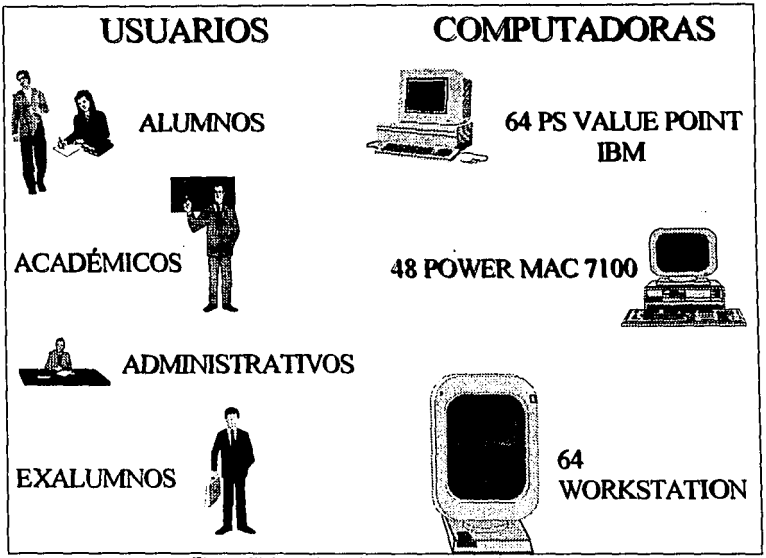


Figura III.1.4. Tipos de usuarios y equipo.

REGLAMENTO PARA LA SOLICITUD DE APARTADO

Los solicitantes del servicio deberán de cumplir el reglamento, entre los que se encuentra:

- Presentar su credencial vigente
- Solicitar el servicio entre las 8:00 y 22:00 hrs.
- Solicitar el apartado con anticipación (que puede ser cualquier hora anterior a la utilización) o realizando su registro en el mostrador directamente si hay disponibilidad de equipo.
- El apartado del equipo es personal y no podrá hacer cambios del mismo sin previa autorización.
- El usuario tiene derecho de usar el equipo una hora por día; en caso de requerir más tiempo deberá renovar su apartado dependiendo de la disponibilidad del mismo.
- Los apartados que no se ocupen dentro de los primeros quince minutos de la hora en curso, se cancelarán para asignar el equipo a cualquier usuario que solicite el equipo.
- Las entradas al C.E.C. pueden ser:

    1. Cuando exista apartado realizado con anterioridad para la hora en curso o
    2. Sin apartado cuando exista disponibilidad de equipo.

- Ningún alumno podrá entrar al C.E.C. dentro de los últimos 15 minutos de cada hora. En caso de requerir acceso, se debe realizar a la hora siguiente.

HORARIOS DE SERVICIO

Existen dos tipo de horario:

- Horario normal:

  - Lunes a viernes de 7:00 a 20:00 hrs.
  - Sábado de 8:00 a 16:00 hrs.

- Extendido (período de exámenes):

  - Lunes a Viernes de 7:00 a 24:00 hrs.
  - Sábado de 8:00 a 20:00 hrs.

POBLACION Y HORAS MAQUINA

La población registrada en enero de 1995 fue de 7500 entre alumnos, administrativos y académicos.

Las hora máquina potenciales para los usuarios por semana en horario normal son:

Para equipo PC : 64 equipos * (13 horas * 5 días + 8 horas) = 4672 horas.

Para equipo MAC : 48 equipos * (13 horas * 5 días + 8 horas) = 3504 horas.

Para equipo WS : 64 equipos * (13 horas * 5 días + 8 horas) = 3674 horas.

Lo anterior nos da un total de 12848 horas, por lo tanto cada usuario podría disponer casi de dos horas a la semana de uso de equipo, pero sabemos que no todos los días requieren equipo ni toda la comunidad al mismo tiempo.

ANTECEDENTES

Se han realizado intentos por contar con un sistema que permita asignar las computadoras con lo que los alumnos y profesores puedan realizar sus prácticas en forma eficiente, sin embargo los diseños han resultado fallidos en el momento de su implementación. Todos utilizaron diferentes plataformas, tales como:

- Con un manejador de base de datos PARADOX
- Desarrollo con el lenguaje de programación C
- Actualmente se utilizan hojas de cálculo con EXCEL

El diseño con PARADOX no cumplía con los requisitos de integridad de información, como resultado se tenían alumnos apartando un mismo equipo el mismo día y hora, también se tenía como disponible un equipo dañado. Aunque se realizaron mejoras, nunca dejaron satisfecho al usuario del sistema, figura III.1.5.

Aunque con el lenguaje de programación C su procesamiento era muy rápido, el principal inconveniente era que todo se hacía en memoria, de tal forma que durante las fallas de energía existían pérdidas de información, figura III.1.5 .

Los dos diseños anteriores fueron realizados por personal que realizaba su servicio social en el ITESM, se requerían estadísticas en cualquier momento, figura III.1.6.

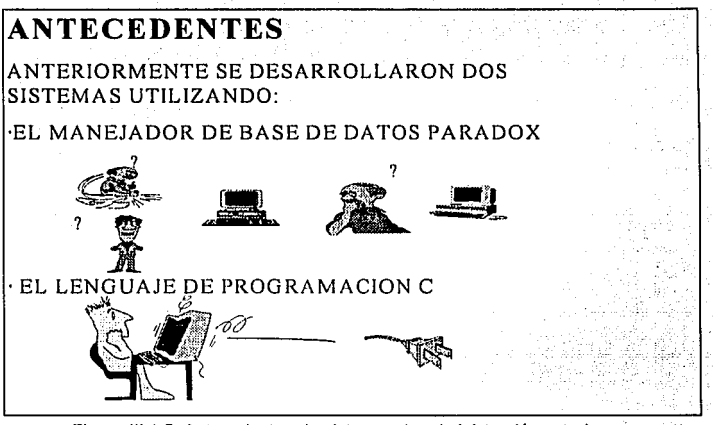


Figura III.1.5. Antecedentes de sistemas de administración anteriores.

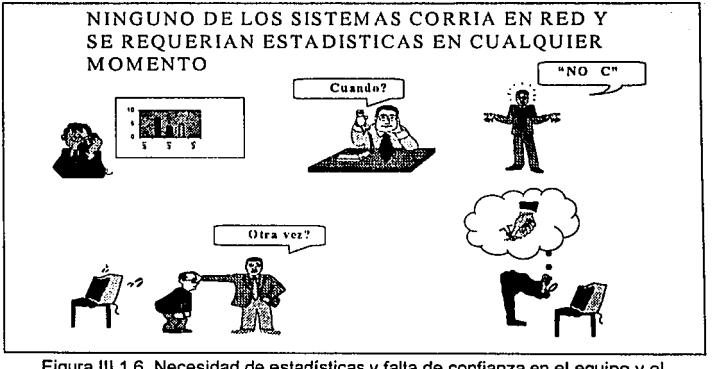


Figura III.1.6. Necesidad de estadísticas y falta de confianza en el equipo y el personal.

Estos sistemas no lograron ayudar a obtener una mejor atención y los responsables del Centro de Cálculo decidieron llevar el control manualmente con el uso de 3 hojas de cálculo con el paquete de EXCEL de Microsoft, figuras III.1.7. Sin embargo pronto se generaron nuevos problemas.

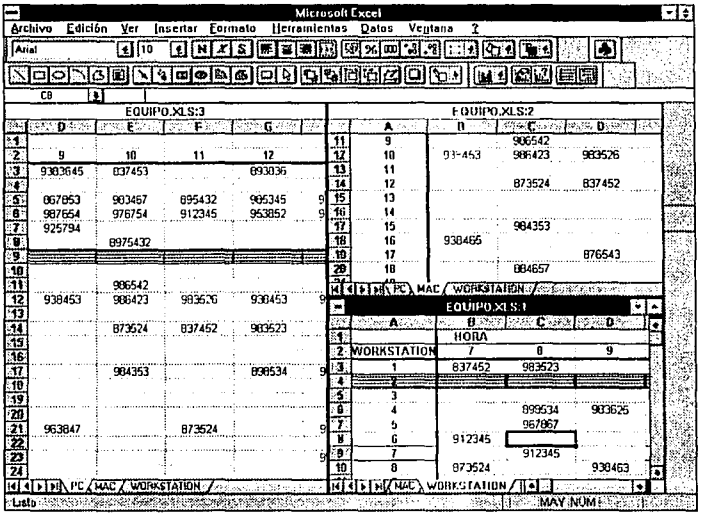


Figura III.1.7. Hojas de cálculo para la administración de equipo de cómputo.

LAS HOJAS DE CALCULO EN EXCEL DE MICROSOFT

Se tienen tres hojas de cálculo que controlan los grupos de equipo, PC, MAC y WORKSTATION, figura III.1.8. En cada hoja las filas representan el número de máquina y las columnas la hora de entrada.

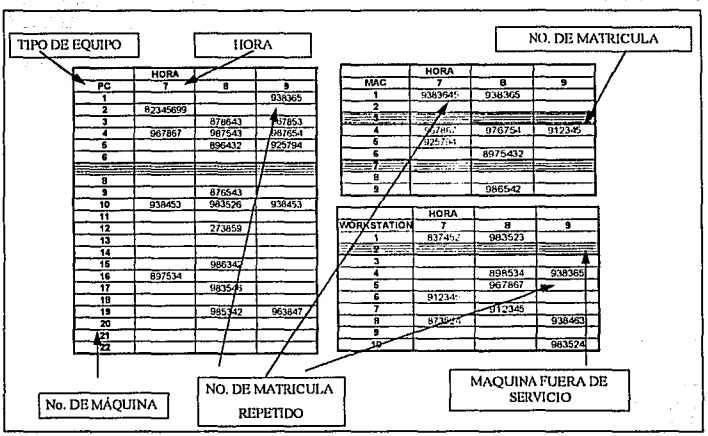


Figura III.1.8. Hojas de cálculo para el control de equipo.

El proceso para apartar equipo es el siguiente:

1. Los alumnos llegan al mostrador y deben de proporcionar su número de matrícula, así como la hora y el tipo de equipo que desean utilizar, ya sea PC, MAC o WORKSTATION.

2. Se verifica si el alumno ya solicitó un equipo por medio del menú de búsqueda de Excel, en cada una de las hojas. Si ya está registrado, se le rechaza.
3. Si el alumno tiene derecho a apartar se activa la hoja de cálculo correspondiente al tipo de equipo.
4. Se busca si existe un equipo disponible a la hora solicitada
5. Si una máquina está fuera de servicio, en la hoja de cálculo se muestra toda la fila con un formato de líneas horizontales que evita introducir el número de matrícula.
6. Si hay equipo disponible, se le asigna dándolo de alta registrando su matrícula en la celda correspondiente.

Como puede observarse esto representa un registro casi manual del proceso de apartado, con sus respectivas desventajas.

DESVENTAJAS DE LAS HOJA DE CALCULO:

1. Demasiadas operaciones, ya que a cada alumno se le debe de buscar en las tres hojas de cálculo.
2. No hay forma de validar si el número de matrícula es el correcto, lo que lleva el riesgo de que un alumno esté más de una vez registrado debido a que se capturó por error alguno de los números.
3. No hay forma de determinar si un alumno que apartó una hora realmente la ocupó.

4. Al no saber que alumno no ocupó su apartado, no hay forma de determinar si se reasigna ese espacio a otro alumno.
5. Cada mañana al empezar el turno deberán de crearse las tres hojas nuevas cambiándoles el nombre, el cual corresponde con la fecha asignada.
6. Si un alumno desea cambiar su apartado en otra hora o tipo de máquina, habrá que buscarlo con el menú de búsqueda de Excel y repetir el proceso de alta en la hoja, hora y numero de equipo correspondiente.
7. No es posible obtener estadísticas que verdaderamente apoyen en la toma de decisiones. Es necesario concentrar la información en otro tipo de tabla lo cual por sí solas las tres hojas actuales, no se prestan fácilmente a su manipulación.

No se cuentan con estadísticas que se requieren para la toma de decisiones, tales como:

1. Determinar el número de alumnos atendidos por hora y día
2. Número de equipos fuera de servicio por hora, día y tipo
3. Número de alumnos que apartaron equipo y no lo ocuparon
4. Estimar la oferta y la demanda del equipo por hora, día y tipo de equipo

Se invirtieron recursos económicos y humanos que a corto plazo produjeron pocos resultados y el servicio empeoraba, figura III.1.9.

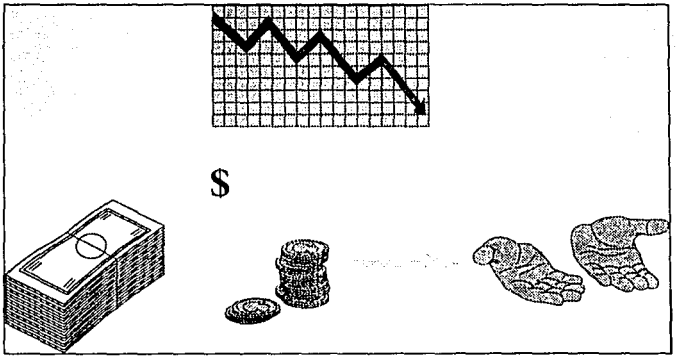


Figura III.1.9. Inversión de recursos y resultados pobres.

Por lo anterior existe la necesidad urgente de contar con un sistema adecuado para fortalecer la excelencia académica de la comunidad mediante la asignación oportuna de equipo de cómputo, figura III.1.10.

NECESIDAD URGENTE

CONTAR CON UN SISTEMA AUTOMÁTICO QUE NOS PERMITA SUPERAR TODAS LAS DEFICIENCIAS ANTERIORES PARA PROPORCIONAR UN SERVICIO ÁGIL Y OPORTUNO A LA COMUNIDAD DEL ITESM CCM

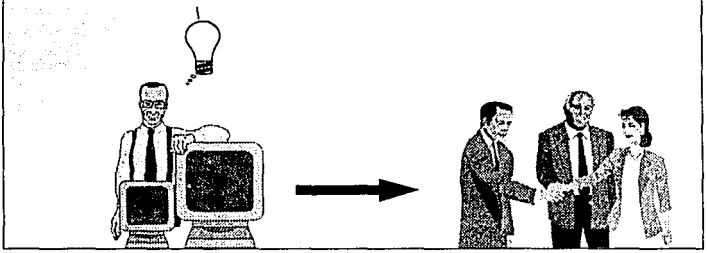


Figura III.10. Necesidad urgente de un sistema de administración de equipo.

III.2 ESTRATEGIA DE SOLUCION

Ante el análisis de la organización, y la revisión bibliográfica de los distintos tópicos teóricos involucrados, es necesaria una revisión exhaustiva de la manera en que se plantea el problema. Para ello, mediante el capítulo III.1, se determina cuál es la manera en que se está resolviendo actualmente el problema.

Es de suponerse, que el desarrollo de un sistema bajo las condiciones en que se realiza el sistema en cuestión, requiere de una evaluación de la forma de pensar de la gente involucrada y su opinión y parecer sobre lo ya planteado y que se encuentra en operación.

Durante el desarrollo, además de atender a las solicitudes de los usuarios, se analiza detalladamente, la perspectiva a futuro del sistema. Todo lo antes citado, se reúne en lo que se denomina Estrategias de Solución, la que debe incluir la elaboración de un plan de trabajo, la recopilación de la información y la clasificación de la misma.

Como estrategia, se decidió realizar una investigación acerca de cuál es el estatus operativo actual, es decir, cuáles son los mecanismos mediante los que se resuelve o se intenta resolver la problemática y que son los que originan la necesidad de una nuevo orden.

Se intenta entonces, recabar la información pertinente, como es lo relativo a las opiniones de los elementos directivos de la organización y de la gente que operará el sistema. Es evidente, que la manufactura del sistema, emana de la información que entrega la parte operativa y que además, las decisiones del desarrollo, se verán sesgadas por las estrategias que gerencialmente se hayan dictado como líneas a seguir y políticas de la organización.

Ante el mar de información que se genera ante los puntos anteriores, se hace necesaria una etapa de clasificación de la información donde se ponen de primera mano, los aspectos más importantes que se recojan y se descarte la información que no aporte elementos válidos al desarrollo del sistema.

III.2.1 PLAN DE TRABAJO

Como plan de trabajo, se ideó la manera de obtener la información más relevante de una manera ordenada, por lo que los puntos medulares para la investigación fueron, una serie de entrevistas orientadas a los distintos puntos de evaluación jerárquica.

Las tres partes a evaluar, fueron dictadas por la misma forma en que se estructura la organización, es decir, se atendió inicialmente, al ejecutivo (director de informática), al ejecutivo de segundo nivel (toda la gente que se encuentra entre el director de

informática como líder de la organización, y las partes operativas) y los posibles operadores del sistema.

A cada uno de los personajes antes citados, fue necesario evaluarlos con aspectos importantes y correspondientes a su nivel. Cabe mencionar, que el director de informática, simplemente debe arrojar los lineamientos principales en cuanto a las líneas a seguir de conformidad con las políticas del ITESM-CCM.

De manera similar, los elementos intermedios, dentro de los que se encuentran el director de servicios computacionales y el coordinador de atención a usuarios, dictan las características que en el nivel inmediato, debe cumplir el sistema en cuanto a seguridad y reportes que debe arrojar. Aportan además algunos aspectos operativos generales que deben cumplirse en el desarrollo.

Por último, los usuarios directos aportan los aspectos más importantes al diseño y al desarrollo; las indicaciones de los usuarios terminales, soportan la utilización cotidiana que el sistema tendrá y son la gente más indicada para realizar las sugerencias pertinentes en cuanto a la elaboración de pantallas y de interfaces con el usuario. Determinan claramente las políticas de desarrollo y el *como hacer* diario.

Por supuesto, existe una calendarización en la que se propone que se defina como prioridad de diseño, la entrevista con el director de informática, que sesgará las decisiones que se deban tomar con las instancias inferiores.

Posteriormente, se atienden las necesidades de los elementos intermedios. En tercer lugar, los usuarios son examinados para extraer de ellos, la parte medular del sistema.

El orden propuesto, tiene su razón de ser en las prioridades de las distintas instancias; las determinaciones del director se superponen a las de la dirigencia de servicios computacionales ya la coordinación de atención a usuarios, que a su vez se superponen a las de los usuarios finales.

Si bien el orden impuesto dicta las jerarquías, no afecta la importancia que tienen las entrevistas con los usuarios directos ya que de manera formal, el diseño basado en necesidades, emana directamente de ellos.

III.2.2 RECOPILACION DE LA INFORMACION

La información que el sistema requiere para su diseño, emana de las distintas partes que se mencionaron en la sección anterior, y adicionalmente proviene de distintas formas de recopilación.

Como ya se mencionó, uno de los aspectos importantes en la recopilación de información, son las entrevistas con las distintas personas involucradas que se calendarizaron conforme a las prioridades antes descritas.

Adicionalmente, se realizaron cuestionarios que puntualicen las entrevistas con documentos impresos. La elaboración de los cuestionarios obedece a la necesidad de completar la documentación que se refuerza mediante las entrevistas.

Los cuestionarios también se orientan a las distintas instancias por lo que se dispone de tres de ellos.

Es singular, notar que a como al nivel del director de informática, se realizan algunas preguntas que se relacionan con aspectos de computación muy especializada, sin embargo, es necesario mencionar que el perfil de la gente encuestada es el de personal involucrado con la computación por lo que es posible que se aporten ideas valiosas.

Cuestionario CEC
Nombre:
Funciones Deseables en el Sistema:
Interacción con el Usuario:
Plataformas de Desarrollo:
Indicadores Deseados:
Políticas a Cumplir:

Figura III.2.2.1 Cuestionario al Director de Informática.

Cuestionario CEC
Nombre:
Funciones Deseables en el Sistema:
Interacción con el Usuario:
Plataformas de Desarrollo:
Indicadores Deseados:

Figura III.2.2.2 Cuestionario a Ejecutivos Medios.

Cuestionario CEC	
Nombre:	
Funciones Deseables en el Sistema:	
Interacción con el Usuario:	
Plataformas de Desarrollo:	

Figura III.2.2.3 Cuestionario a Usuarios Finales

III.2.3 CLASIFICACION DE LA INFORMACION

Como ya se mencionó, la información proviene de distintas fuentes en distintas formas, por ello, la adecuada selección de la información es un punto de suma importancia para el diseño del sistema.

Típicamente, la fuente principal de información, es la entrevista y en este caso no se omitió su importancia, por lo que la información se recabó y clasificó de la siguiente forma:

- Plataformas de Desarrollo.
- Requerimientos Gerenciales.
- Requerimientos Intermedios.
- Interface al Usuario.

En el punto correspondiente a Plataformas de desarrollo, se atendió básicamente a las opiniones del director de informática ya que desde el punto de vista más realista, es la persona encargada de la decisión de adquirir herramientas de programación y diseño con las que el campus no contase. La conclusión es que habrían de evitarse las herramientas que implicaran un costo extra a los ya realizados en herramientas de este género.

Los requerimientos gerenciales e intermedios, indican el grado de cumplimiento del sistema a un reglamento. Se estima que un punto esencial en el diseño, es que el sistema de manera natural, deberá de ser capaz de manejar la reglamentación aplicable sin la necesidad de esfuerzo extra en ese aspecto para el usuarios final. Las anotaciones de las partes a este nivel, indican todas que es necesario que se cumpla esta característica.

La interface al usuario, emanó principalmente de las entrevistas con los usuarios finales que son los que manejarán el sistema de forma cotidiana, y son ellos los que ofrecen la fuente más fidedigna de las características que el sistema debe contemplar en una interface así como en su diseño operativo cuyo quehacer cotidiano conocen perfectamente.

III.3 REQUERIMIENTOS DEL USUARIO

De acuerdo a las necesidades que tiene actualmente, el Centro Electrónico de Cálculo, con respecto al apartado de equipo de cómputo para los alumnos, se tiene el propósito de optimizar su uso, apartado y disponibilidad. Donde, por medio del empleo de estadísticas generadas automáticamente a partir de la información capturada, y la cual servirá para determinar la frecuencia de apartado de los equipos por tipo; así como los horarios de mayor demanda, las fechas de mayor demanda dentro de los ciclos escolares, etc.; sirviendo como apoyo a la toma de decisiones gerenciales. Para ello, es necesario determinar los requerimientos que el sistema debe de cubrir para así permitir un mejor desempeño del nuevo procedimiento de apartado. Esto es, lograr que los usuarios de manera rápida y sencilla consigan apartar equipo para realizar sus trabajos escolares y de aprendizaje sin la espera que surge actualmente en este sentido, debido al procedimiento manual.

Además, es importante mencionar que es necesario hacer uso de la infraestructura con la que actualmente cuenta el Centro Electrónico de Cálculo, tanto en hardware como en software. Mencionaremos que el ITESM, cuenta con el siguiente equipo para trabajar en ambiente de red: un servidor PS Value Point IBM con procesador 486, conectado en un backbone de fibra óptica con el que cuenta el campus utilizando como sistema operativo Netware 3.12 de Novell. Por otra parte, el sistema se desarrollará en Turbo Pascal versión 7.0 a petición expresa de la institución, ya que este lenguaje provee de los recursos necesarios para el desarrollo de la aplicación pues trabaja de manera directa con una herramienta de Novell, llamada Btrieve, la cual permite el manejo de archivos compartidos dentro de la red.

El sistema deberá generar un manejo de inventarios sobre los usuarios y los equipos, por medio del uso de un menú gráfico para el acceso de otros menús utilizando dos dispositivos de entrada principalmente: el mouse y el teclado. Así como contener un menú de ayuda del usuario en donde se explique el manejo del sistema

Deberá contar con un formato general para el apartado del día y hora del equipo que desea utilizar dentro del campo.

De lo anterior, podemos definir que los requerimientos del sistema se pueden clasificar en cuatro grupos principales conformados por:

- Requerimientos de equipo de cómputo.
- Requerimientos de sistema operativo en el que se instalará el sistema.
- Requerimientos en la operación de apartado.
- Requerimientos de interface gráfica.

✓ ***Requerimientos de equipo de cómputo***

Las características de equipo de cómputo mínimas en las cuales se deberá ejecutar de manera independiente el sistema de administración, serán:

- Procesador 386sx
- Memoria RAM de 4MB
- Disco duro con al menos 20 MB libres, para el manejo de la base de datos
- Monitor a color VGA de preferencia SVGA
- Impresora opcional, la cual es requerida solo en caso de necesitarse reportes

Por otra parte, en caso de ocupar un sistema operativo de red para ejecutar el sistema administrativo de manera compartida, el equipo deberá contar con las siguientes características, mínimas:

- Procesador 386sx
- Memoria RAM de 4MB

- Monitor a color VGA de preferencia SVGA
- El disco duro y la impresora serán recursos compartidos dentro de la red
- El servidor deberá contar con al menos 20 MB libres, para el manejo de la base de datos

✓ ***Requerimientos de sistema operativo donde se instalará el sistema.***

De acuerdo a las características de la infraestructura del Centro de Cálculo, el sistema deberá correr en una red Novell versión 3.12, con la cual ya se cuenta, y que permitirá compartir la base de datos en cualquier parte de la institución donde se encuentre disponible conexión a la red. Esta facilidad permitirá tanto la consulta de información estadística, como la administración y mantenimiento del sistema. Permitiendo además, en caso de existir diferentes centros de préstamo de equipo emplear la misma base de datos de los alumnos, así como una base de datos común referente al equipo existente en todos los centros de préstamo.

El sistema se desarrollará aprovechando las capacidades y características de btrieve de Novell. Producto que permite el manejo de archivos compartidos en un ambiente multiusuario, como lo es Novell. Por la facilidad que presenta en su interface a btrieve; además de ser el lenguaje sugerido por la misma institución; el lenguaje de desarrollo deberá ser Turbo Pascal de Borland versión 7, que además de esta facilidad, cuenta con una serie de librerías de apoyo en el desarrollo de una interface gráfica amigable

para el usuario. Esto en caso de trabajar únicamente el sistema con el ambiente gráfico propio de Pascal.

La conexión física de la red donde correrá el sistema será una red Ethernet de par trenzado nivel 5 en lo que son los nodos, ya que estos, a su vez, se encuentran conectados a un **backbone** (cableado principal, en un cableado estructurado) de fibra óptica, ejemplificado en la figura III.3.1.

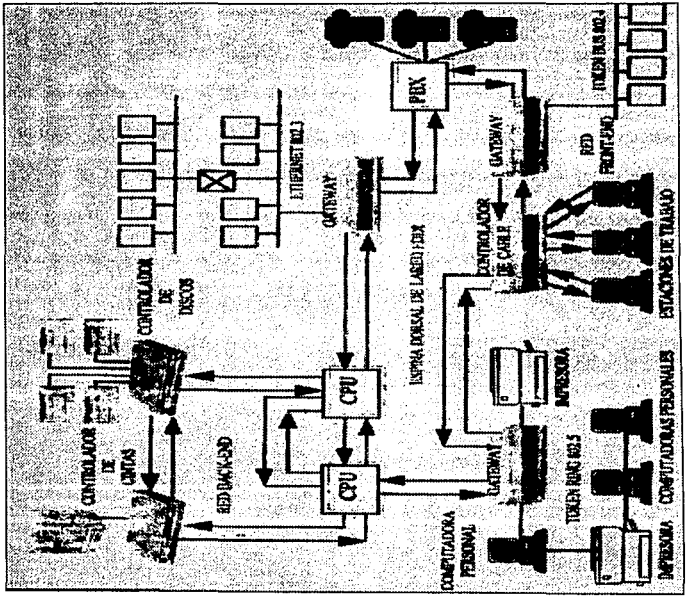


Figura III.3.1 Ejemplo de un cableado estructurado con un backbone.

Por otra parte, se debe considerar la posibilidad de trabajar el sistema administrativo en una computadora personal no conectada a la red solamente con sistema operativo DOS y de manera monousuario.

✓ ***Requerimientos en la operación del apartado de equipo***

En lo que se refiere a la operación cotidiana de apartado de equipo, el sistema deberá permitir al usuario (alumno), apartar un equipo de cómputo de manera sencilla, considerando las restricciones de un equipo al día y solo una hora. La información que proporcionará el alumno, será su número de matrícula, la hora en la que desea apartar el equipo y el tipo de equipo que desea apartar. Para lograr lo anterior, el operador del sistema (encargado del centro de Cálculo) contará con un menú de ayuda en el cual aparecerá en la hora solicitada el equipo que se encuentra disponible de acuerdo también al tipo de equipo que se desea, PC, MAC o Workstation. Cabe mencionar que los equipos con los cuales cuentan los alumnos para realizar sus prácticas y trabajos escolares son:

- 64 PS/Value Point de IBM con procesador DX a 33MHz.
- 48 PCMAC 7100 con procesador de 32 bits a 66MHz
- 64 Workstation RS/6000 de IBM Modelo 220

El sistema deberá permitir llevar un registro de todos los tiempos apartados y que en realidad fueron utilizados, así como los que tuvieron que ser reasignados por falta de la presencia del interesado que aparto el tiempo. La información anterior permitirá llevar un registro estadístico en el cual se tenga el porcentaje de ausentismo y el porcentaje de solicitud en un determinado tipo de equipo, esto es, cual equipo es el que más se está solicitando por parte del alumnado, y cual es el periodo de mayor solicitud de equipos.

Una parte importante en el manejo del sistema es el mantenimiento de archivos del mismo. Por lo que el sistema deberá contar con un grupo de programas de ayuda para el mantenimiento de la base de datos, reindexaciones, construcciones de índices, y una opción de respaldo de la información, así como una de restauración de información respaldada.

Para mantener la base de datos de alumnos inscritos en el periodo vigente, la sección de servicios escolares proporcionará la lista respectiva del semestre, sin embargo el sistema deberá ser capaz de permitir el ingreso de usuarios extra, como profesores, usuarios externos, etc.

✓ ***Requerimientos de interface gráfica***

Ya que en la actualidad las interfaces gráficas están ligadas a los nuevos ambientes de trabajo como el de Windows de Microsoft. El sistema deberá permitir trabajar con una interface gráfica de este tipo, permitiendo una interacción directa con Windows a partir de una herramienta como lo es Visual Basic. Esta herramienta es sumamente versátil, y se utilizará como el medio para dar al usuario un nuevo Front End, diferente al manejado de manera directa con Turbo Pascal.

Este Front End estará conformado por botones, despliegue de información por medio de ventanas contando con barras de desplazamiento y siendo el sistema iniciado desde un **icono** (elemento de ejecución dentro de Windows) en un grupo determinado de Windows. Con estas características tendremos la posibilidad de hacer uso de las ventajas proporcionadas por el btrieve en el manejo de archivos y al mismo tiempo una aplicación con la presentación gráfica amigable de Windows. Como ejemplo de una pantalla de este tipo tenemos la figura III.3.3.

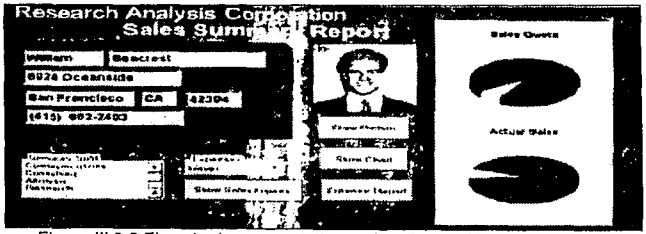


Figura III.3.3 Ejemplo de una pantalla con interface gráfica de Windows.

III.4 ANALISIS

La construcción de una base de datos de un nivel aceptable de funcionamiento para sus usuarios es una tarea compleja. Normalmente deben cubrirse dos grandes *etapas*:

- **Análisis** (o diseño lógico). Contempla identificar los *requerimientos* a ser procesados en el sistema de base de datos y la definición de las *estructuras de datos* necesarias para programarlos. En esta etapa también se identifican factores importantes que influyen en el procesamiento de los datos como lo es la *redundancia* y *consistencia* de información, y *seguridad* e *integridad* del sistema.

- **Programación** (o diseño físico). Toma los resultados del análisis: una estructura lógica de almacenamiento que puede ser procesada por cualquier manejador de bases de datos (DBMS). En esta etapa se programan los *métodos de acceso* y los *programas de aplicación* de los datos.

En lo que resta del presente capítulo se trata en detalle el proceso de análisis, reservando la etapa de programación al Capítulo IV (Desarrollo del Sistema).

III.4.1 ESPECIFICACION DEL ANALISIS

Las entradas y resultados que pueden considerarse dentro de un proceso de análisis de un sistema de base de datos son las siguientes:

Entradas:

- Requerimientos de información general.
- Requerimientos de procesamiento.
- Especificaciones del DBMS.
- Configuración de hardware y sistema operativo.
- Especificaciones de los programas de aplicación.

Salidas:

- Estructura lógica de la base de datos (vista por el usuario).
- Estructura de almacenamiento (vista del programador).

Los elementos anteriores se ilustran en la figura III.4.1.1. Trabajar con ellos requiere:

- Un análisis que consiste en una serie de pasos donde se escoge una alternativa de entre varias presentadas.
- Técnicas de análisis para realizar la identificación de criterios de evaluación para seleccionar alternativas en cada paso.
- Herramientas de modelado para describir entradas y salidas, así como sus relaciones en cada paso del análisis.

Los requerimientos de información general son las ideas, descripciones, comentarios, observaciones y en general toda clase de políticas de varios usuarios de la organización acerca de los objetivos de la base de datos y las distintas vistas que se desean tener de ella. Toda clase de propuesta debe considerarse independiente del DBMS, en caso contrario, se disminuye la adaptabilidad y consistencia de las soluciones de los requerimientos. Debe de aceptarse que el papel de una herramienta de programación es ser un medio que permite implementar las *características* de un sistema de cómputo.

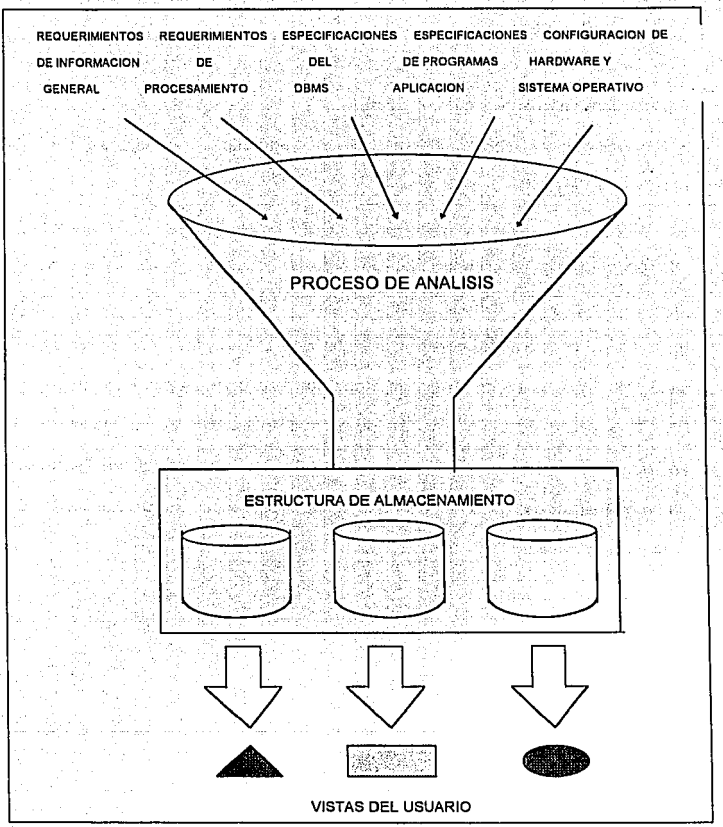


Figura III.4.1.1 Elementos del proceso de análisis

Para el procesamiento de los requerimientos debe considerarse lo siguiente

- Los datos que se requieren en cada aplicación.
- El volumen de datos y su crecimiento esperado.
- La frecuencia de procesamiento en términos del número de ocasiones en que cada aplicación debe ejecutarse por unidad de tiempo.

Como complemento a lo anteriormente mencionado, influyen las *restricciones* y la *capacidad* de desempeño que se presentan en el sistema. Es decir, los tiempos de respuesta, la recuperación de información en caso de fallas, o datos específicos necesarios para requerimientos de seguridad o integridad.

En la fase de análisis se identifican los requerimientos del sistema, lo cual involucra el establecimiento de los objetivos y la documentación de estos. En el análisis del sistema colaboran usuarios, desarrolladores y administradores del sistema.

Es recomendable que los objetivos específicos y los requerimientos de la base de datos a realizar, deben obtenerse de un nivel alto de la organización de los usuarios.

De esta forma, el equipo de análisis de datos debe realizar entrevistas personales con diferentes niveles de gerencia y empleados claves en el procesamiento de bienes,

servicio y organización de datos. El resultado de tales entrevistas deben ser diagramas de proceso.

El análisis de los requerimientos de un sistema normalmente llevan también, a la elaboración de un diagrama entidad-relación. En éste, se identifican las entidades (o tablas) y sus relaciones que son necesarias construir para resolver el problema definido por los usuarios. Los siguientes puntos detallan los retos y acciones que implican la elaboración de un diagrama entidad-relación.

CONSTRUCCION DEL DIAGRAMA ENTIDAD-RELACION

Como se ha explicado, el diagrama entidad-relación es una representación del sistema. Se trata esencialmente de una vista del mundo real de los datos organizados en términos de entidades, atributos y relaciones. Este nos permite incluir aquellas entidades que se requieren para un problema particular de procesamiento de datos. Durante las etapas iniciales es posible que no se conozcan todos los atributos de las entidades. Sin embargo, a medida que los atributos se determinan, se debe documentar la definición de cada uno en el diccionario de datos.

Prácticamente la primera versión de un diagrama entidad-relación puede calificarse como intuitiva, es decir, se trata sólo de una idea que puede ser mejorada. A medida que se avanza en el análisis se definen las relaciones entre entidades, las cuales, son

revisadas, modificadas, creadas y eliminadas paulatinamente como resultado del conocimiento adquirido sobre los requerimientos del sistema. Un análisis serio y profundo permite:

- Reducir redundancia en las relaciones.
- Determinar cuales entidades son significativas en el sistema y en los requerimientos del usuario.
- Resolver relaciones no binarias entre entidades.

ETAPAS DE INTEGRACION DE UN DIAGRAMA ENTIDAD-RELACION

Las etapas requeridas para integrar un diagrama entidad-relación son las siguientes: Identificar cada sinónimo u homónimo en los diferentes modelos. Esta tarea es sencilla si se usa el diccionario de datos. Los componentes con homónimos deben ser renombrados. Los componentes con sinónimos deben usar el mismo nombre.

Los diagramas entidad-relación de áreas diferentes se integran superponiendo los tipos de entidad que sean idénticos o similares en ambos diagramas. Esto puede incrementar el número total de atributos del tipo de entidad, ya que entidades idénticas pueden usar diferentes atributos.

Como resultado de la integración, el diagrama resultante puede contener relaciones redundantes. Esta redundancia debe ser eliminada realizando un análisis adicional sobre la interacción entre ambos, en donde se definen los datos que son comunes y necesarios entre sí para sus respectivos procesos a ejecutar.

DERIVACION DE ARCHIVOS LOGICOS A UN DIAGRAMA ENTIDAD-RELACION

En realidad no existen reglas para esta derivación. La distribución de datos, con los cuales los archivos lógicos son construidos pueden ser la misma que se requiere para las respectivas entidades. Sin embargo, los siguientes procedimientos pueden seguirse cuando se trate de convertir archivos planos a un diagrama entidad-relación:

- Listar todos los tipos de archivos en los programas relevantes.
- Listar todos los registros físicos en los archivos.
- Listar todos los datos en los registros.
- Eliminar las redundancias e inconsistencias en los datos y los registros lógicos.
- Listar todas las combinaciones posibles de entidades de los registros lógicos. El nombre del registro es un indicador de una entidad.
- Realizar un análisis preliminar de los datos.
- Acomodar los atributos con sus respectivas entidades.

Estos procedimientos dan como resultado un diagrama entidad-relación que sirve como estructura preliminar, la cual, como se recomendó anteriormente, es sometida a otras revisiones, las cuales son necesarias para un análisis más concreto.

COMBINACION DE DIAGRAMAS ENTIDAD-RELACION

Al convertir bases de datos existentes en su equivalente de diagrama entidad-relación, es común tener diferentes diagramas de acuerdo a los programas o aplicaciones de los cuales los diagramas fueron derivados. Para este caso, se debe intentar eliminar las redundancias e inconsistencias al combinar los diagramas, y esto debe conducir a un diagrama entidad-relación integrado, el cual, permite determinar lo siguiente:

- Cuáles son las entidades y atributos comunes.
- Cuáles son las inconsistencias y redundancias en los nombres y uso de atributos, las cuales existen cuando se observa que dos entidades con diferentes nombres son la misma.
- Si algunos atributos son considerados en varias entidades.
- La existencia de relaciones innecesarias entre las entidades.

El diagrama que se consigue integrar puede ser usado como la estructura de un análisis global en lugar de otros pequeños orientados a aplicaciones específicas.

III.4.2 DIAGRAMA DE DESCOMPOSICION FUNCIONAL.

Puede ser difícil comprender en su totalidad un proceso de un sistema si se emplea para ello sólo una descripción verbal. Sin embargo, a través de las herramientas de modelado de flujo de datos es posible representar los componentes esenciales de un sistema y las interacciones entre ellos .

De esta forma, siguiendo la estrategia de un diagrama de flujo de datos, un *diagrama de descomposición funcional* muestra las características esenciales de un sistema y la forma en que se ajustan entre sí.

El diagrama de descomposición funcional muestra las partes fundamentales de la estructura del sistema. En el se muestra las diferentes entidades y como podemos llegar a ellas de acuerdo al flujo de información ya sea manual o automatizada, incluyendo procesos y retroceso de información.

Personas ajenas al estudio del sistema pueden comprender con un diagrama sencillo el funcionamiento de éste. Por consiguiente, los analistas pueden trabajar con los usuarios y lograr que participen en el estudio de los diagramas del flujo de los datos.

Los usuarios pueden hacer sugerencias para modificar los diagramas de flujo de datos. También pueden hacer sugerencias para modificar los diagramas con la finalidad de describir la actividad con mayor exactitud. Así mismo al examinar los diagramas de descomposición funcional es posible reconocer con rapidez problemas de un nivel de abstracción superior; y esto permite efectuar las correcciones necesarias antes de que comiencen otras tareas relacionadas con el proceso de análisis.

En la figura III.4.2.1 se muestra el diagrama de descomposición de nuestro sistema, en él se pueden apreciar los diferentes módulos de funcionamiento y su flujo de datos. El diagrama muestra que existe un Menú Principal, y se tienen submenus para cada módulo de éste.

Existen cuatro niveles de acceso al sistema:

- **Nivel 1.** El usuario tiene acceso a todas las opciones del *Menú Principal.*
- **Nivel 2.** El usuario tiene acceso a los módulos de Mantenimiento y Salir incluyendo sus correspondientes submenus.
- **Nivel 3.** El usuario tiene acceso a los módulos de *Apartados, Consultas* y *Salir* incluyendo sus correspondientes submenus.
- **Nivel 4.** El usuario tiene acceso a los módulos de *Consultas* y *Salir* con sus correspondientes submenus; y sólo al *submenú de apartados* del módulo de *Apartados.*

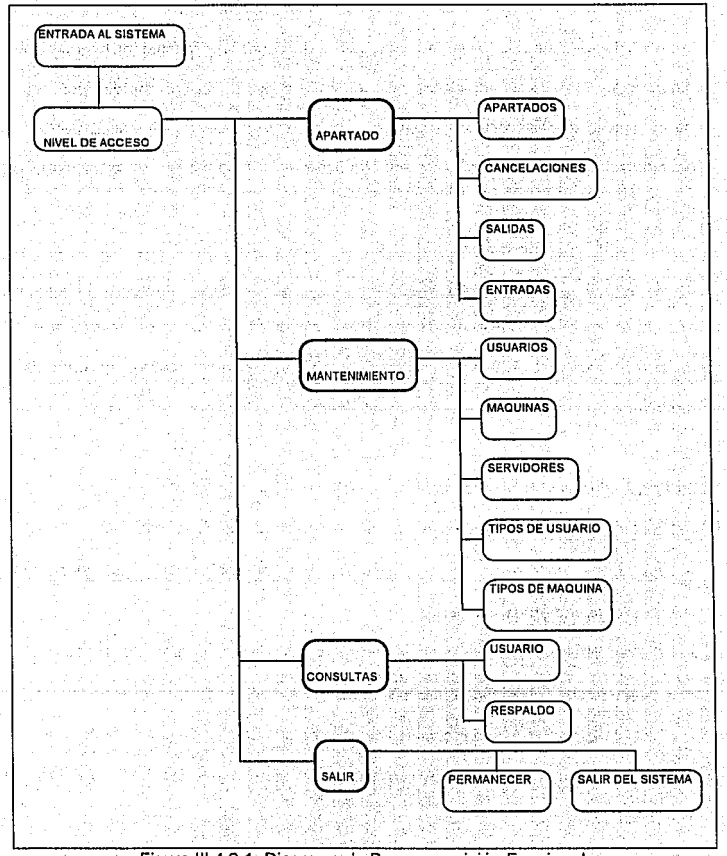


Figura III.4.2.1 Diagrama de Descomposición Funcional.

III.5 OPCIONES DE SOLUCION

En la actualidad, debido al auge que ha tenido el modelo relacional de datos y el poder que se ha logrado proporcionar a las computadoras personales en cuanto a rapidez, capacidad de almacenamiento y conectividad a mainframes a través de las redes locales, el mercado tecnológico ofrece una serie de productos de desarrollo orientados a la organización y al manejo de datos para estas, dependiendo del ambiente donde estén interactuando, teniendo como base el Modelo de Datos Relacional. Esto implica una mayor posibilidad de encontrar algún producto que satisfaga con mayor exactitud las necesidades de los usuarios.

CRITERIOS DE EVALUACION

A continuación, se mostrarán las características consideradas en la selección de la herramienta de software para el desarrollo del sistema. Los criterios de evaluación se obtuvieron considerando las siguientes fuentes de información:

- Revistas especializadas.
- Experiencias de los usuarios.

De estas se obtuvieron características generales y posteriormente se propusieron criterios de selección con base a los requerimientos del sistema.

Para llevar a cabo la selección, se consideraron los siguientes aspectos generales:

- Facilidad de aprendizaje.
- Facilidad de uso.
- Recursos de programación.
- Campos de datos
- Manejo de datos.
- Manejo de reportes
- Poder de programación
- Niveles de Seguridad.
- Precio

FACILIDAD DE APRENDIZAJE.

Un factor importante es la claridad en la documentación incluida así como la claridad de los tutoriales. Tomando en cuenta que la mayoría de los usuarios de estos productos no son expertos en computación, un factor importante a considerar es la claridad en sus manuales.

FACILIDAD DE USO

Una vez que el usuario aprenda a usar el programa, los manuales se vuelven menos importantes que la interfaz de usuario que le pueda proporcionar el producto.

La mejor interfaz de usuario puede considerarse la que presenta las opciones mediante menús que pueden recorrerse y cambiar de uno a otro. Además es ideal que permita el uso de un ratón para trabajar aún más cómodamente, siendo este aspecto no relevante ya que se puede prescindir de él.

Los diversos productos comerciales poseen interfaces que oscilan desde adecuadas hasta excelentes. Lo importante a considerar en este aspecto es que utilizando el producto, no exista la dificultad de recordar cuales son las funciones asignadas a las teclas, o bien tener la consideración de haber incluido alguna plantilla que indique las funciones de cada una de ellas para colocar sobre el teclado.

Algunos manejadores permiten definir queries con una serie de fórmulas cortas que usan operadores lógicos tales como AND y OR.

La complejidad de las fórmulas disminuye debido a la sencillez de los queries que involucran uno, dos o tres campos; pero si se desea obtener datos desde diversos

archivos relacionados, se obtendrá un ahorro en tiempo mediante la técnica de "query by example" (QBE) disponible en algunos productos. QBE permite mostrar el tipo de información que se desea llenando en tablas temporales el ejemplo para el "query".

RECURSOS DE PROGRAMACION.

Un factor importante es que el **RDBMS** (Relational Data Base Managment System) proporcione facilidades para poder construir aplicaciones complejas, mediante la creación de pantallas para entrada de datos, incluyendo validaciones, atributos de color despliegue de mensajes de ayuda, etc.

La mayoría de los sistemas manejadores de base de datos requieren una cantidad considerable de programación para control del manejo de los datos. Esto implica que el tiempo invertido en el desarrollo de implementación del sistema se prolongue, debido a esto, no solo es importante contar con flexibilidad en la captura, sino también contar con flexibilidad en la creación de la aplicación que manipula a los datos tanto en consultas como en reportes.

CAMPOS DE DATOS

Un aspecto importante para todo desarrollador es el poder definir campos de longitud variable, lo cual permite obtener un ahorro considerable de espacio en disco por no

almacenar los espacios después de nombre cortos o líneas en blanco.

MANEJO DE DATOS

Cualquier manejador de base de datos relacional debe permitir realizar cálculos sobre los datos y desplegar los resultados en formas y reportes, y exportar e importar datos de otros RDBMS o de otros productos de software como las hojas de cálculo.

MANEJO DE REPORTES

A través de un acceso a datos y de un formato de salida predefinido por los usuarios finales se producen reportes complejos; este es un sello característico de todo buen sistema manejador de base de datos relacional.

NIVELES DE SEGURIDAD

Dado que la mayoría de los productos permiten optimizar la aparición de menús dependiendo del usuario que use la aplicación, el desarrollador puede hacer uso de niveles de seguridad. De esta forma, se puede restringir el acceso a ciertos menús del sistema, evitando que algunos usuarios no obtengan datos y evitar que estos sufran alteraciones o sean eliminados.

Se requieren cuidados extremos en la protección de información importante, entonces es necesario usar un manejador que permita el acceso a cierta información mediante claves de acceso.

PRECIO

Es uno de los factores determinantes en la elección no sólo de un RDBMS en particular sino de cualquier software en general. Claro está que el precio también depende en gran medida de las capacidades que este mismo ofrece. Es importante considerar si no se están adquiriendo capacidades de más y que finalmente no serán utilizadas para el desarrollo de las aplicaciones.

HERRAMIENTAS DE SOFTWARE

En el desarrollo de nuestro sistema es necesario manejar una herramienta que nos proporcione tener una aplicación en un ambiente gráfico, como lo son *Visual Basic* y *Visual C++*; También es necesario utilizar un manejador de bases de datos para tener acceso a nuestra base de datos, en este caso consideramos *Oracle*, *Clipper* e *Informix-SQL* y *Btrive*.

Las características de *Visual Basic* y *Btrive* ya se han mencionado en el capítulo II. A continuación se enuncian las características de las otras herramientas.

VISUAL C++

A través de *Visual C++* es posible elaborar aplicaciones bajo ambiente Windows.

Algunas de sus ventajas son: además del manejo de programación C++, tiene mayores beneficios en cuanto a la velocidad y capacidades de manejo de memoria que *Visual Basic*, al trabajar en un ambiente Windows tiene una gran variedad de opciones de interfaces y desempeño como lo es el manejo de iconos y ventanas para el desarrollo de sistemas más amigables, seguros y confiables, de fácil manejo para el usuario en general, y sobre todo, de calidad.

CLIPPER

Se enfoca esencialmente hacia programadores. Si bien carece de algunas de las excelentes funciones de generación de códigos que tienen los demás paquetes, Clipper ofrece una gran variedad de elementos de programación para sus desarrolladores.

Entre los elementos de programación que distinguen a ***Clipper*** se encuentran su generador de reportes (RL) y una función (DBU) para la creación y manejo de los archivos de la base de datos, escritos en el propio lenguaje de programación de

Clipper, y se incluye el código fuente, el cual puede utilizarse como referencia, o modificarlo para añadirlo a las aplicaciones de base de datos.

La pantalla de DBU enlista las opciones a través de su parte superior, junto con las teclas de funciones asignadas. El resto de la pantalla está dedicado a una representación visual del panorama de la base de datos activa, en donde es posible identificar los índices activos y los nombres de los campos. El generador de reportes RL no es flexible, y no permite ver la salida en forma preliminar conforme se va generando.

El compilador del programa es muy veloz y ofrece varias opciones de compilado, destacando las siguientes: soporta los llamados en bloque de código, secciones de código ejecutable que se pueden almacenar como variables, o pasar a otros programas como argumentos. *Clipper* puede leer archivos binarios de DOS y escribir en ellos. Cuenta con un depurador para analizar el funcionamiento del programa.

ORACLE

Hasta hace poco tiempo, *ORACLE* se caracterizó por ser utilizado en mainframes de grandes corporaciones y de oficinas gubernamentales. Es el manejador de bases de datos que a marcado el punto de referencia en el desarrollo de sus competidores.

Cuenta con una gran variedad de herramientas para el manejo de datos, todas ellas se integran para el desarrollo y explotación de bases de datos. Su calidad es alta. Desafortunadamente también son altos sus requerimientos de instalación y funcionamiento, así como su precio.

INFORMIX

Se trata de un manejador de bases de datos con un lenguaje de consulta estructurado (SQL).

Las pantallas de éste, son de estilo Lotus, manipuladas por menús, automatizan las operaciones más significativas de la base de datos, tales como la creación de tablas, la definición y modificación de los campos, etc. ***Informix*** también incluye un generador de reportes, y un sistema para la ejecución de archivos de la definición de formas diseñadas.

En ***Informix*** se verifican la integridad de los índices en caso de encontrarse una discrepancia entre un archivo de datos y uno de sus índices, y de ser así, el índice será reformado.

SELECCION DE HERRAMIENTAS DE SOFTWARE

A través de los criterios y opciones mencionados en la presente sección, la decisión de las herramientas de software de nuestro sistema se determinaron en base a los requerimientos de la etapa de desarrollo del sistema, ya que al usuario no le es relevante de que forma se le presenta la visión que tiene del sistema.

La herramienta de software que utilizamos, principalmente debe ser fácil de aprender y de usar, debe permitir manejar datos en distintas formas, dar seguridad a la información, funcionar en ambiente de red y ser de precio accesible.

Nuestras selecciones son *Visual Basic* y *Btrive*. El Centro de Cálculo cuenta con una red Novell, la cual interactua con *Btrive* en forma nativa y sin un costo adicional. La complejidad del ambiente gráfico puede ser resuelta satisfactoriamente con *Visual Basic* versión 3.1, ya que no son necesarios los conceptos de programación orientada a objetos.

Capítulo IV

Desarrollo del Sistema

IV. DESARROLLO DEL SISTEMA

IV.1 DISEÑO

IV.1.1 ESPECIFICACIONES DE DISEÑO

Se desarrollará un sistema para optimizar el uso, apartado, disponibilidad y estadísticas de uso de los equipos de cómputo en el C.E.C. el cual podrá ejecutarse en red o localmente en una estación.

El uso y apartado se otorgará por hora a los usuarios que se encuentren registrados en el catálogo de usuarios.

La seguridad de acceso se basará en el password del usuario.

El sistema se podrá operar con teclado o con Mouse.

Se utilizará BTRIVE para que el sistema pueda correr en red y él sea quien maneje el acceso a los archivos, acceso a usuarios, etc.

El sistema contará con lo siguientes módulos:

- Apartado
- Mantenimiento
- Consulta

Los módulos se dividirán de la siguiente manera:

- Apartado
  - Apartado
  - Cancelación
  - Entradas
  - Salidas

- Mantenimiento
  - Usuarios
    - Altas
    - Bajas
    - Cambios
    - Consultas

Estaciones

- Altas
- Bajas
- Cambios
- Consultas

Servidores

- Altas
- Bajas
- Cambios
- Consultas

Tipos de usuarios

- Altas
- Bajas
- Cambios
- Consultas

Tipos de estaciones

- Altas
- Bajas

Cambios

Consultas

- Consultas

  Usuario

  Respaldo

En el módulo de apartado se podrá realizar el apartado de equipo, cancelar un apartado previo, registrar el uso del equipo y la finalización del mismo. Este módulo tendrá el siguiente submenú:

- Apartado
- Cancelaciones
- Entrada

En apartado se reservará el uso de equipo de acuerdo a la disponibilidad del mismo y se solicitará la matrícula del alumno que requiere el servicio, el tipo de estación a solicitar y la hora en que se desea usar.

En cancelaciones se realizará la cancelación de un apartado previo solicitando la matrícula del alumno.

En entrada se registrará y permitirá el acceso al uso del equipo para determinada hora verificando que el alumno esté dado de alta en el catálogo de usuarios . Se manejarán dos formas de entrada:

Con apartado previo: se verifica que exista un apartado previo y se registra el acceso del alumno al C.E.C mediante la matrícula del alumno, se verificará que se este en el rango del minuto 0 al minuto 15 de la hora de apartado.

Sin apartado: en el caso de que no exista apartado previo y se tenga disponibilidad de equipo se podrá registrar la entrada del alumno mediante la matrícula de éste, esto se podrá realizar del minuto 0 al minuto 45.

En salida se registrará la salida del alumno del C.E.C. mediante la matrícula del alumno.

En el módulo de mantenimiento se podrá realizar la actualización y consulta a los catálogos del sistema. Este módulo tendrá el siguiente submenú:

- Usuarios
- Estaciones
- Servidores
- Tipos de usuarios
- Tipos de estaciones

En el catálogo de usuarios se tendrán las siguientes opciones:

- Altas
- Bajas
- Cambios
- Consultas

En altas se solicitará el número de matrícula del usuario a dar de alta y se verificará que no exista en el catálogo de usuarios, si no existe procede el alta y si existe se desplegará un mensaje el cual indicará que el usuario ya está dado de alta. Para dar de alta un usuario se solicitaran los siguientes datos:

- Número de matrícula
- Apellido paterno
- Apellido materno
- Nombre (s)
- Tipo de usuario
- Carrera

En bajas se solicita el número de matrícula del usuario a eliminar del catálogo, se verificará que exista el usuario en el catálogo, si existe se confirma la baja y se elimina del catálogo.

En cambios se solicita el número de matrícula del usuario, se verifica que exista en el catálogo y se permitirán desplegarán los datos que se tienen capturados con la opción de modificación, los datos son:

- Apellido paterno
- Apellido materno
- Nombre
- Tipo de usuario
- Carrera

En consultas se desplegará la información del usuario en base al número de matrícula.

En el catálogo de estaciones se tendrán las siguientes opciones:

- Altas
- Bajas
- Cambios
- Consultas

En altas se preguntará si se quiere dar de alta nuevas estaciones, si la respuesta es negativa se termina el requerimiento y se presenta la opción de estaciones, si la respuesta es afirmativa se solicita la siguiente información:

- Número de estación
- Tipo de estación
- Número de serie de CPU
- Número de serie del teclado
- Número de serie del monitor
- Número de serie del Mouse
- Modelo
- Capacidad de memoria RAM
- Capacidad en disco duro
- Tipo de microprocesador
- Capacidad del disco flexible
- Número de servidor
- Número de isla
- Servicio

El número de isla es el número correspondiente a la ubicación física de la estación y el rango es de 1 a 12.

El número de servidor es el servidor al que está conectada esa estación.

El servicio es una 'S' para ponerla a disponibilidad de uso o una 'N' para indicar que la estación no está en servicio.

En bajas se solicita el número de estación, se verifica que exista si no existe se despliega mensaje de no existe estación y se habilita la opción de estaciones, si existe la estación se elimina la estación del catálogo de estaciones.

En cambios se solicita el número de estación, se verifica que exista si no existe se despliega mensaje de no existe estación y se habilita la opción de estaciones, si existe se despliega la información que contiene el catálogo con la opción de modificar datos.

En consultas se solicita la clave de la estación y se presenta la información

El catálogo de servidores tendrá las siguientes opciones:

- Altas
- Bajas
- Cambios
- Consultas

En altas se solicitara la siguiente información:

- Número de servidor
- Nombre
- Dirección de IP
- Dirección de Ethernet

Se verificará que no exista el servidor que se está dando de alta y si no existe se efectúa la alta al catálogo.

En cambios se solicita el número de servidor a dar de baja, se confirma que exista en el catálogo, si no existe se habilita la opción de servidores, si existe se procede a eliminar el servidor del catálogo.

En consultas se solicitará el número de servidor a consultar, se verificará que exista, si no existe se habilita la opción de servidores, si existe se presentará la información del servidor solicitado.

En el catálogo de tipos de usuario se tendrán las siguientes opciones:

- Altas
- Bajas
- Cambios
- Consultas

En altas se solicitará la siguiente información:

- Clave de tipo
- Descripción

Se verificará que no exista el tipo de usuario a dar de alta para actualizar el catálogo de tipos de usuarios si ya existe se desplegará el mensaje de "Tipo de usuario ya está dado de alta".

En bajas se solicitará la clave de tipo de usuario, se verifica que exista en el catálogo, si no existe se presenta un mensaje de "No existe este tipo de usuario" y se habilita la opción de tipos de usuario, si existe se eliminará del catálogo de tipos de usuario y se presentará el mensaje "Tipo de usuario borrado" habilitándose la opción de tipos de usuario.

En cambios se solicitará la clave de tipo de usuario, se verificará que exista en el catálogo, si no existe se presentará un mensaje de "No existe esta tipo de usuario" y se habilita la opción de tipos de usuario, si existe se presentan los datos que se tienen en el catálogo con la opción de modificar los datos y se actualizarán los datos del catálogo si la respuesta es negativa se habilita la opción de tipos de usuario.

En consultas se solicitará la clave de tipo de usuario, se verificará que exista en el catálogo, si no existe se presentará un mensaje de "No existe ese tipo de usuario" y se habilita la opción de tipos de usuario, si existe presentan los datos que tiene el catálogo de usuarios para el usuario en especifico.

En el catálogo de tipos de estaciones tendrán las siguientes opciones:

- Altas
- Bajas
- Cambios
- Consultas

En altas se solicitará la siguiente información:

- Clave de tipo
- Descripción

Se verificará que no exista el tipo de estación a dar de alta para actualizar el catálogo de tipos de estación si ya existe se desplegará el mensaje de "Tipo de estación ya está dado de alta".

En bajas se solicitará la clave de tipo de estación, se verificará que exista en el catálogo, si no existe se presentará un mensaje de "No existe este tipo de estación" y se habilita la opción de tipos de estación, si existe se eliminará del catálogo de tipos de estación y se presentará el mensaje "Tipo de estación borrado" habilitándose la opción de tipos de estación.

En cambios se solicitará la clave de tipo de estación, se verificará que exista en el catálogo, si no existe se presentará un mensaje de "No existe este tipo de estación" y se habilita la opción de tipos de estación, si existe se presentan los datos que se tienen en el catálogo con la opción de modificar los datos y se actualizarán los datos del catálogo.

En consultas se solicitará la clave de tipo de estación, se verificará que exista en el catálogo, si no existe se presentará un mensaje de "No existe este tipo de estación" y se habilitará la opción de tipos de estación, si existe se presentan los datos que tiene el catálogo de estaciones para la estación en especifico.

En el módulo de consultas se solicitará el número de matrícula a consultar y se proporcionará el número de estación en caso de que se tenga estación asignada y la hora que tiene apartada se tendrá la opción de realizar un respaldo en disco flexible.

Se estandarizaran las siguientes teclas para navegar en el sistema:

- ☑ **ESC** Para cancelar o regresar al menú anterior.
- ☑ **ENTER** Para pasar al siguiente campo a capturar o para indicar que se ha seleccionado una opción en el menú.
  combinación para actualizar la información del catálogo.

Los botones estándares en pantalla son:

☑ **CONTINUAR** Indica que se continúe con la operación respectiva.

☑ **REGRESAR** Regresar a la opción o pantalla anterior.

☑ **TERMINAR** Invoca al menú principal para poder terminar la operación.

Los botones anteriores se activarán con el botón izquierdo del mouse o con la tecla de "Enter".

La seguridad se dividirá en cuatro niveles de acceso los cuales son los siguientes:

- Acceso a todas las opciones del menú principal.
- Acceso al módulo de mantenimiento incluyendo los submenus correspondientes y al módulo de salir del sistema.
- Acceso al módulo de apartados incluyendo todos los submenus correspondientes, el módulo de consultas y al módulo de salir del sistema.
- Acceso al submenú de apartados dentro del módulo de apartados, al modulo de consultas y al módulo de salir del sistema.

La información que se almacenará para poder obtener estadísticas sobre el uso de equipos, horas pico de uso, estaciones de uso y ocupaciones que se tuvieron con apartado ó sin apartado, se explotará con otra herramienta y en está tesis no

describiremos como es que se hace la explotación ni cuales son los reportes que se pueden obtener, sólo mencionaremos que se deja un archivo el cual es tomado por EXCEL para poder generar sus reportes.

IV.1.2. DIAGRAMA DE FLUJO DE DATOS

A continuación se presentarán los diagramas de flujo de datos para el sistema de administración del centro de cálculo.

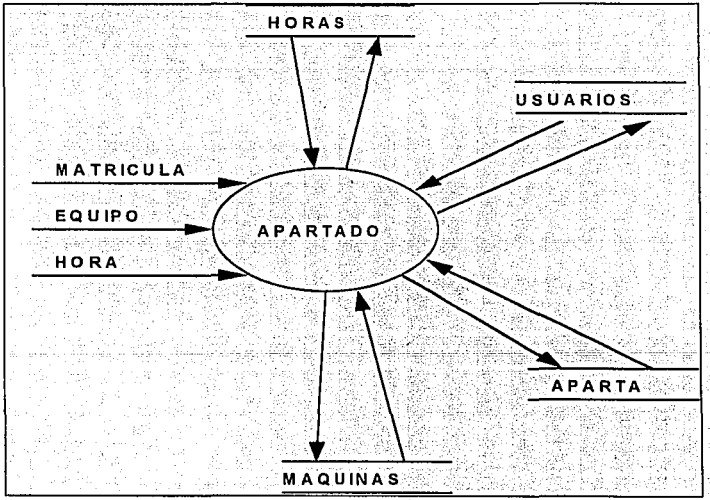


Figura IV.1.2.1 DFD Apartado de estaciones

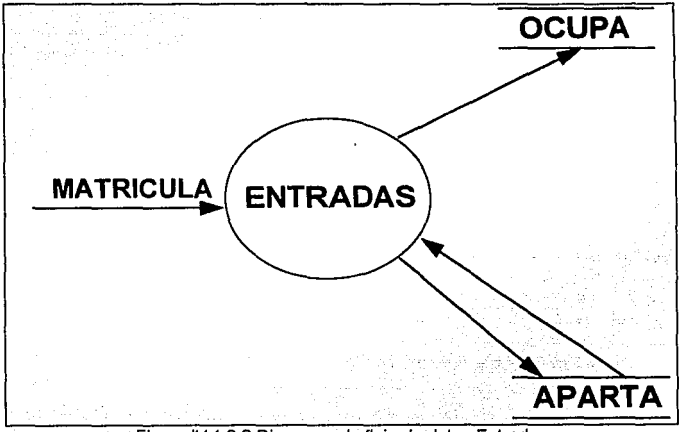


Figura IV.1.2.2 Diagrama de flujo de datos Entradas

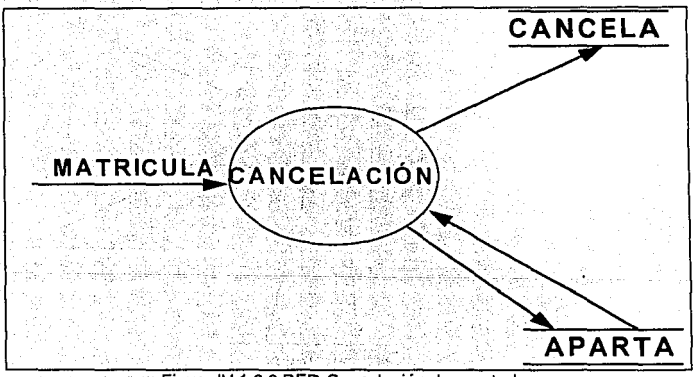


Figura IV.1.2.3 DFD Cancelación de apartados

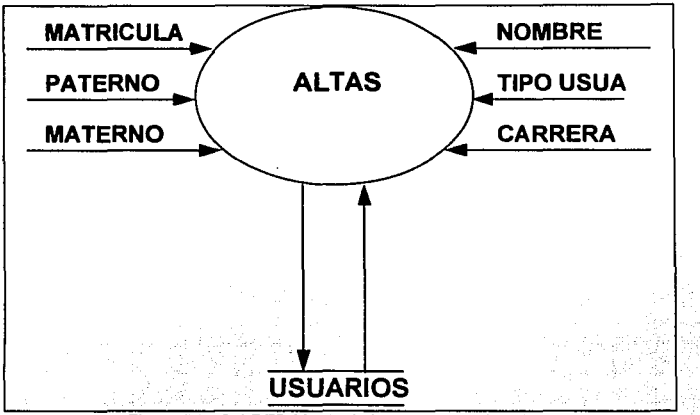


Figura IV.1.2.4 DFD Mantenimiento de usuarios altas

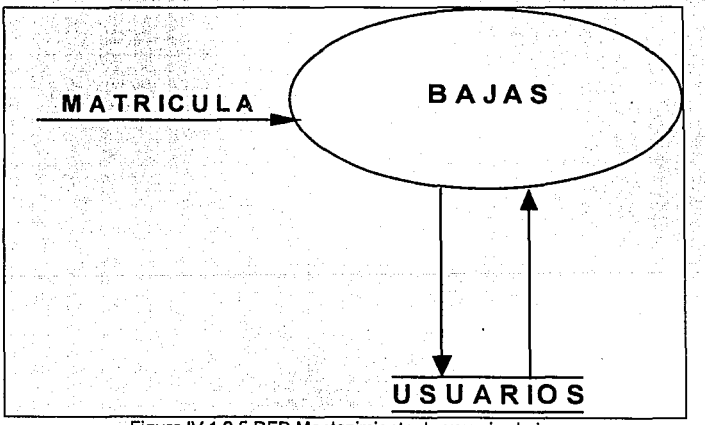


Figura IV.1.2.5 DFD Mantenimiento de usuarios bajas

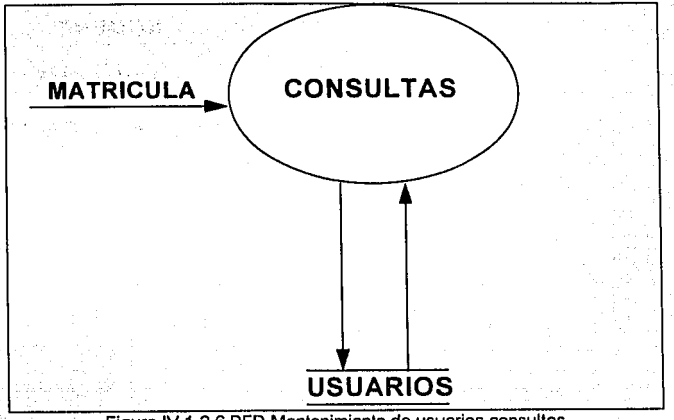


Figura IV.1.2.6 DFD Mantenimiento de usuarios consultas

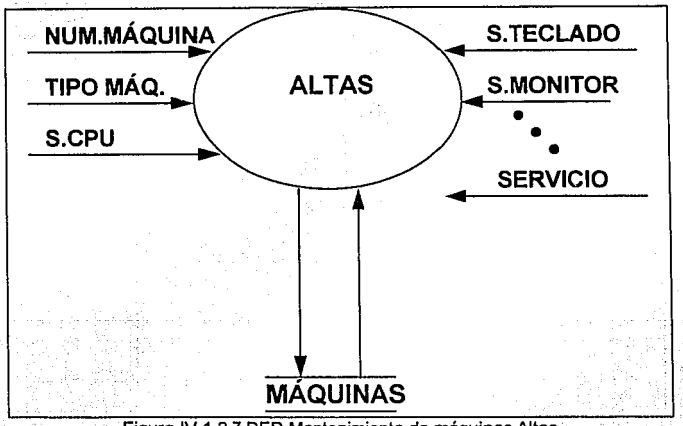


Figura IV.1.2.7 DFD Mantenimiento de máquinas Altas

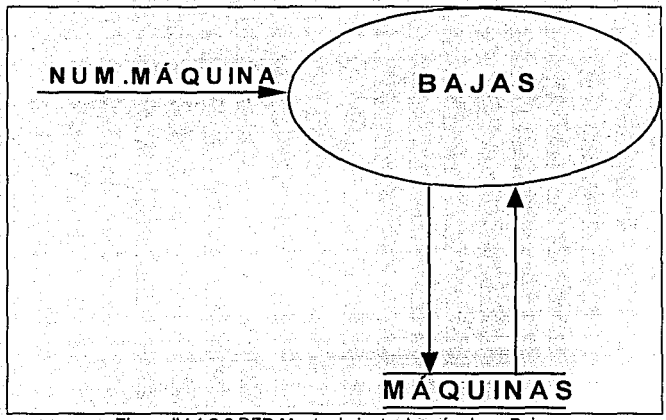


Figura IV.1.2.8 DFD Mantenimiento de máquinas Bajas

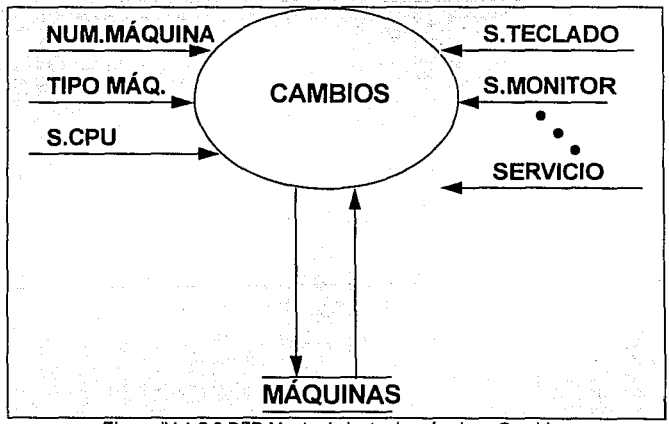


Figura IV.1.2.9 DFD Mantenimiento de máquinas Cambios

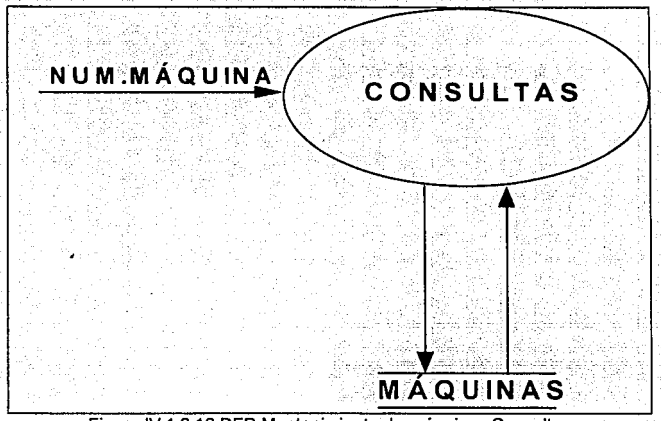


Figura IV.1.2.10 DFD Mantenimiento de máquinas Consultas

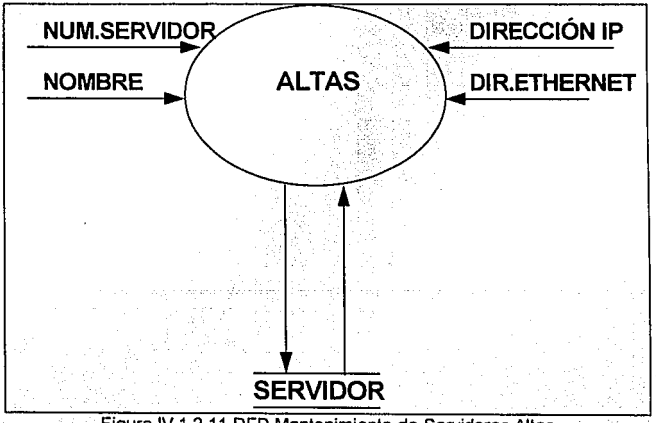


Figura IV.1.2.11 DFD Mantenimiento de Servidores Altas

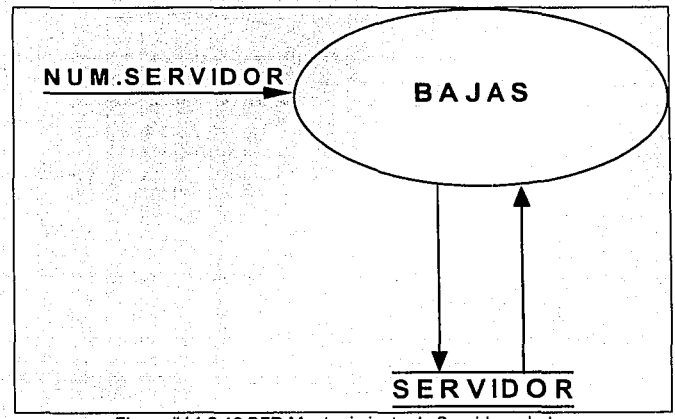


Figura IV.1.2.12 DFD Mantenimiento de Servidores bajas

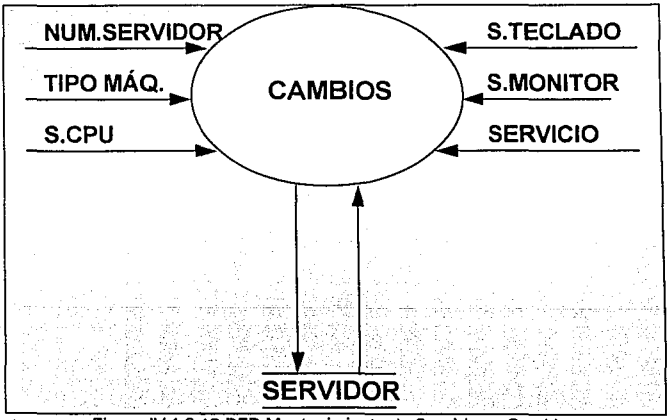


Figura IV.1.2.13 DFD Mantenimiento de Servidores Cambios

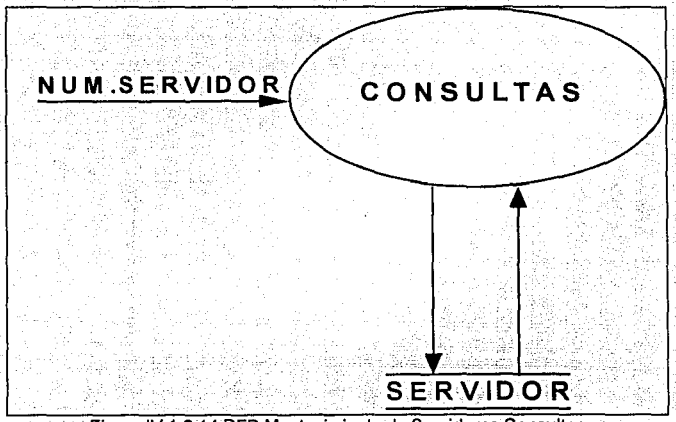


Figura IV.1.2.14 DFD Mantenimiento de Servidores Consultas

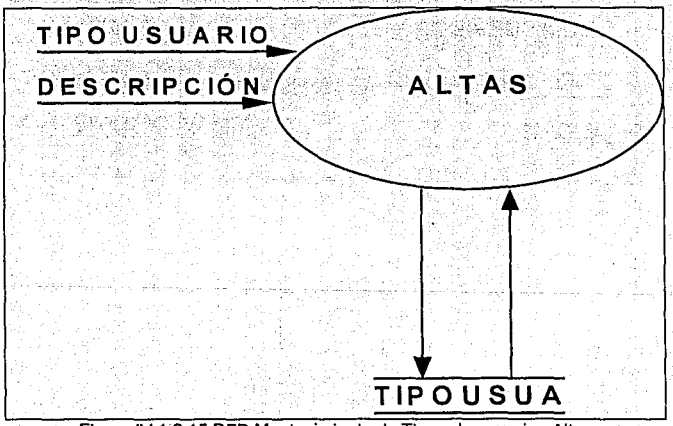


Figura IV.1.2.15 DFD Mantenimiento de Tipos de usuarios Altas

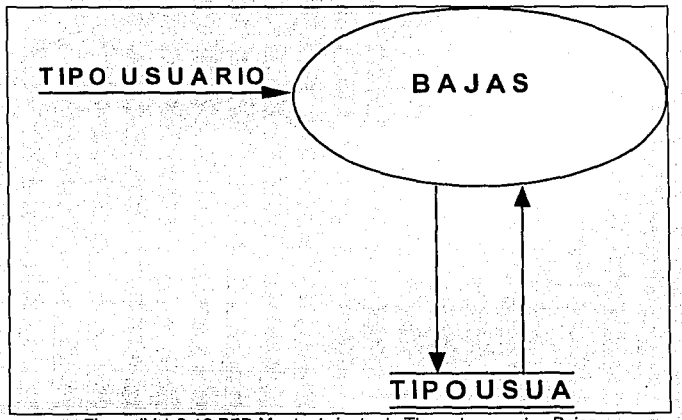


Figura IV.1.2.16 DFD Mantenimiento de Tipos de usuarios Bajas

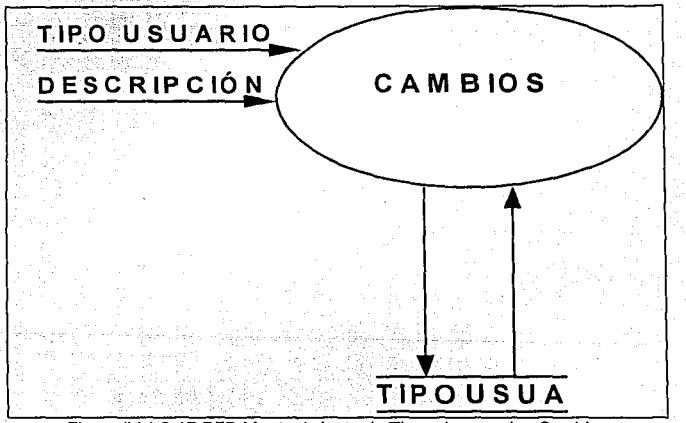


Figura IV.1.2.17 DFD Mantenimiento de Tipos de usuarios Cambios

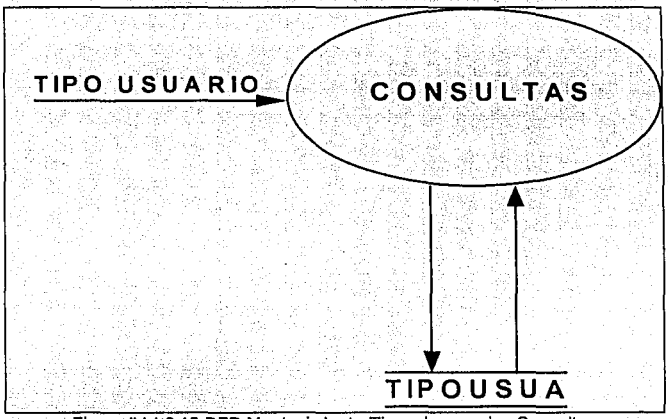


Figura IV.1.2.18 DFD Mantenimiento Tipos de usuarios Consultas

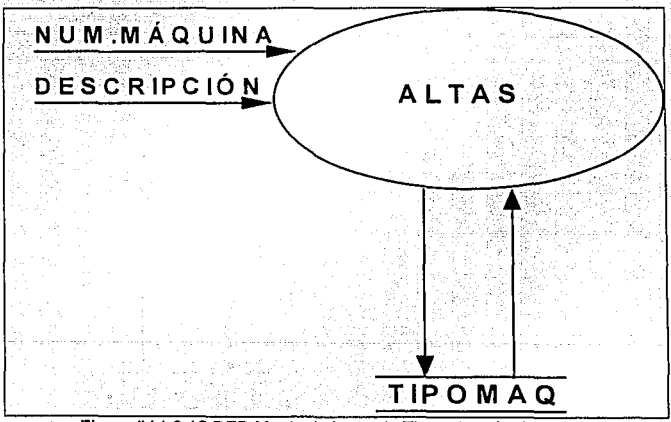


Figura IV.1.2.19 DFD Mantenimiento de Tipos de máquina Altas

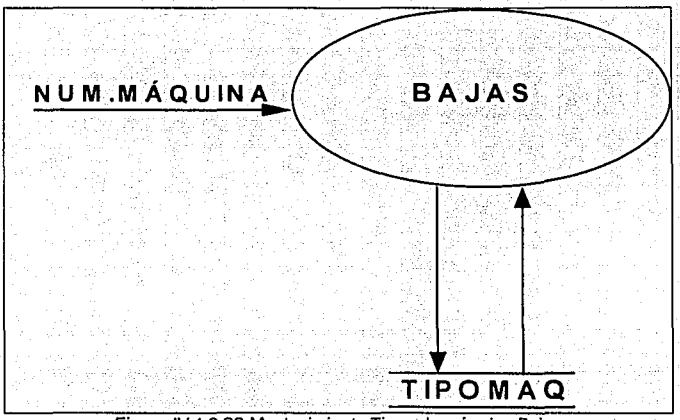


Figura IV.1.2.20 Mantenimiento Tipos de máquina Bajas

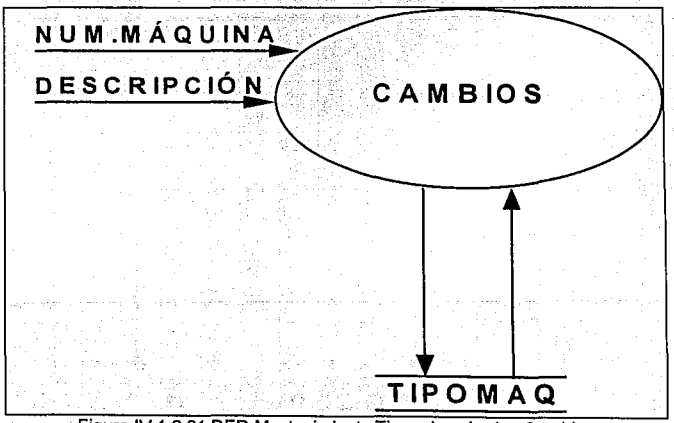


Figura IV.1.2.21 DFD Mantenimiento Tipos de máquina Cambios

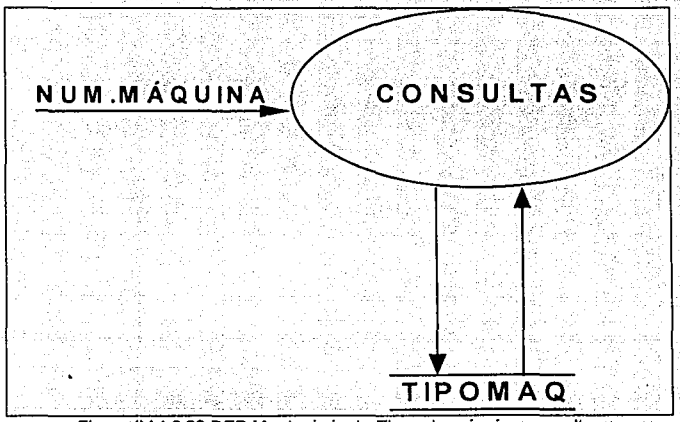


Figura IV.1.2.22 DFD Mantenimiento Tipos de máquina consultas

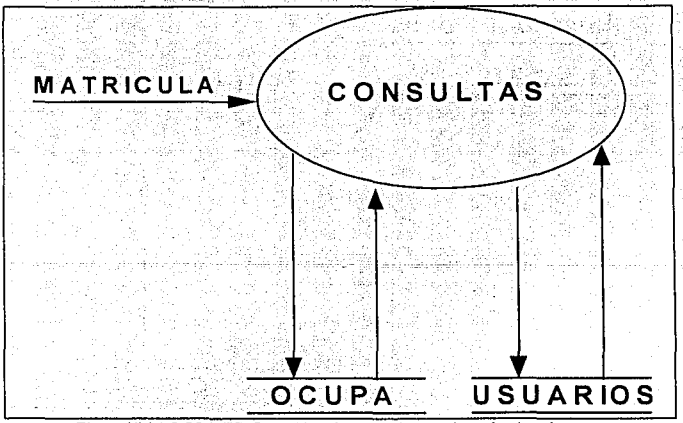


Figura IV.1.2.23 DFD Consulta de usuarios en la sala de cómputo

IV.1.3 DICCIONARIO DE DATOS

A continuación se describirá el diccionario de datos que se utilizó para el sistema de apartado del Tecnológico de Monterrey, Campos Ciudad de México.

Tabla:	Aparta		
Descripción:	Contiene la información de Apartados		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
matr_ap	string	6	Número de matrícula que tiene apartada una hora
cve_hora_ap	string	2	Hora en la que la matrícula tiene apartado
cve_maq_ap	string	3	Clave de la estación que tiene apartada la matrícula

Tabla:	Cancela		
Descripción:	Contiene la información de apartados cancelados		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
matricula_can	string	6	Número de matrícula que cancelo un apartado

Tabla:	Horas		
Descripción:	Contiene el catálogo de horas al día.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
clavehora	string	2	Clave de la hora, por ejemplo: la clave A significa de 7:00 a 7:59.
horas	string	13	Descripción de la clave de hora, por ejemplo: 7:00 a 7:59.

Tabla:	Máquina		
Descripción:	Contiene el catálogo de maquinas.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
claveMaq	string	3	Clave de la estación
claveTipo	string	1	Clave del tipo de estación
serieCPU	string	15	Número de serie del CPU
serieTeclado	string	15	Número de serie del teclado
serieMonitor	string	15	Número de serie del monitor
serieMouse	string	15	Número de serie del mouse
modelo	string	15	Número de modelo de la estación
RAM	string	3	Tamaño de memoria RAM de la estación
HD	string	4	Tamaño del disco duro de la estación
microprocesador	string	6	Tipo de microprocesador de la estación
video	string	5	Tipo de tarjeta de video
FD	string	5	Capacidad del disco flexible
servidor	string	2	Número de servidor al que está conectado esa estación
isla	string	2	Número de isla donde se encuentra físicamente la estación.
EnServicio	string	1	Indica si está en servicio la estación

Tabla:	Ocupa		
Descripción:	Contiene la información de usuarios dentro del Centro Electrónico de Cálculo		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
matr_oc	string	6	Indica el número de matricula que está ocupando una estación.
hora_entrada	string	2	Contiene la hora en que se ocupo la estación.
cve_maq_oc	string	3	Indica el tipo de máquina que se ocupa.
cve_tipo_oc	string	1	Indica la clave del tipo de ocupación, por ejemplo: 0 - Alumnos y 1 - Profesores

Tabla:	Servidor		
Descripción:	Contiene el catálogo de servidores.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
claveSer	string	2	Indica la clave del servidor
nombreSer	string	20	Indica el nombre del servidor
dirIP1	string	3	Dirección de IP 1
dirIP2	string	3	Dirección de IP 2
dirIP3	string	3	Dirección de IP 3
dirIP4	string	3	Dirección de IP 4
dirEth	string	15	Dirección de Ethernet

Tabla:	Tipousu		
Descripción:	Contiene el catálogo de Tipos de Usuario		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
claveTipoUsu	string	6	Clave del tipo de usuario
descripTipoUsu	string	20	Descripción del tipo de usuario

Tabla:	Tipomaq		
Descripción:	Contiene el catálogo de Tipos de Máquina.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
claveTipoMaq	string	1	Clave del tipo de máquina, por ejemplo: 0 Personal Computer, 1 Machintoch, 2 Workstation.
descripTipoMaq	string	20	Descripción del tipo de máquina

Tabla:	Usuario		
Descripción:	Contiene el catálogo de usuarios.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
matricula	string	6	Número de matricula del usuario
paterno	string	20	Apellido paterno del usuario
materno	string	20	Apellido materno del usuario
nombre	string	20	Nombre del usuario
tipo	string	1	Tipo de usuario
carrera	string	6	Número de carrera a la que pertenece

Tabla:	Lunes		
Descripción:	Contiene la información del uso de máquinas por hora al día lunes.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_lun	integer	2	Clave de la hora
cont_hr_lun0	integer	2	Contador de horas de entrada para las PC.
cont_maq_lun0	integer	2	Contador de PC disponibles.
cont_hr_lun1	integer	2	Contador de horas de entrada para las MAC.
cont_maq_lun1	integer	2	Contador de MAC disponibles.
cont_hr_lun2	integer	2	Contador de horas de entrada para WS.
cont_maq_lun2	integer	2	Contador de WS disponibles.

Tabla:	Martes		
Descripción:	Contiene la información del uso de máquinas por hora al día martes.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_mar	integer	2	Clave de la hora
cont_hr_mar0	integer	2	Contador de horas de entrada para las PC.
cont_maq_mar0	integer	2	Contador de PC disponibles.
cont_hr_mar1	integer	2	Contador de horas de entrada para las MAC.
cont_maq_mar1	integer	2	Contador de MAC disponibles.
cont_hr_mar2	integer	2	Contador de horas de entrada para WS.
cont_maq_mar2	integer	2	Contador de WS disponibles.

Tabla:	Miércoles		
Descripción:	Contiene la información del uso de máquinas por hora al día miércoles.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_mie	integer	2	Clave de la hora
cont_hr_mie0	integer	2	Contador de horas de entrada para las PC.
cont_maq_mie0	integer	2	Contador de PC disponibles.
cont_hr_mie1	integer	2	Contador de horas de entrada para las MAC.
cont_maq_mie1	integer	2	Contador de MAC disponibles.
cont_hr_mie2	integer	2	Contador de horas de entrada para WS.
cont_maq_mie2	integer	2	Contador de WS disponibles.

Tabla:	Jueves		
Descripción:	Contiene la información del uso de maquinas por hora al día jueves.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_jue	integer	2	Clave de la hora
cont_hr_jue0	integer	2	Contador de horas de entrada para las PC.
cont_maq_jue0	integer	2	Contador de PC disponibles.
cont_hr_jue1	integer	2	Contador de horas de entrada para las MAC.
cont_maq_jue1	integer	2	Contador de MAC disponibles.
cont_hr_jue2	integer	2	Contador de horas de entrada para WS.
cont_maq_jue2	integer	2	Contador de WS disponibles.

Tabla:	Viernes		
Descripción:	Contiene la información del uso de maquinas por hora al día viernes.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_vie	integer	2	Clave de la hora
cont_hr_vie0	integer	2	Contador de horas de entrada para las PC.
cont_maq_vie0	integer	2	Contador de PC disponibles.
cont_hr_vie1	integer	2	Contador de horas de entrada para las MAC.
cont_maq_vie1	integer	2	Contador de MAC disponibles.
cont_hr_vie2	integer	2	Contador de horas de entrada para WS.
cont_maq_vie2	integer	2	Contador de WS disponibles.

Tabla:	Sabado		
Descripción:	Contiene la información del uso de maquinas por hora al día sábado.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_sab	integer	2	Clave de la hora
cont_hr_sab0	integer	2	Contador de horas de entrada para las PC.
cont_maq_sab0	integer	2	Contador de PC disponibles.
cont_hr_sab1	integer	2	Contador de horas de entrada para las MAC.
cont_maq_sab1	integer	2	Contador de MAC disponibles.
cont_hr_sab2	integer	2	Contador de horas de entrada para WS.
cont_maq_sab2	integer	2	Contador de WS disponibles.

Tabla:	Estlun		
Descripción:	Contiene la información estadística de acceso al C.E.C con y sin apartados por hora al día lunes.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_est_lun	integer	2	Clave de la hora
cont_tot_apa_lun0	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_lun0	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_lun0	integer	2	Contador de entradas sin apartado para PC.
cont_tot_apa_lun1	integer	2	Contador de números de apartados para las MAC.
cont_apa_sinent_lun1	integer	2	Contador de apartados sin entrada para las MAC.
cont_ent_sinapa_lun1	integer	2	Contador de entradas sin apartado para MAC.
cont_tot_apa_lun2	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_lun2	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_lun2	integer	2	Contador de entradas sin apartado para PC.

Tabla: Estmar			
Descripción: **Contiene la información estadistica de acceso al C.E.C con y sin apartados por hora al día martes.**			
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_est_mar	integer	2	Clave de la hora
cont_tot_apa_mar0	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_mar0	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_mar0	integer	2	Contador de entradas sin apartado para PC.
cont_tot_apa_mar1	integer	2	Contador de números de apartados para las MAC.
cont_apa_sinent_mar1	integer	2	Contador de apartados sin entrada para las MAC.
cont_ent_sinapa_mar1	integer	2	Contador de entradas sin apartado para MAC.
cont_tot_apa_mar2	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_mar2	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_mar2	integer	2	Contador de entradas sin apartado para PC.

Tabla:	Estmie		
Descripción:	Contiene la información estadística de acceso al C.E.C. con y sin apartados por hora al día miércoles.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_est_mie	integer	2	Clave de la hora
cont_tot_apa_mie0	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_mie0	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_mie0	integer	2	Contador de entradas sin apartado para PC.
cont_tot_apa_mie1	integer	2	Contador de números de apartados para las MAC.
cont_apa_sinent_mie1	integer	2	Contador de apartados sin entrada para las MAC.
cont_ent_sinapa_mie1	integer	2	Contador de entradas sin apartado para MAC.
cont_tot_apa_mie2	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_mie2	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_mie2	integer	2	Contador de entradas sin apartado para PC.

Tabla:	Estjue		
Descripción:	Contiene la información estadística de acceso al C.E.C. con y sin apartados por hora al día jueves.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_est_jue	integer	2	Clave de la hora
cont_tot_apa_jue0	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_jue0	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_jue0	integer	2	Contador de entradas sin apartado para PC.
cont_tot_apa_jue1	integer	2	Contador de números de apartados para las MAC.
cont_apa_sinent_jue1	integer	2	Contador de apartados sin entrada para las MAC.
cont_ent_sinapa_jue1	integer	2	Contador de entradas sin apartado para MAC.
cont_tot_apa_jue2	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_jue2	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_jue2	integer	2	Contador de entradas sin apartado para PC.

Tabla:	Estvie		
Descripción:	Contiene la información estadística de acceso al C.E.C. con y sin apartados por hora al día viernes.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_est_vie	integer	2	Clave de la hora
cont_tot_apa_vie0	integer	2	Contador de números apartados para las PC.
cont_apa_sinent_vie0	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_vie0	integer	2	Contador de entradas sin apartado para PC.
cont_tot_apa_vie1	integer	2	Contador de números de apartados para las MAC.
cont_apa_sinent_vie1	integer	2	Contador de apartados sin entrada para las MAC.
cont_ent_sinapa_vie1	integer	2	Contador de entradas apartado para MAC.
cont_tot_apa_vie2	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_vie2	integer	2	Contador de apartados entrada para las PC.
cont_ent_sinapa_vie2	integer	2	Contador de entradas apartado para PC.

Tabla:	Estsab		
Descripción:	Contiene la información estadística de acceso al C.E.C. con y.sin apartados por hora al día sábado.		
NEMONICO	TIPO	TAMAÑO	DESCRIPCION
cve_hora_est_sab	integer	2	Clave de la hora
cont_tot_apa_sab0	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_sab0	integer	2	Contador de apartados sin entrada para las PC.
cont_ent_sinapa_sab0	integer	2	Contador de entradas apartado para PC.
cont_tot_apa_sab1	integer	2	Contador de números de apartados para las MAC.
cont_apa_sinent_sab1	integer	2	Contador de apartados entrada para las MAC.
cont_ent_sinapa_sab1	integer	2	Contador de entradas sin apartado para MAC.
cont_tot_apa_sab2	integer	2	Contador de números de apartados para las PC.
cont_apa_sinent_sab2	integer	2	Contador de apartados entrada para las PC.
cont_ent_sinapa_sab2	integer	2	Contador de entradas apartado para PC.

IV.1.4. DIAGRAMA DE ENTIDAD-RELACION

Los primeros diagramas fueron formalizados por C.W. Bachman a finales de los 60´s. La teoría de bases de datos relacionales fué formalizada por el doctor E.F. Codd en 1970. El aprovechamiento de la entidad-relación en el modelado de datos ha provocado un gran número de cambios para soportar extensiones en modelos más complejos, los cuales se desarrollaron como una teoría relacional, pasando del ambiente académico al ambiente de negocios de RDBMS (Manejadores de Bases de Datos Relacionales).

El uso de modelos entidad-relación, tiene sus beneficios, entre los que mencionaremos los siguientes:

- Las discusiones se enfocan en la importancia de las relaciones.
- Emplea una sintaxis en los diagramas que permite reunir, una gran cantidad de información de una manera bastante comprensible.
- Captura los requerimientos de datos del mundo real, los cuales son comprensibles al usuario final y al diseñador de la base de datos.
- El modelo entidad-relación es una excelente fuente de documentación para los administradores de la base de datos y a los desarrolladores de aplicaciones.

De lo anterior se desprende que, los objetos de información importantes y sus relaciones pueden ser organizados y fácilmente comprendidos. Este modelo lógico puede ser usado para "mapear" y construir la base de datos física.

Existen muchos factores importantes que contribuyen a un diseño satisfactorio de un modelo de datos. El aprender y aplicar las diferentes técnicas de modelado de datos de manera separada no garantizará un adecuado modelo. Un diseñador de la base de datos deberá tener un amplio conocimiento y la profundidad respectiva en el proyecto del negocio y sus requerimientos. Una comunicación interactiva y productiva deberá ocurrir entre el usuario final y el diseñador de la base de datos a través del proceso de diseño. El uso de una metodología estructurada es crítica durante el ciclo de vida del desarrollo.

Hay muchas fases y etapas asociadas con la metodología de desarrollo. Esas fases pueden ser descritas como:

- Factibilidad
- Requerimientos funcionales
- Diseño técnico
- Codificación y pruebas
- Implementación
- Mantenimiento

El desarrollo de diagramas de entidad-relación es solamente un paso en el ciclo de vida del proyecto.

Como bosquejo para el diseño de una base de datos y a partir de lo mencionado con anterioridad, tenemos una serie de consideraciones las cuales mostraremos en los siguientes puntos:

- Identificar los principales objetos de datos.
- Definir los principales objetos de datos.
- Obtener un diagrama de los objetos de datos, utilizando las capacidades de entidad-relación.
- Resolver el modelo lógico de datos.
- Normalizar el modelo lógico de datos.
- Convertir el modelo de datos lógicos en un esquema físico (Base de datos).

Se puede apreciar el uso de un concepto denominado, objetos de datos, los cuales se pueden clasificar en tres clases representados en los modelos de datos de entidad-relación:

- Entidades
- Relaciones
- Atributos

Definiremos lo que es **entidad**: Una entidad en el principal objeto de datos que es de un interés significativo para el usuario. Esto puede ser una persona, lugar cosa o evento de un interés informativo. Una entidad, puede representar algo real, tangible o abstracto.

Un buen momento para descubrir entidades es durante la entrevista con el usuario. Se le pregunta al usuario que dé una descripción de sus actividades en el negocio. Se registra cualquier elemento de información al que el usuario haga referencia durante la entrevista. Considerando los objetos de mayor interés (gente, lugares, cosas o eventos). La lista inicial de entidades puede cambiar a través de la entrevista.

Considerando lo anterior y aplicándolo al problema actual tenemos que el diagrama de entidades estaría representado por la figura IV.1.4.1. El cual muestra cosas, personas, lugares y eventos.

El segundo paso en el modelado de datos relacional es, identificar las relaciones entre las entidades.

Ahora definiremos el concepto de **relación**: Una relación representa asociaciones del mundo real entre una o más entidades. La representación más común es la conectividad entre entidades. Identificar y definir las relaciones de entidades, es una

de las partes más importantes en el proceso de diseño de una base de datos relacional.

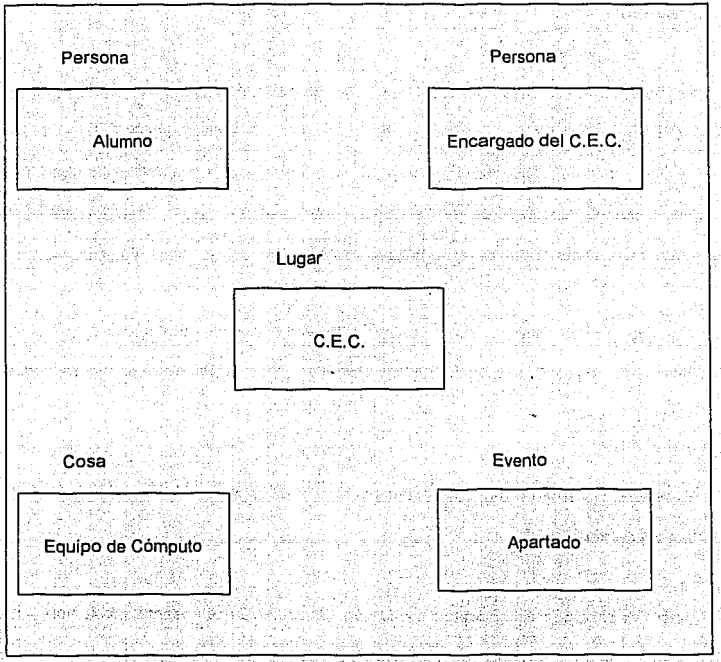


Figura IV.1.4.1 Representación de las entidades involucradas para el sistema de apartado.

El diagrama de relaciones se muestra en la figura IV.1.4.2. Y como se puede apreciar, existe una descripción más detallada sobre la forma en que interactúa cada una de las entidades con la otra.

Las relaciones listadas en la figura, fueron identificadas en el sistema de apartado de equipo de cómputo. El sujeto está localizado a la izquierda del diagrama, mientras que el objeto se encuentra a la derecha. Un solo verbo o preposición es colocado entre las entidades identificando la relación. El verbo o preposición localizado a la izquierda describe la relación sujeto-objeto. Mientras que el verbo o preposición en el lado derecho describe la relación inversa objeto-sujeto.

Las relaciones entre entidades son descritas en términos de **conectividad, cardinalidad,** y **existencia.**

La representación más común es la conectividad entre entidades, La conectividad, cardinalidad y existencia, ayudan a definir las reglas de negocio para una empresa de negocio. La semejanza entre el modelado de datos con el diagrama entidad-relación proporciona una sintaxis de diagrama que representa esta conectividad, cardinalidad, y existencia.

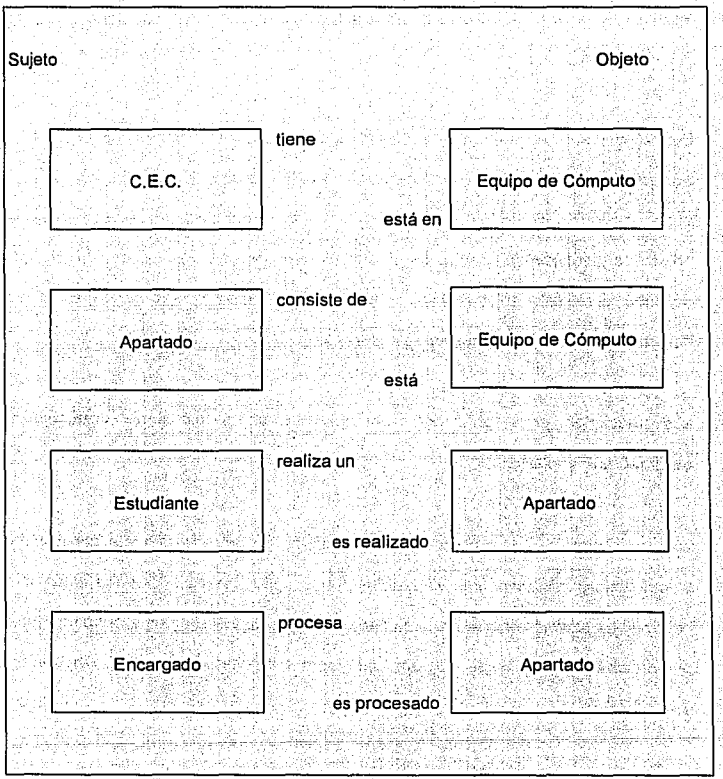


Figura IV.1.4.2. Diagrama de relación entre entidades para el sistema de apartado.

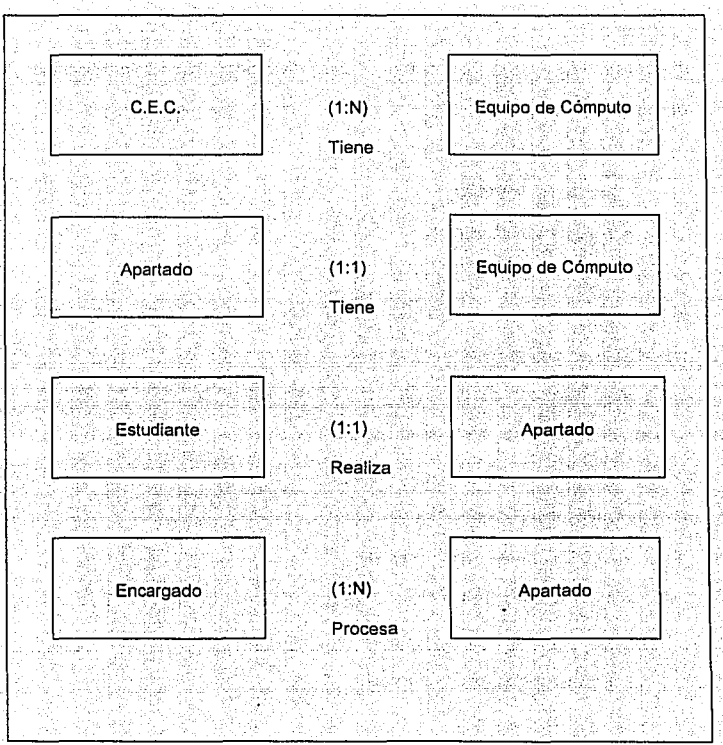


Figura IV.1.4.3. Representación de la conectividad en el sistema de apartado.

La conectividad, describe el número de ocurrencias de una entidad. Además de que describe las reglas de negocio de una empresa. La conectividad esta representada por tres tipos:

- Uno a uno (1:1)
- Uno a muchos (1:N)
- Muchos a muchos (M:N)

El tipo más común de las conectividades es uno a muchos. Se recomienda tratar de representar el mayor número de relaciones como relaciones uno a muchos. Para comprender este concepto tomaremos como base la figura IV.1.4.3. La cual muestra la relación de conectividad entre las entidades.

La figura anterior la podemos describir como:

- El C.E.C. tiene muchos equipos
- Un apartado consiste de un equipo
- Un estudiante aparta un equipo
- El encargado procesa muchos apartados

Sin embargo, esta conectividad solamente nos menciona que existe una relación entre las entidades, sin mencionar un número más cercano a la realidad. Para ello, la cardinalidad define esta restricción en el número de ocurrencias, de una entidad, que se encuentran relacionadas.

Es importante identificar cualquier restricción de cardinalidad en el diseño de la base de datos; las restricciones también son consideradas y necesarias en el diseño de la aplicación.

Como ejemplo de lo anterior, consideremos la figura IV.1.4.4. en la cual se define una cardinalidad de 1:1 en la relación entre estudiante y el evento de apartado; lo que significa que un estudiante solamente podrá realizar el apartado de un equipo de cómputo.

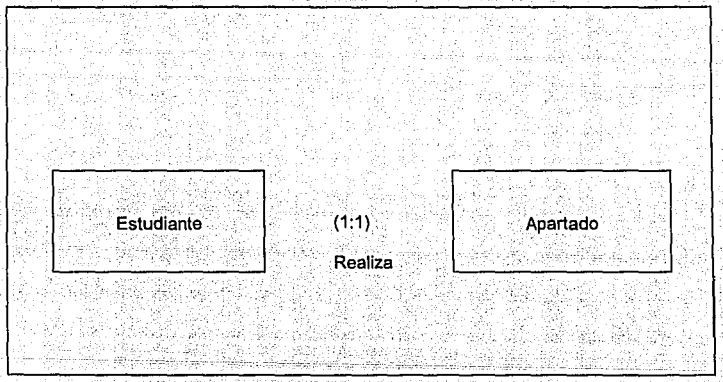


Figura IV.1.4.4. Ejemplo de asignación de la cardinalidad en la relación entre las entidades de estudiante y apartado.

Sin embargo, en este caso particular, la cardinalidad se da desde el momento mismo de efectuar la conectividad ya que la conectividad es 1:1. Pero en lo que se refiere a las otras relaciones en las cuales la conectividad es de 1:N ; se deberá considerar la posibilidad de tener un número ilimitado de ocurrencias, pues no es posible determinar un número exacto o restringido de apartados de equipo por día, además de que sería inoperante.

Por otra parte, tenemos la dependencia de existencia, lo cual se refiere a la necesidad de existencia de las instancias en las relaciones de las entidades; de donde la dependencia de existencia se presenta de alguna de las dos siguientes formas:

- Obligatoria: donde una instancia de la entidad deberá siempre existir en la relación.
- Opcional: si una instancia de la entidad no es necesario que exista en la relación.

Hay muchas circunstancias donde la existencia de ambas entidades es necesaria para la existencia misma de la relación. El modelo de datos entidad relación proporciona la sintaxis de diagrama para indicar la dependencia de existencia. Esta es una consideración muy importante cuando diseñamos la base de datos física y en el desarrollo de la aplicaciones.

Aplicando lo anterior en nuestra definición de las relaciones de entidades del sistema de apartado tenemos la figura IV.1.4.5. Donde se muestra cuales son las dependencias de existencia obligatorias y opcionales.

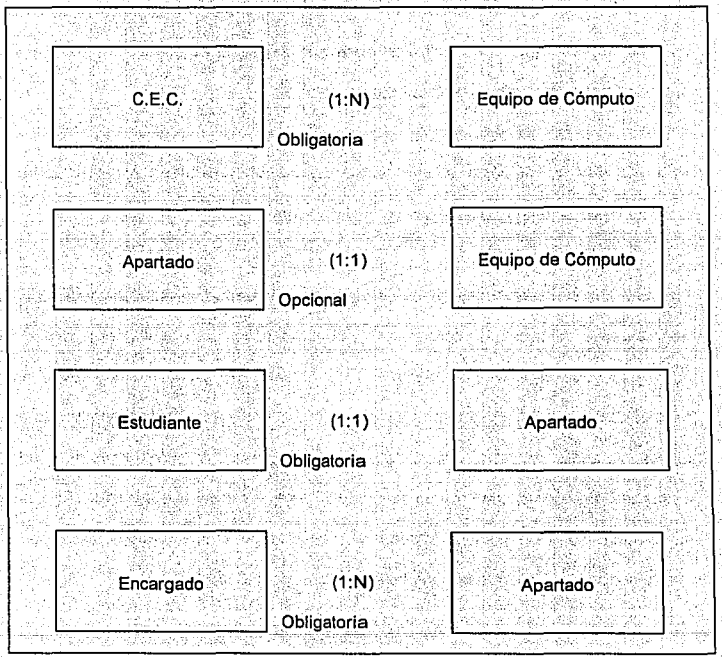


Figura IV.1.4.5. Dependencia de existencia para el modelo que representa el sistema de apartado.

Para determinar como se dan estas dependencias de existencia; consideremos primeramente la relación entre las entidades de C.E.C. y Equipo de cómputo. Si pensamos en que no exista el C.E.C. podemos decir que el Equipo de cómputo no existe, esto es, si no existe el lugar no existen las máquinas para préstamo. Por su parte, el hecho de que la relación de las entidades Apartado y Equipo de cómputo sea opcional se debe a que en el caso de no existir la entidad Apartado no implica que la entidad Equipo de cómputo desaparezca o no se dé.

Una vez que hemos visto algunos aspectos de las **entidades** y las **relaciones**, consideremos el último punto que es el de **atributos**.

Los atributos son características que proporcionan información detallada y descriptiva acerca de las entidades. Un atributo no se puede descomponer nuevamente sin perder su significado original; por ejemplo, una cuenta de contabilidad puede estar representada por el siguiente número: 10-1613-6100. Si el número de cuenta representa actualmente un número de centro de costos, un número de departamento y un número de cuenta, este puede ser dividido en tres atributos separados. Ahora, un número de orden de compra puede ser representado como 109540, pero este no puede ser dividido nuevamente sin que pierda su significado original.

Existen dos clasificaciones generales de los atributos: el **identificador** y el **descriptor**. Un identificador también es conocido como **llave**.

- Un identificador es un atributo que especifica una característica única de una instancia en particular de la entidad. Por ejemplo un atributo identificador de una entidad empleados puede ser el número de seguro social. Un número de seguro social siempre será un identificador único de un empleado.
- Un descriptor es un atributo que especifica una característica no única de una instancia en particular de la entidad. Un atributo descriptor de la entidad empleado puede ser el apellido. Un apellido no siempre identifica a un único empleado si dos empleados tienen el mismo apellido.

Los atributos identificados para las diferentes entidades son representados en la figura IV.1.4.6. De los cuales, con una **k** identificaremos los que representan un atributo de tipo llave.

Para la entidad usuario es necesario contar con la matrícula del usuario, sus apellidos tanto paterno como materno, su nombre, que tipo de usuario es, alumno, académico, etc., si está habilitado, cual es su estatus, las sanciones que tiene acumuladas y la carrera a la que pertenece.

En el caso de la entidad equipo, tenemos los siguientes atributos: clave de la máquina, tipo de máquina, números de serie de cpu, teclado, monitor , mouse, el modelo, la

cantidad de memoria RAM, el disco duro, el microprocesador, el video, la unidad de disco, el servidor que representa en la red, la "isla" en la que se encuentra el equipo y si esta en servicio o no.

La entidad aparta contendrá los atributos de número de matrícula de apartado, la clave de la hora de apartado, la clave de la máquina apartada y la clave de tipo de apartado.

La entidad cancela solamente contendrá el número de matrícula del usuario que cancela para poder desasignar su apartado y dejarlo disponible para otro usuario, así como para tener el total de cancelaciones que se den en cada hora.

La entidad ocupa contará con los atributos de número de matrícula, la hora de entrada, la clave de la máquina que está ocupando, y el tipo de la máquina que esté ocupando. Es importante entender que esta entidad tiene como objetivo determinar cuantos equipos fueron ocupados en cada hora, durante el día y esta información se utilizará en la entidad de cada día de la semana.

Para la entidad de cada día de la semana, se contemplan los atributos de clave de hora, contador de horas del día para el equipo Pc, un contador de los equipos Pc utilizados en dicha hora, así como un contador de horas del día para el equipo Macintosh y su contador de equipos utilizados, finalmente tenemos un contador de las horas utilizadas en equipo workstation y cuantos equipos workstation fueron utilizados.

La entidad de estadísticas por día estará conformada por los atributos siguientes: clave de la hora, y un contador del número total de apartados, un contador de apartados que no entraron, un contador de entradas sin un apartado previo, donde para cada tipo de equipo, Pc, Macintosh y workstation existirán sus respectivos grupos de contadores, como se aprecia en la figura IV.1.4.6.

Existe un grupo de tablas conocidas como Dominios, las cuales describen los valores válidos que puede asumir un atributo. Por lo regular el conjunto de valores válidos para un atributo puede ser representado por el uso de códigos. Esos dominios en muchos casos , resultan en la creación de una nueva entidad. El uso de códigos puede ayudar a reducir los requerimientos de almacenamiento y salvar espacio en disco.

Como un ejemplo de dominio tenemos lo que son los códigos de estados. Donde estos son representados por una entidad llamada estados; con el código del estado y la descripción como sus atributos. En nuestro caso tenemos un conjunto de dominios, representados con las entidades mostradas en la figura IV.1.4.7. Estos dominios corresponden a los tipos de servidores, tipos de máquinas, tipos de usuarios y las claves de horas.

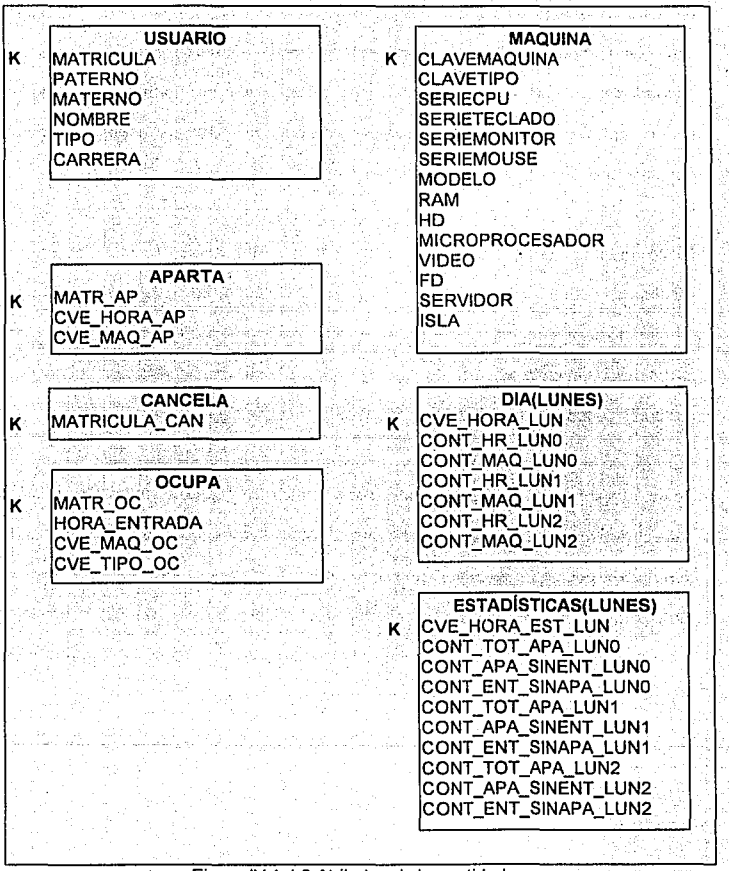


Figura IV.1.4.6 Atributos de las entidades.

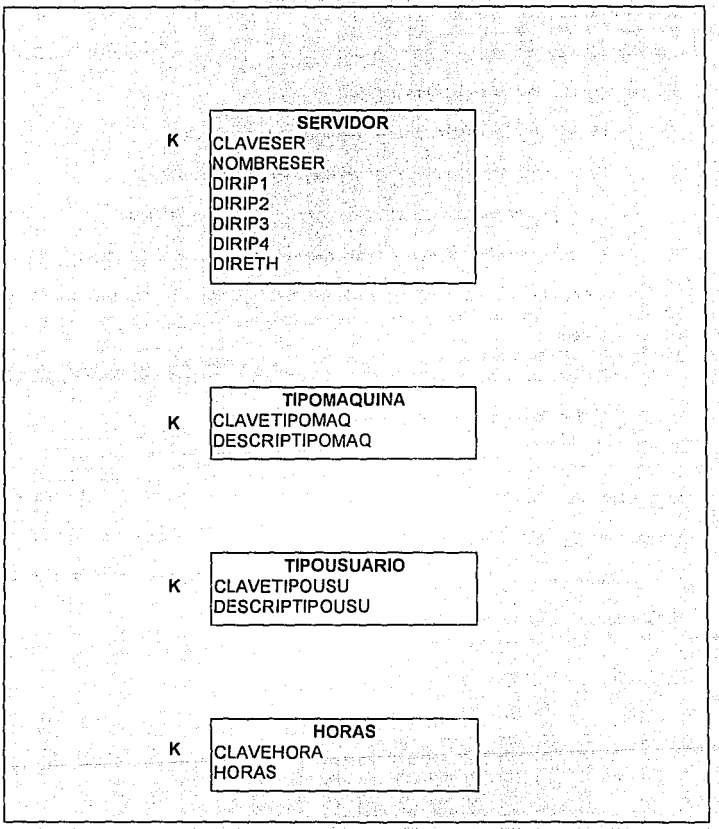


Figura IV.1.4.7 Dominios del sistema, representados en sus respectivas entidades.

Los diagramas de entidad relación juegan un papel muy importante en las metodologías de diseño de bases de datos relacionales. Sus principales funciones quedan definidas en los siguientes puntos:

- Modela la información necesaria para una organización.
- Es utilizado para identificar las entidades y sus relaciones.
- Es empleado como el punto de inicio en la definición de datos.
- Es una excelente fuente de documentación para los desarrolladores de la aplicación, los administradores de la base de datos y los administradores del sistema.
- Es usado para crear el diseño físico de una base de datos, que será traducido en un esquema de la base de datos.

Los diagramas entidad relación serán representados utilizando el esquema formalizado por C.W. Bachman a finales de los sesentas. Los diagramas proporcionan una sintaxis diagramática que es fácil de entender. Los objetos utilizados son mostrados en la figura IV.1.4.8.

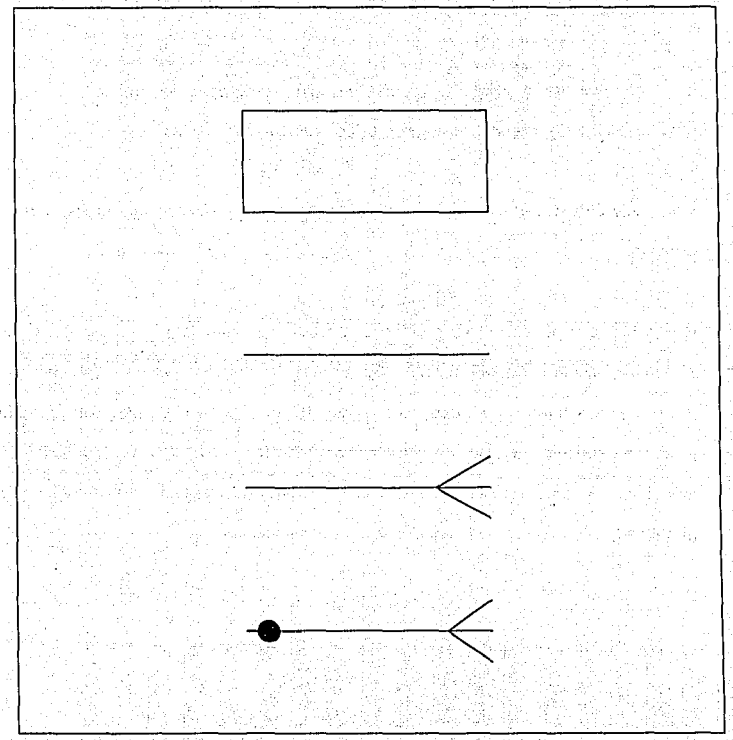


Figura IV.1.4.8 Objetos básicos empleados en la definición del diagrama entidad relación.

Una entidad está representada por un cuadrado. El nombre de la entidad es colocado dentro del cuadro en singular y en minúsculas.

Una relación es representada por una línea entre las entidades.

Una línea con uno o ambos extremos terminados con más líneas implica que una o más instancias de una entidad están asociadas con la otra entidad.

Un círculo atravesando la línea de relación indica que la existencia de la relación es opcional.

Como ejemplo podemos representar una parte del sistema por medio de estos diagramas, como se muestra en la figura IV.1.4.9. Donde tenemos la entidad del usuario y la entidad de apartado representados por medio de un diagrama entidad relación. Este diagrama nos muestra como varios usuarios pueden tener solo una máquina apartada y además dicha relación no es opcional ya que si consideramos que la entidad usuario no se dá la entidad apartado tampoco existirá.

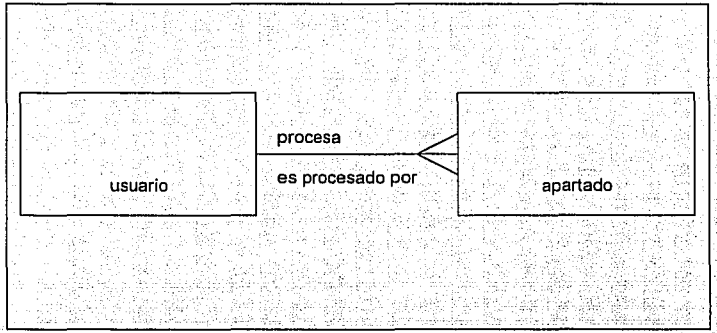


Figura IV.1.4.9 Ejemplo de la utilización de los elementos para un diagrama entidad relación

Con lo anterior podemos generar nuestro diagrama entidad relación de una manera sencilla y comprensible. Este diagrama se muestra en la figura IV.1.4.10.

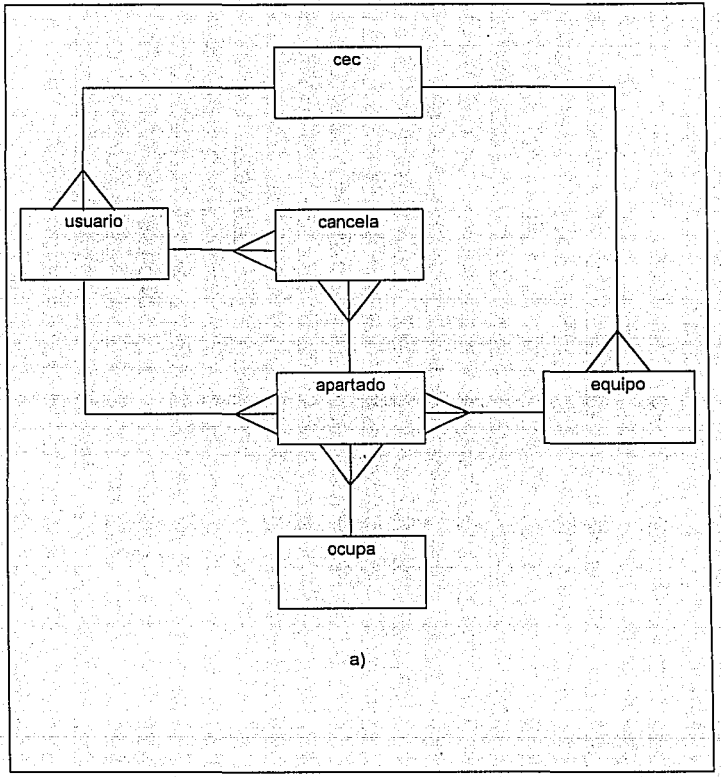


Figura IV.1.4.10 a) Diagrama entidad relación del sistema de apartado del C.E.C.

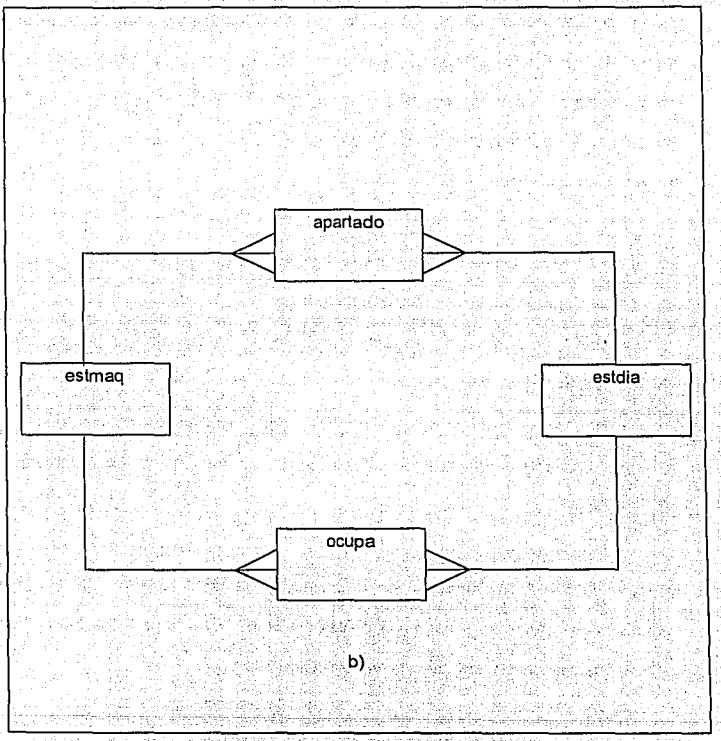


Figura IV.1.4.10 b) Diagrama entidad relación para la obtención de la información estadística.

Todas las entidades deberán tener una llave primaria. La llave primaria deberá ser un atributo o combinación de atributos que identifica únicamente una instancia de la entidad. Los atributos de tipo identificador, son los mejores candidatos para ser las llaves primarias ya que los atributos de tipo identificador cumplen con los requerimientos de ser únicos. Una llave primaria deberá existir para todas las instancias de cada entidad.

Esta llave primaria nos permitirá definir la estructura de las diferentes tablas. La llave primaria se puede resumir en los siguientes puntos:

- Una llave primaria es un atributo o combinación de atributos que identifican una instancia de la entidad.
- Una llave primaria deberá existir para todas las instancias de la entidad.
- Cada tabla deberá tener una llave primaria y solamente una llave primaria por tabla.

La figura IV.1.4.11 nos muestra los diagramas de entidad relación con las llaves primarias que conforman a cada una de las entidades. Esto, a partir de los atributos definidos para cada entidad como se vio en la figura IV.1.4.6.

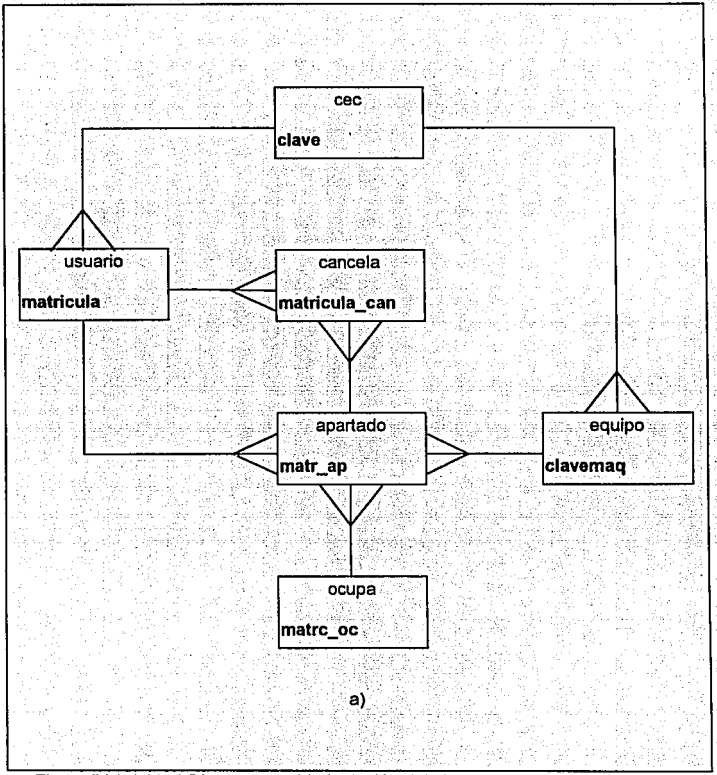


Figura IV.1.4.11 a) Diagrama entidad relación del sistema de apartado del C.E.C. mostrando las llaves primarias.

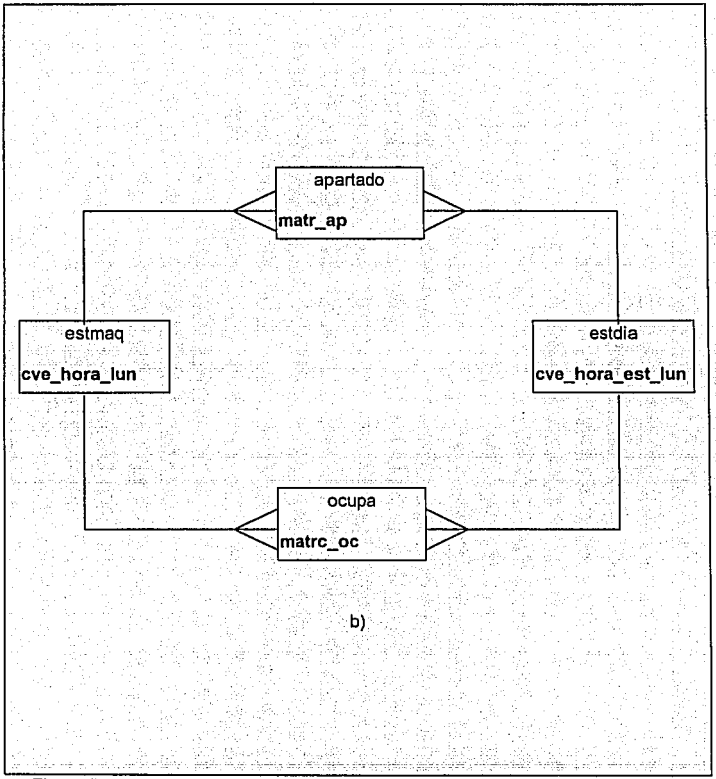


Figura IV.1.4.11 b) Diagrama entidad relación para la obtención de la información estadística, mostrando las llaves primarias.

Existen unas reglas para las llaves primarias, las cuales nos permitirán definir de una manera más sencilla la base de datos:

- No permiten nulos.
- Deben ser únicas.
- No influyen en el orden de los renglones o columnas.
- No influyen en el acceso de los renglones.

Por otra parte, existen también las llamadas llaves secundarias y las cuales son el medio de relación con las otras entidades. Esta llave secundaria deberá hacer referencia a una llave primaria existente en la entidad asociada. Podemos resumir sus características en los siguientes puntos:

- Es un atributo que deberá hacer referencia a una llave primaria en la otra entidad.
- Es un atributo o combinación de atributos que son usados para establecer la relación entre las entidades.
- Pueden ser nulos.
- Pueden contener duplicados.
- Pueden ser modificados.

Las llaves secundarias en el diagrama entidad relación del sistema de apartado se pueden apreciar en la figura IV.1.4.12

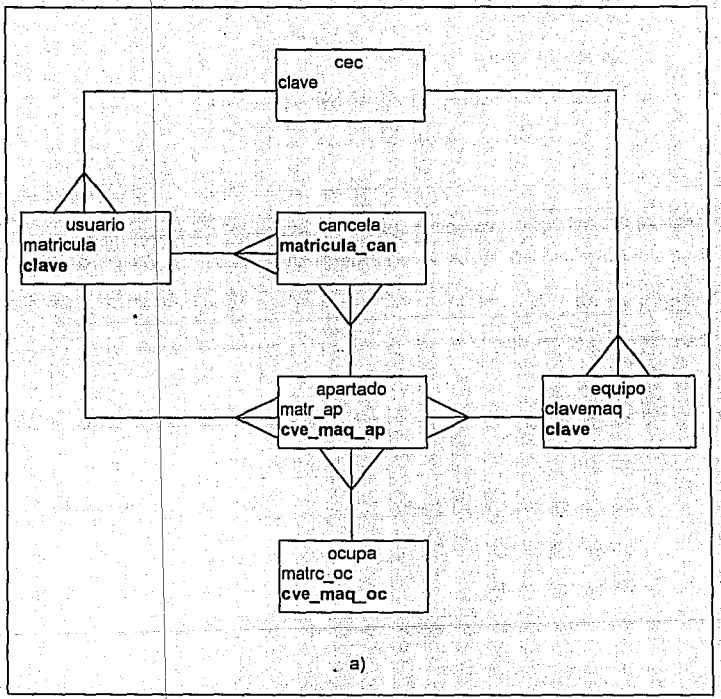


Figura IV.1.4.12 a) Diagrama entidad relación del sistema de apartado del C.E.C. mostrando las llaves secundarias.

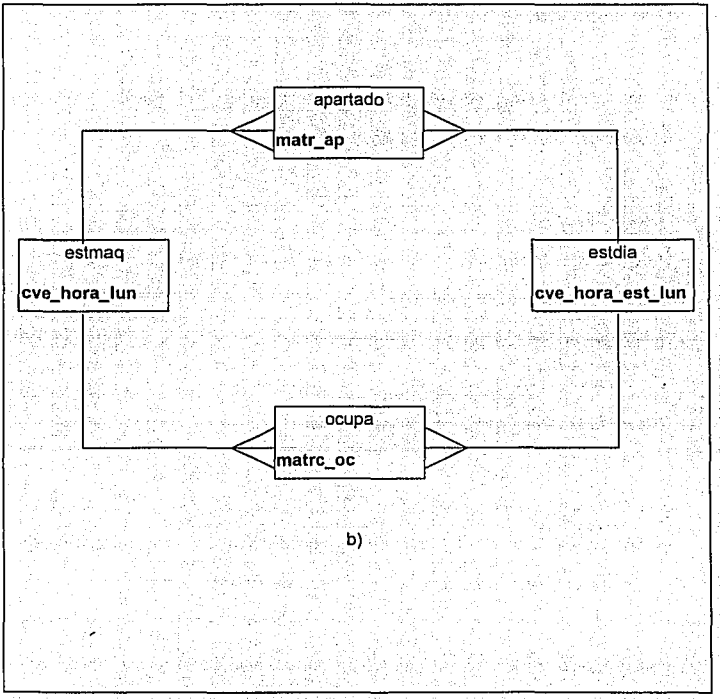


Figura IV.1.4.12 b) Diagrama entidad relación para la obtención de la información estadística.

Como se puede apreciar en la figura IV.1.4.12 b), no existen llaves secundarias en estas relaciones ya que solamente se utiliza la información de las entidades de apartado y se ocupa para actualizar únicamente las entidades de estmaq y estdia.

NORMALIZACION

Una vez definido el diagrama entidad relación aplicaremos las reglas de normalización y así conseguir los siguientes beneficios:

- Gran flexibilidad.
- Asegura que los atributos estén colocados en las tablas apropiadas.
- Reduce la redundancia de datos.
- Incrementa la efectividad de los programadores.
- Disminuye el costo de mantenimiento de la aplicación.
- Maximiza la estabilidad del modelo de datos.

Un modelo de datos normalizado es más flexible y permite soportar un amplio rango de necesidades del usuario final con un mínimo de cambios a la estructura de datos; desacuerdo a los requerimientos de las nuevas necesidades.

La normalización reduce la redundancia de datos; haciendo ésto fácil y manteniendo la consistencia de los mismos; y minimizando la cantidad de espacio requerido para almacenar los datos.

El diseño sencillo y lógico resulta en un incremento en la productividad. La normalización reduce el costo de mantenimiento para una aplicación, por que los cambios son hechos fácilmente.

Para aplicar de una manera eficiente la normalización se sugiere seguir las siguientes reglas:

- Cada forma normal forza los datos a ser más organizados que la anterior.
- Cada forma normal deberá ser llevada acabo antes de que la siguiente forma pueda ser aplicada.

PRIMER FORMA NORMAL

Una entidad está en su primera forma normal si no existen grupos repetidos (atributos que tienen el mismo dominio). Cada entidad deberá tener un número fijo de atributos con valores individuales. Veamos la figura IV.1.4.13 donde tenemos las tablas del sistema de apartado y que serán las tablas a normalizar.

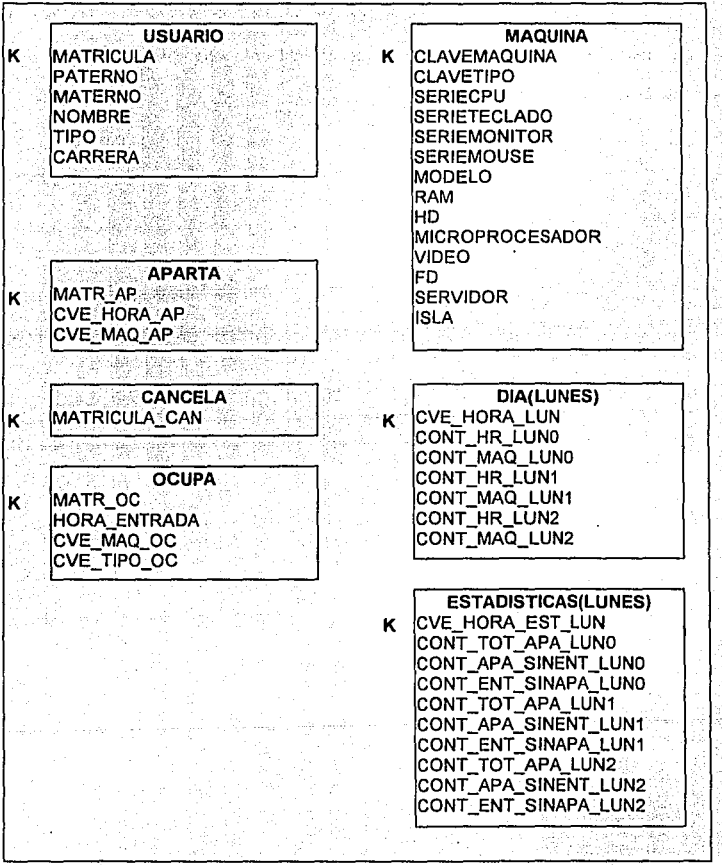


Figura IV.1.4.13 Atributos de las entidades del sistema de apartado.

De la figura podemos observar que existen dos tablas que presentan la repetición de dominios en su definición. Estas tablas son: las estadísticas por día del equipo (lunes, martes, ..., sábado) y las tablas de estadísticas del uso (estlun, estmar, ..., estsab).

Aplicando la primer forma normal; como en el ejemplo, únicamente a las tablas lunes y estlun; tenemos un cambio tanto en el diagrama entidad relación como en la estructura de las tablas como se puede apreciar en la figura IV.1.4.14.

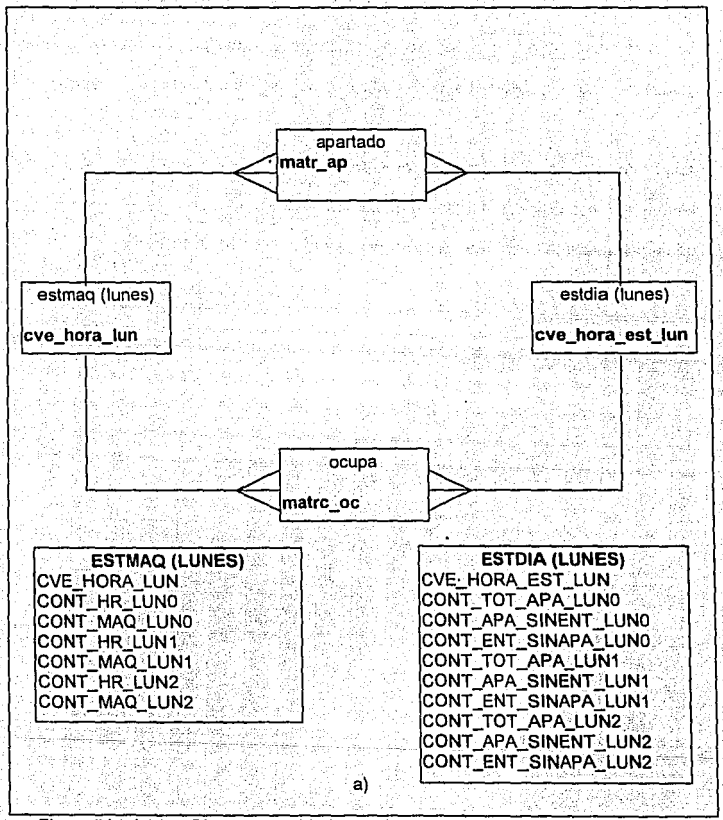


Figura IV.1.4.14 a) Diagrama entidad relación para la obtención de la información estadística y las tablas con dominios repetidos.

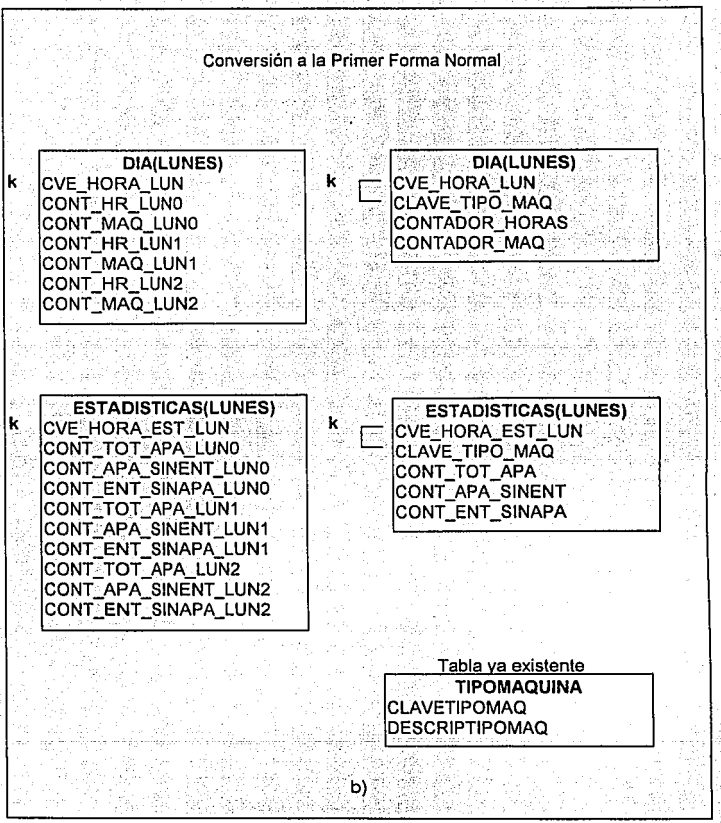


Figura IV.1.4.14 b) Conversión de las tablas estadísticas a la primer forma normal.

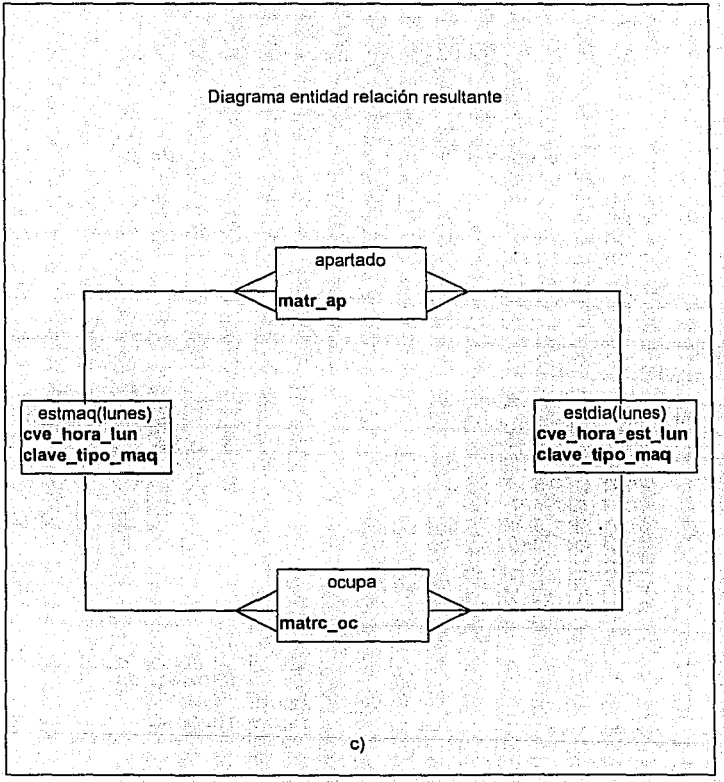


Figura IV.1.4.14 c) Diagrama entidad relación resultante.

Como podemos observar de la figura, no existe un cambio sustancial en el diagrama entidad relación ya que se mantiene igual, sin embargo, en lo relacionado a la estructura de las tablas, en este caso si hay cambios y consisten en agregar en ambas tablas un atributo de **clave_tipo_maq** el cual permitirá determinar cual es el equipo del que se tiene la estadística, conservando los atributos contadores para cada una de las respectivas tablas.

Ahora, si observamos con detalle la definición de las tablas, veremos que estas tablas normalizadas constan de los mismos campos para la llave primaria, por lo que estas dos tablas pueden quedar resumidas en una sola. Lo que significa un cambio más en la estructura de las tablas y un cambio en el diagrama entidad relación para la obtención de las estadísticas, como lo podemos apreciar en la figura IV.1.4.15

De lo anterior, se puede ver como esta nueva estructura tiene mayor flexibilidad en el manejo de la información, pues se deja de limitar el control de solo tres tipos de equipos por un número mayor y solo se agregó en la tabla de dominios **tipomaq**. Así como, el nuevo equipo y al momento de llevar las estadísticas por equipo, fácilmente este nuevo equipo es incluido y considerado. Además, en el caso excepcional en el que se llegara a apartar equipo de un solo tipo en un día, no se utilizaría un mayor espacio del disco duro, pues solo se almacenará la información correspondiente y necesaria.

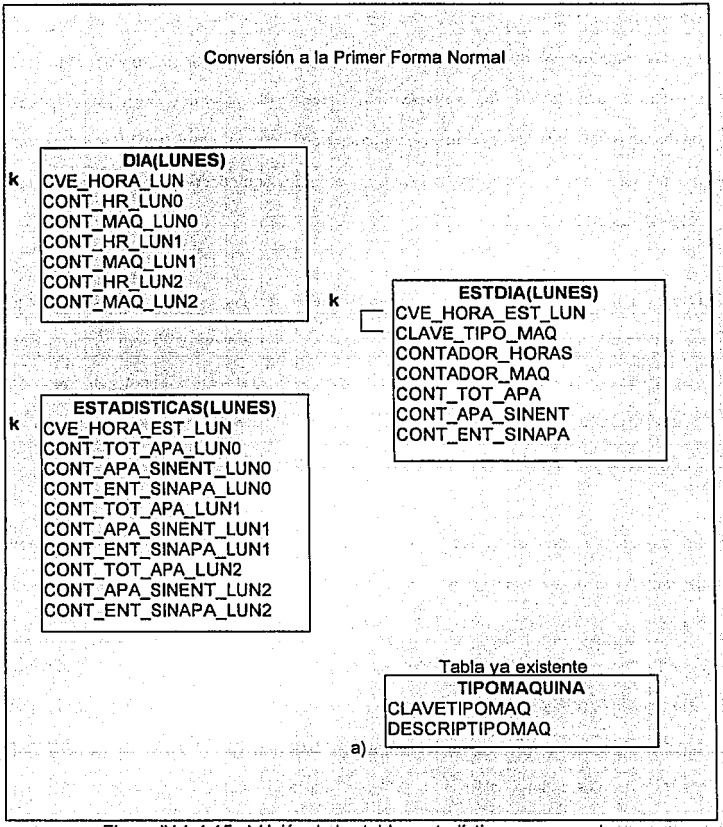


Figura IV.1.4.15 a) Unión de las tablas estadísticas en una sola.

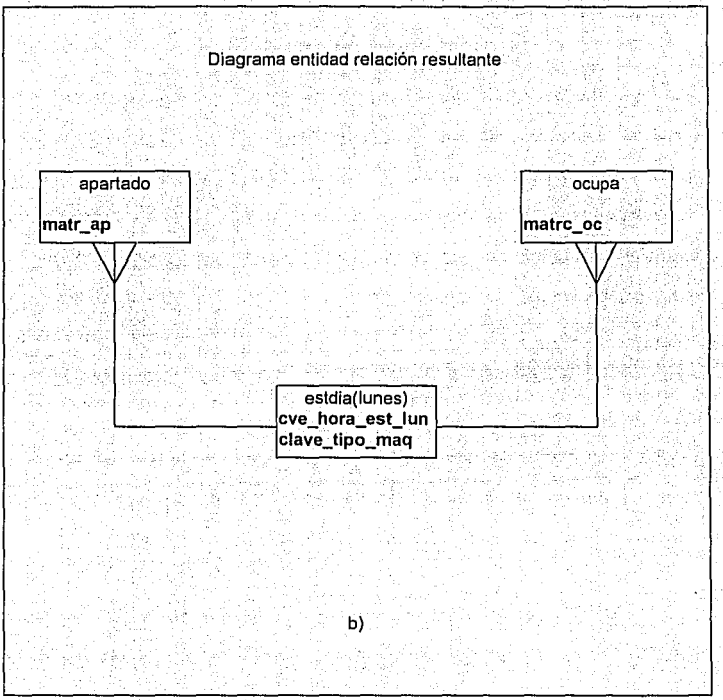


Figura IV.1.4.15 b) Diagrama entidad relación resultante al dejar una sola tabla para las estadísticas.

SEGUNDA FORMA NORMAL

Una relación se encuentra en la segunda forma normal si ésta se encuentra en la primer forma normal y todos sus atributos dependen de la llave primaria. Recuerdese, que una llave primaria es un conjunto mínimo de atributos requeridos para determinar de manera única una instancia de una entidad. Un atributo no-llave o descriptivo, es cualquier atributo que no es parte de la llave primaria. La segunda forma normal requiere que todo atributo no-llave necesite de la llave primaria por completo para su identificación única. Los atributos no-llave deben ser completamente dependientes funcionales de la llave primaria.

La dependencia funcional indica que hay una liga entre los valores de dos diferentes columnas de una misma tabla. Veamos nuevamente en la figura IV.1.4.16 la descripción de las tablas del sistema de apartado, ahora con la nueva tabla.

De la figura, podemos observar que los atributos que las conforman y sus llaves principales; identifican la existencia de la dependencia funcional entre los atributos y la llave principal.

Para verificar lo anterior analizaremos la taba de estadísticas para determinar su dependencia funcional. Tenemos en la figura IV.1.4.17 su representación.

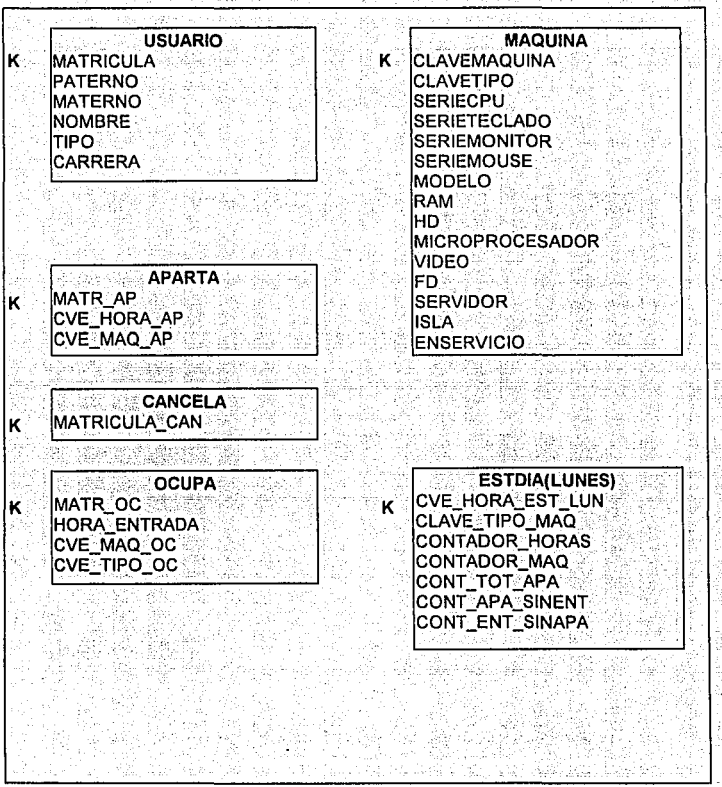


Figura IV.1.4.16 Atributos de las entidades.

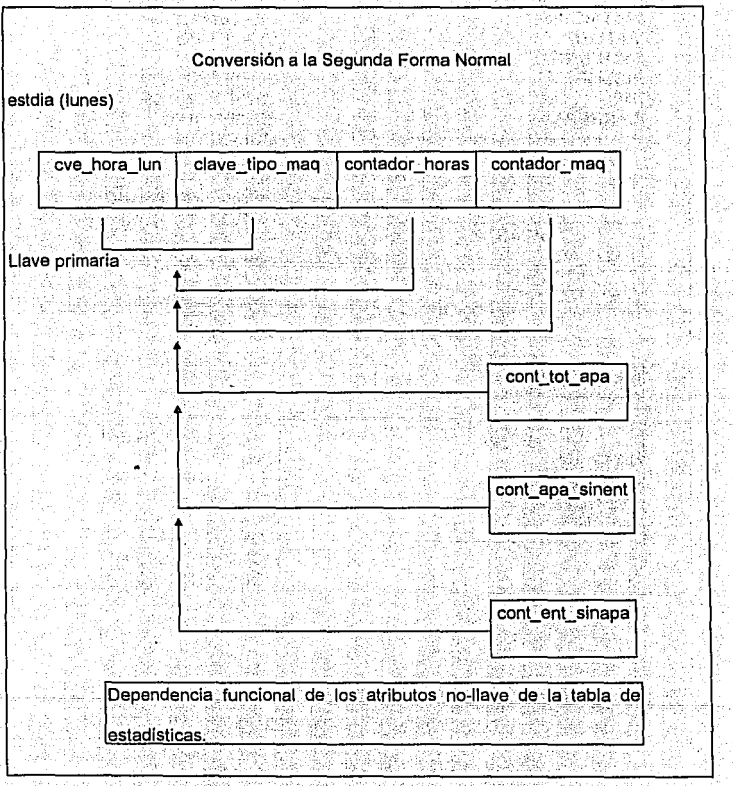


Figura IV.1.4.17 Revisión de las tablas estadísticas en la conversión a la segunda forma normal.

De la figura IV.1.4.17, podemos ver que la tabla de estadisticas cumple con la segunda forma normal, por lo que no es necesario llevar a cabo cambios o modificaciones en esta tabla, y en general las demás tablas que constituyen el sistema de apartado, cumplen con la segunda forma normal, si vemos la figura IV.1.4.18.

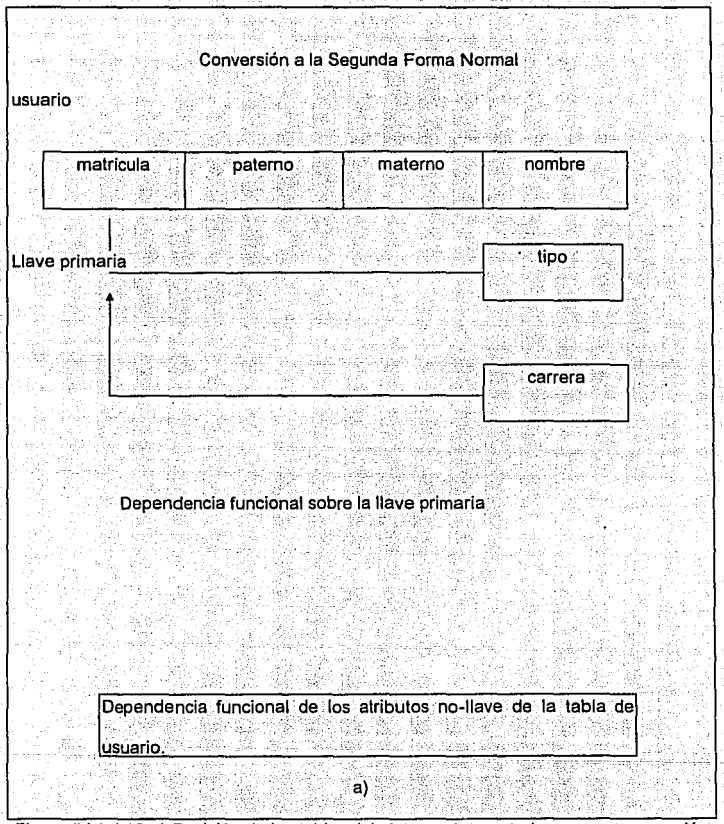


Figura IV.1.4.18 a) Revisión de las tablas del sistema de apartado para su conversión a la segunda forma normal.

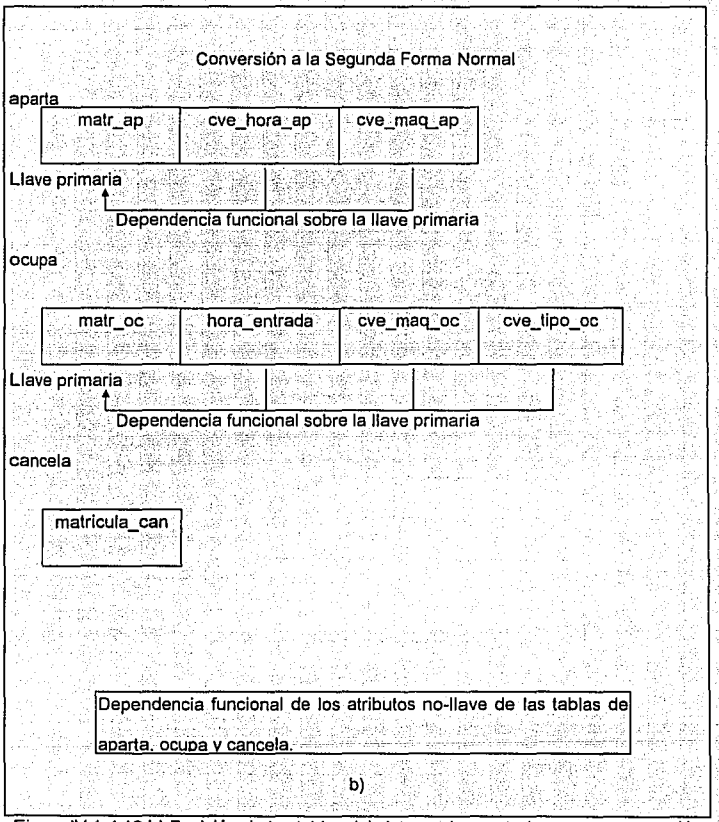


Figura IV.1.4.18 b) Revisión de las tablas del sistema de apartado para su conversión a la segunda forma normal.

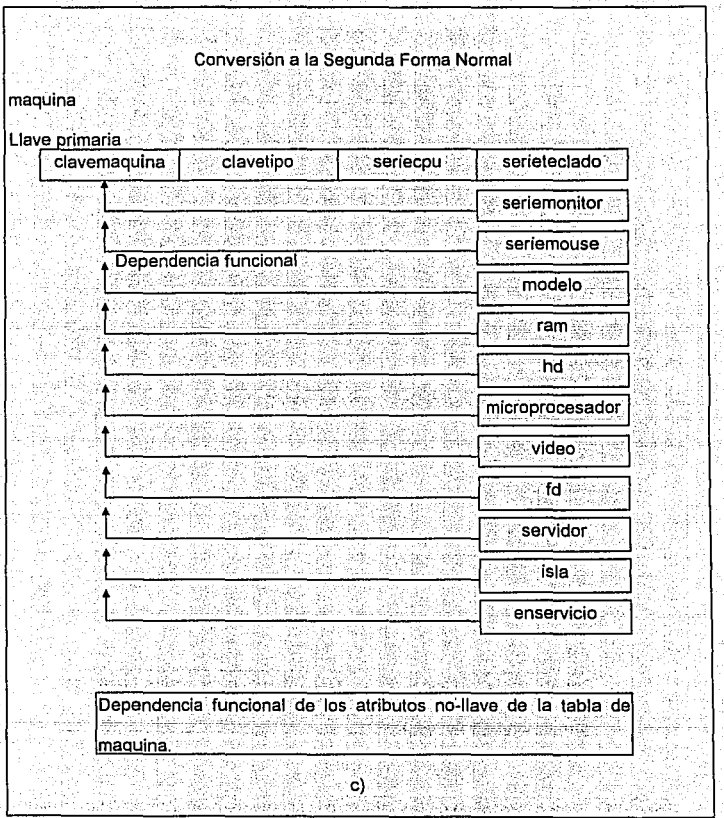


Figura IV.1.4.18 c) Revisión de las tablas del sistema de apartado para su conversión a la segunda forma normal.

De la figura IV.1.4.18 apreciamos que la dependencia funcional existe, lo cual no requiere de una modificación en las estructuras de las tablas ni del diagrama entidad relación.

TERCER FORMA NORMAL

Una relación está en la tercer forma normal si todos sus atributos no son dependientes transitivos de la llave primaria. La tercer forma normal requiere que todo atributo no llave se encuentre en la segunda forma normal y que dependan exclusivamente de la llave primaria. Esto eliminará la dependencia transitiva, en la cual los atributos no llave no solo dependen de la llave primaria sino que a su vez dependen de otro atributo no llave, donde éstos también dependen de la llave primaria.

Veamos ahora al aplicar la tercer forma normal a nuestras tablas del sistema de apartado. Como ejemplo aplicaremos la tercer forma normal a la tabla de usuario la cual se muestra en la figura IV.1.4.19.

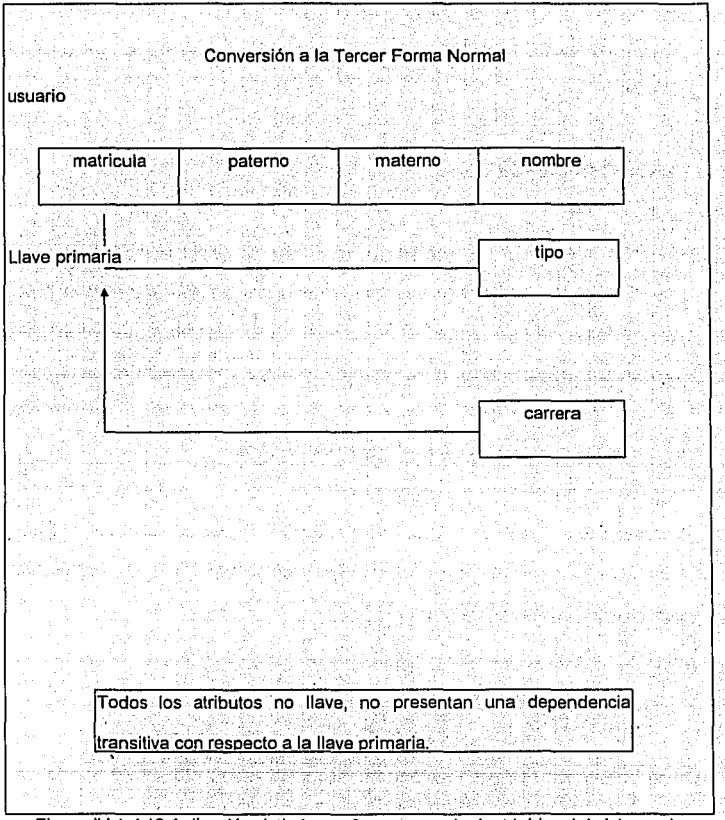


Figura IV.1.4.19 Aplicación de la tercer forma normal a las tablas del sistema de apartado, en específico a la tabla de usuario.

Como podemos ver en la figura IV.1.4.19, la dependencia transitiva no se presenta en la entidad usuario, pues los atributos no llave no dependen entre sí, únicamente dependen de la llave primaria. Por ello, la entidad usuario cumple con la tercer forma normal y no hay necesidad de realizar un cambio en la estructura de la tabla.

Ahora, en la figura IV.1.4.20, tenemos las demás tablas del sistema de apartado donde se analiza si existe una dependencia transitiva entre los diferentes atributos de las entidades.

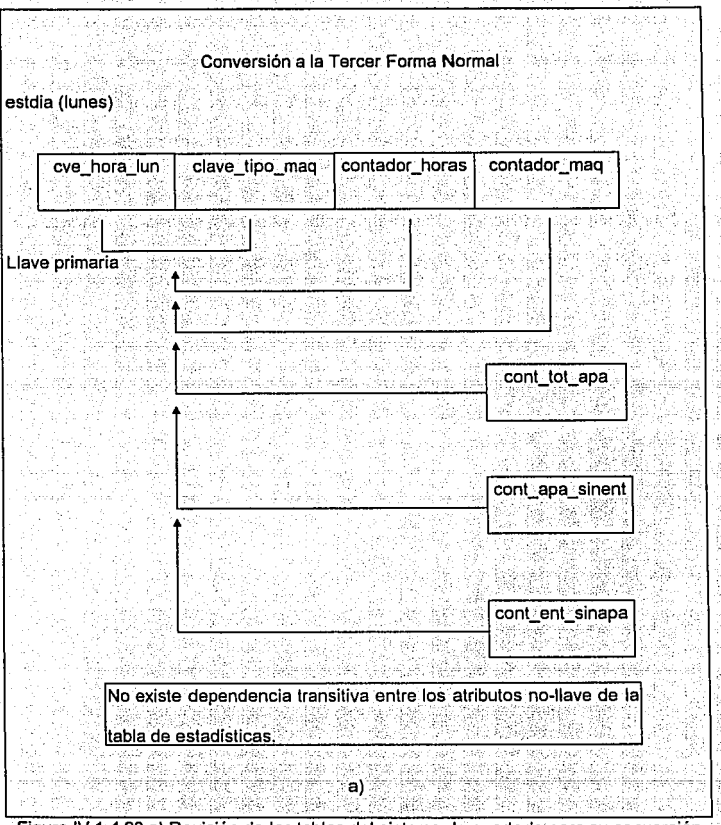


Figura IV.1.4.20 a) Revisión de las tablas del sistema de apartado para su conversión a la tercer forma normal.

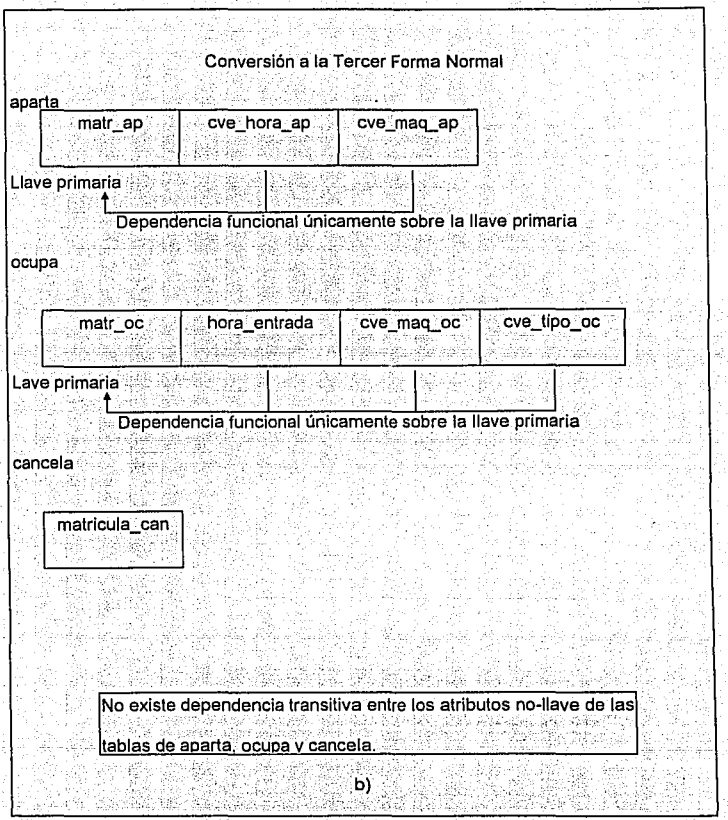


Figura IV.1.4.20 b) Revisión de las tablas del sistema de apartado para su conversión a la tercer forma normal.

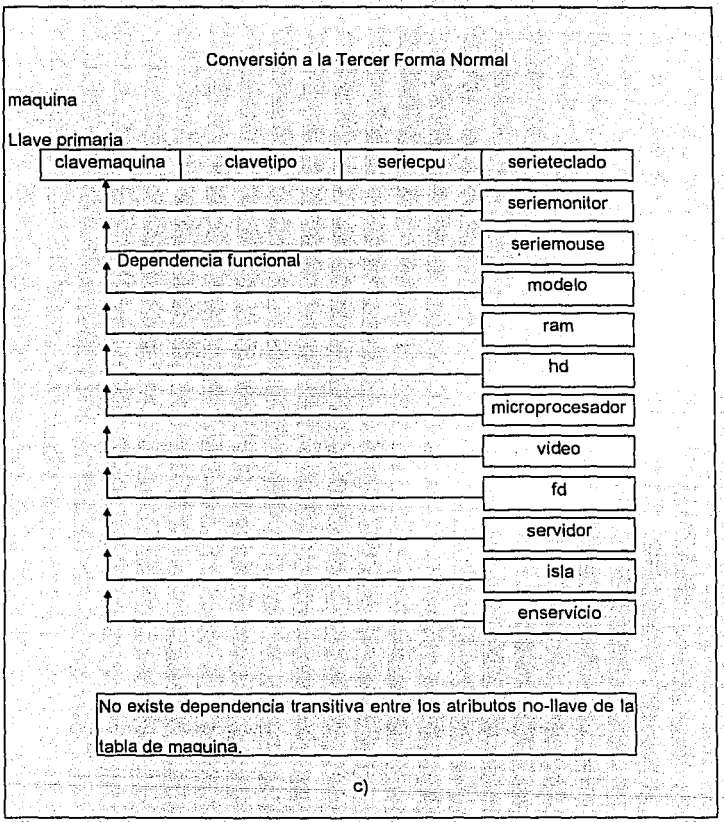


Figura IV.1.4.20 c) Revisión de las tablas del sistema de apartado para su conversión a la tercer forma normal.

Como podemos observar de las gráficas en la figura IV.4.1.20, tenemos que las tablas del sistema de apartado cumplen con la tercer forma normal al no presentar dependencia transitiva entre los atributos no llave que las conforman.

En resumen:

- Una tabla en la primer forma normal no contiene columnas repetidas.
- Una tabla en la segunda forma normal está en la primer forma normal y contiene solamente columnas que dependen de la llave primaria.
- Una tabla en la tercer forma normal está en la segunda forma normal y contiene únicamente columnas que son dependientes no-transitivas de la llave primaria.

CONSIDERACIONES EN LA NORMALIZACION

- No sobre-normalizar.
- Un modelo de datos normalizado, puede no mantener todos los objetivos del diseño.
- Una desnormalización selectiva puede ser usada para considerar mejoras en el rendimiento de la aplicación.

IV.1.5 DISEÑO Y CONSTRUCCION DE LA BASE DE DATOS

A partir del punto anterior, la definición de las tablas que conforman a la base de datos del sistema de apartado, quedaría como sigue, de acuerdo al formato que maneja Btrieve en su estructura de archivos:

Descripción de tablas

TABLE:	aparta
LOCATION:	C:\tp\bin\aparta.dat
DESCRIPCION :	Contiene la información de Apartados

COLUMN NAME	TYPE	SIZE
matr_ap	string	6
cve_hora_ap	string	2
cve_maq_ap	string	3

LLAVES				
PATH	COLUMN	DUPS	CASE	SEG
0	matr_ap	no	yes	no
1	cve_hora_ap	no	yes	yes
1	cve_maq_ap	no	yes	yes
1	cve_tipo_ap	no	yes	no
2	cve_hora_ap	yes	no	no
3	cve_hora_ap	yes	yes	yes
3	cve_tipo_ap	yes	yes	no
4	cve_maq_ap	yes	yes	no

Descripción de tablas

TABLE	cancela	
LOCATION:	C:\tp\bin\cancela.dat	
DESCRIPCION :	Contiene la información de Apartados Cancelados	
COLUMN NAME	TYPE	SIZE
matricula_can	string	6

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	matricula_can	no	no	no

Descripción de tablas

TABLE:	horas	
LOCATION:	C:\tp\bin\horas.dat	
DESCRIPCION :	Contiene el Catálogo de Horas al día.	
COLUMN NAME	TYPE	SIZE
clavehora	string	2
horas	string	13

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	clavehora	no	no	no

Descripción de tablas

TABLE	maquina	
LOCATION	C:\tp\bin\maquina.dat	
DESCRIPCION	Contiene el Catálogo de Máquinas	
COLUMN NAME	TYPE	SIZE
claveMaq	string	3
claveTipo	string	1
serieCPU	string	15
serieTeclado	string	15
serieMonitor	string	15
serieMouse	string	15
modelo	string	15
RAM	string	3
HD	string	4
microprocesador	string	6
video	string	5
FD	string	5
servidor	string	2
isla	string	2
EnServicio	string	1

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	claveMaq	no	no	no
1	claveTipo	yes	yes	yes
1	EmServicio	yes	yes	no

Descripción de tablas

TABLE:	ocupa	
LOCATION:	C:\tp\bin\ocupa.dat	
DESCRIPCION :	Contiene la información de Usuarios dentro del C.E.C..	
COLUMN NAME	TYPE	SIZE
matr_oc	string	6
hora_entrada	string	2
cve_maq_oc	string	3
cve_tipo_oc	string	1

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	matr_oc	no	no	no
1	cve_maq_oc	no	no	no
2	hora_entrada	yes	no	no
3	cve_tipo_oc	yes	no	no

Descripción de tablas

TABLE:	servidor
LOCATION:	C:\tp\bin\servidor.dat
DESCRIPCION	Contiene el Catálogo de Servidores.

COLUMN NAME	TYPE	SIZE
claveSer	string	2
nombreSer	string	20
dirIP1	string	3
dirIP2	string	3
dirIP3	string	3
dirIP4	string	3
dirEth	string	15

LLAVES :

PATH	COLUMN	DUPS	CASE	SEG
0	claveSer	no	no	no
1	nombreSer	yes	yes	no
2	dirEth	no	no	no
3	dirIP1	no	no	yes
3	dirIP2	no	no	yes
3	dirIP3	no	no	yes
3	dirIP4	no	no	no

Descripción de tablas

TABLE:	tipoMaq
LOCATION	C:\tp\bin\tipoMaq.dat
DESCRIPCION :	Contiene el Catálogo de tipos de Máquinas.

COLUMN NAME	TYPE	SIZE
claveTipoMaq	string	1
DescripTipoMaq	string	20

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	claveTipoMaq	no	no	no
1	DescripTipoMaq	yes	yes	no

Descripción de tablas

TABLE:	tipousu
LOCATION:	C:\tp\bin\tipousu.dat
DESCRIPCION :	Contiene el Catálogo de tipos de Usuario.

COLUMN NAME	TYPE	SIZE
claveTipoUsu	string	1
descripTipoUsu	string	20

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	claveTipoUsu	no	yes	no
1	descripTipoUsu	yes	yes	no

Descripción de tablas

TABLE:	usuario
LOCATION	C:\tp\bin\usuario.dat
DESCRIPCION	Contiene el Catálogo de Usuarios.

COLUMN NAME	TYPE	SIZE
matricula	string	6
paterno	string	20
materno	string	20
nombre	string	20
tipo	string	1
carrera	string	6

LLAVES

PATH	COLUMN	DUPS	CASE	SEG
0	matricula	no	no	no
1	paterno	yes	yes	yes
1	materno	yes	yes	yes
1	nombre	yes	yes	bi
2	paterno	no	yes	yes
2	materno	no	yes	yes
2	nombre	no	yes	yes
2	matricula	no	yes	no
3	matricula	no	yes	yes
3	carrera	no	yes	no

Descripción de tablas

TABLE:	estlun.dat
LOCATION:	C:\tp\bin\estlun.dat
DESCRIPCION :	Contiene la estadística de acceso al C.E.C. con y sin apartados por hora al día.

COLUMN NAME	TYPE	SIZE
cve_hora_est_lun	integer	2
clave_tipo_maq	integer	2
contador_horas	integer	2
contador_maq	integer	2
cont_tot_apa	integer	2
cont_apa_sinent	integer	2

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	cve_hora_est_lun	no	no	no

Descripción de tablas

TABLE:	estmar.dat
LOCATION:	C:\tp\bin\estmar.dat
DESCRIPCION :	Contiene la estadistica de acceso al C E C con y sin apartados por hora al día.

COLUMN NAME	TYPE	SIZE
cve_hora_est_mar	integer	2
clave_tipo_maq	integer	2
contador_horas	integer	2
contador_maq	integer	2
cont_tot_apa	integer	2
cont_apa_sinent	integer	2

LLAVES :

PATH	COLUMN	DUPS	CASE	SEG
0	cve_hora_est_mar	no	no	no

Descripción de tablas

TABLE:	estmie.dat
LOCATION:	C:\tp\bin\estmie.dat
DESCRIPCION:	Contiene la estadistica de acceso al C.E.C. con y sin apartados por hora al día.

COLUMN NAME	TYPE	SIZE
cve_hora_est_mie	integer	2
clave_tipo_maq	integer	2
contador_horas	integer	2
contador_maq	integer	2
cont_tot_apa	integer	2
cont_apa_sinent	integer	2

LLAVES :

PATH	COLUMN	DUPS	CASE	SEG
0	cve_hora_est_mie	no	no	no

Descripción de tablas

TABLE:	estjue dat
LOCATION	C \tp\bin\estjue dat
DESCRIPCION	Contiene la estadística de acceso al C E C con y sin apartados por hora al día.

COLUMN NAME	TYPE	SIZE
cve_hora_est_jue	integer	2
clave_tipo_maq	integer	2
contador_horas	integer	2
contador_maq	integer	2
cont_tot_apa	integer	2
cont_apa_sinent	integer	2

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	cve_hora_est_jue	no	no	no

Descripción de tablas

TABLE:	estvie.dat
LOCATION:	C:\tp\bin\estvie.dat
DESCRIPCION:	Contiene la estadística de acceso al C.E.C. con y sin apartados por hora al día.

COLUMN NAME	TYPE	SIZE
cve_hora_est_vie	integer	2
clave_tipo_maq	integer	2
contador_horas	integer	2
contador_maq	integer	2
cont_tot_apa	integer	2
cont_apa_sinent	integer	2

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	cve_hora_est_vie	no	no	no

Descripción de tablas

TABLE	estsab dat	
LOCATION	C \tp\bin\estsab dat	
DESCRIPCION	Contiene la estadística de acceso al C E C con y sin apartados por hora al dia	
COLUMN NAME	TYPE	SIZE
cve_hora_est_sab	integer	2
clave_tipo_maq	integer	2
contador_horas	integer	2
contador_maq	integer	2
cont_tot_apa	integer	2
cont_apa_sinent	integer	2

LLAVES :				
PATH	COLUMN	DUPS	CASE	SEG
0	cve_hora_est_sab	no	no	no

A continuación se presentarán los módulos que conforman el sistema por medio de diagramas de flujo:

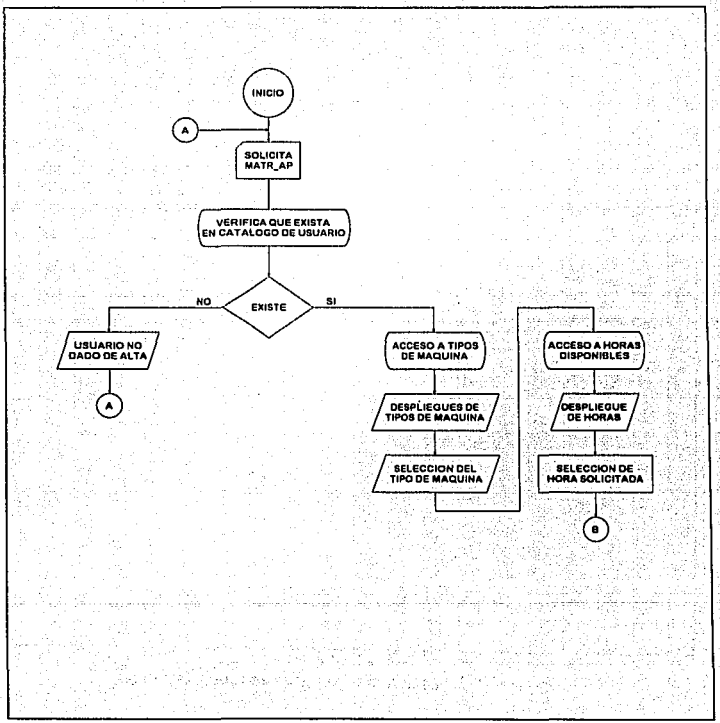


Figura IV.1.5.1 Apartado de estaciones (primera parte)

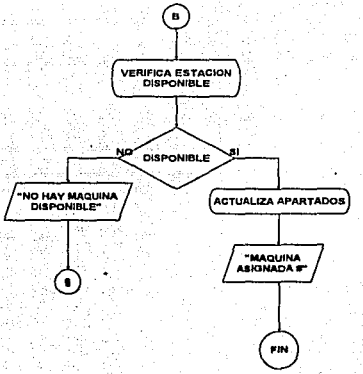


Figura IV.1.5.2 Apartado de estaciones (segunda parte)

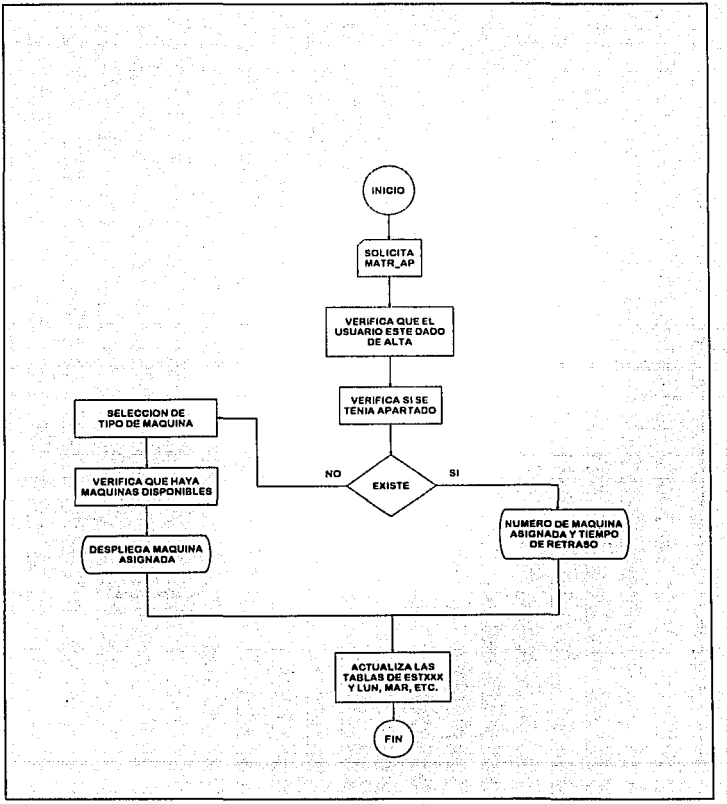


Figura IV.1.5.3 Entradas de apartado

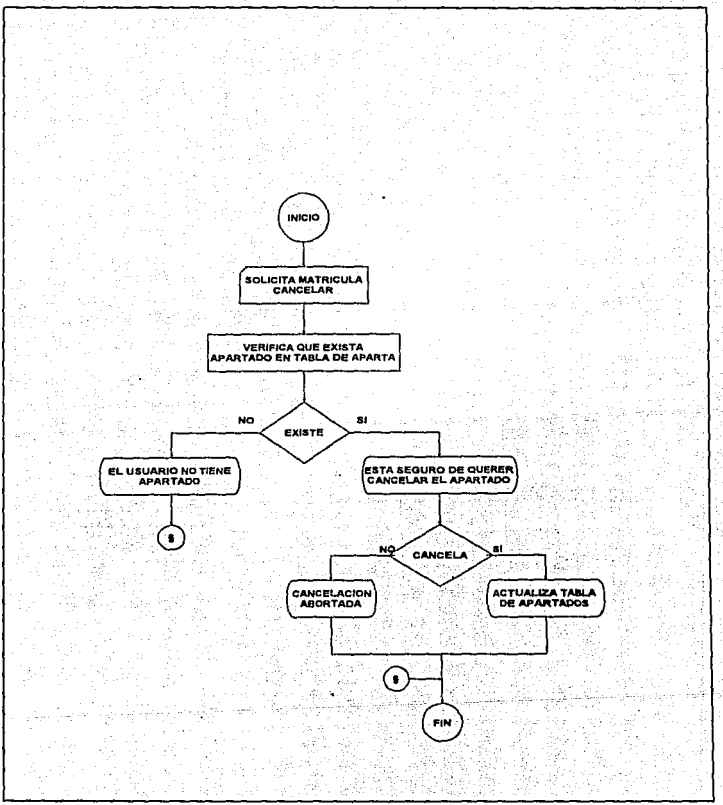


Figura IV.1.5.4 Apartado Cancelaciones

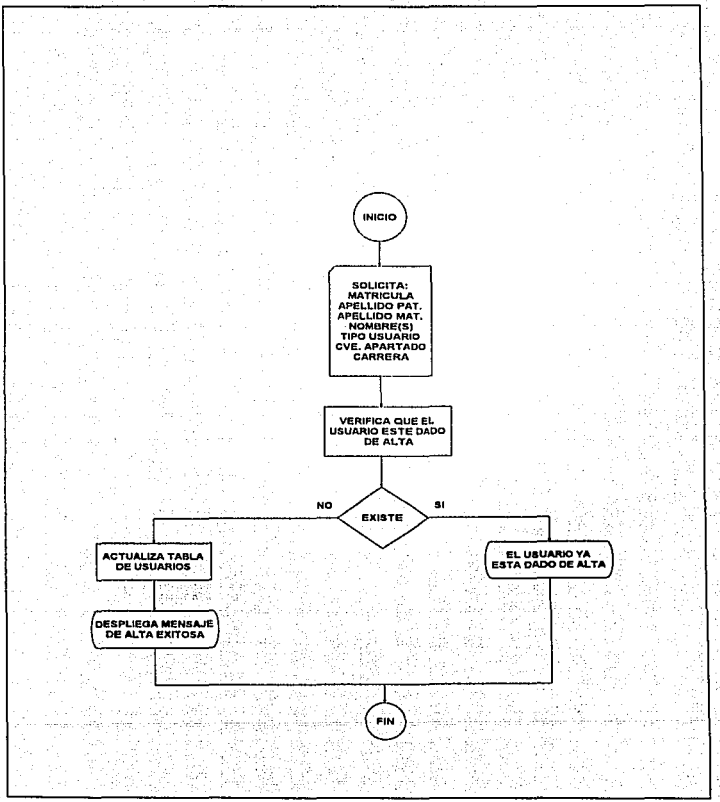


Figura IV.1.5.5 Mantenimiento de Usuarios altas

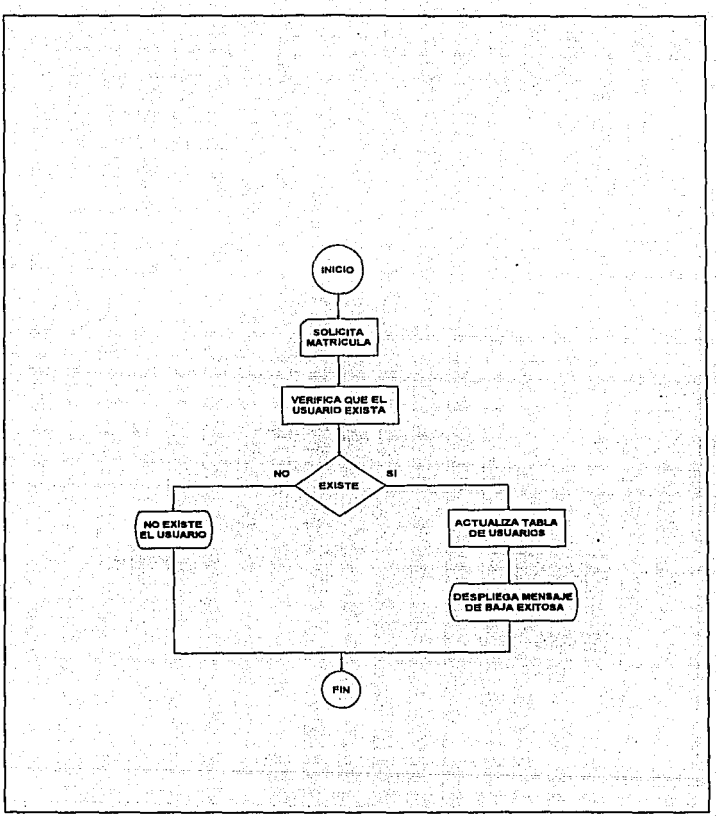


Figura IV.1.5.6 Mantenimiento de Usuarios Bajas

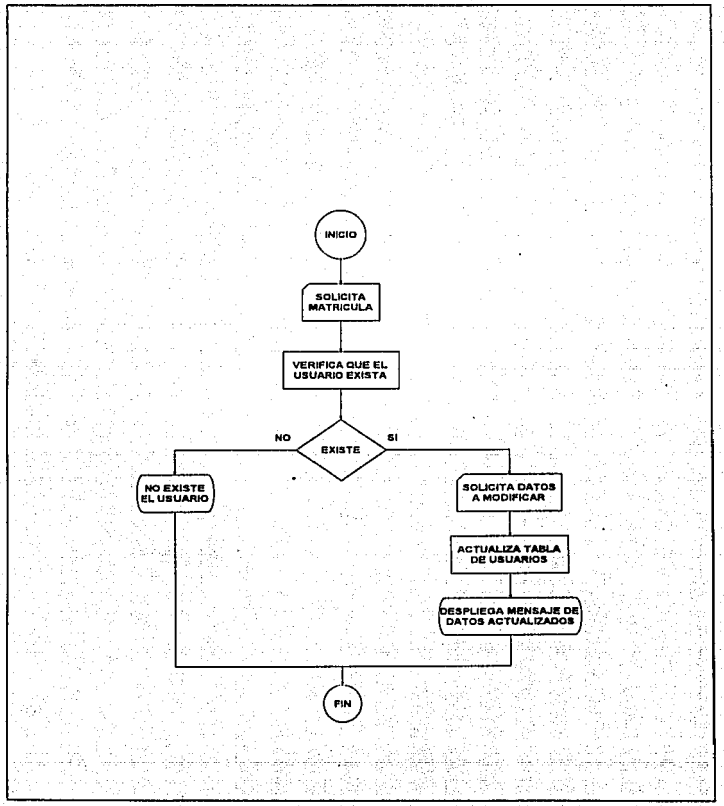


Figura IV.1.5.7 Mantenimiento de Usuarios cambios

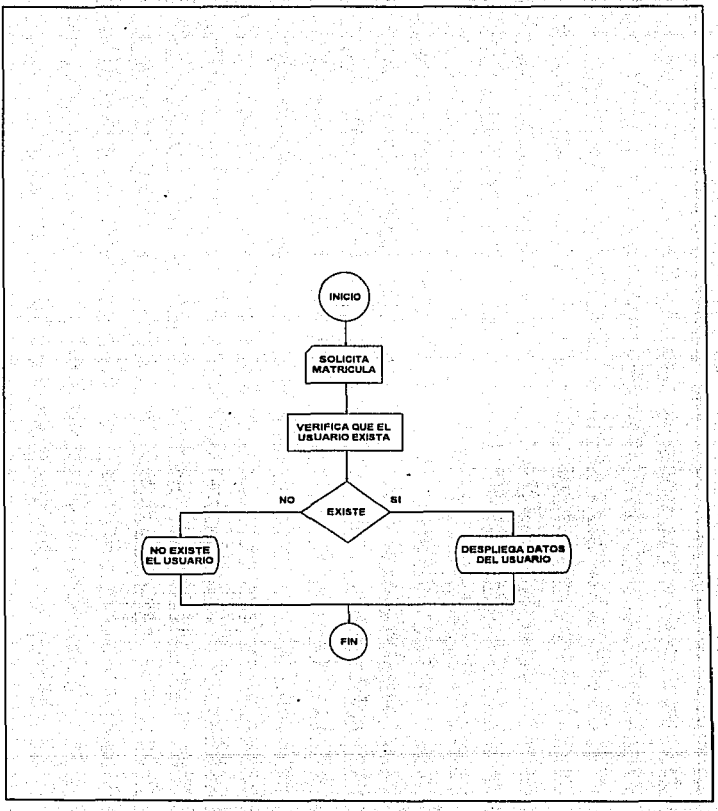


Figura IV.1.5.8 Mantenimiento de Usuarios Consulta

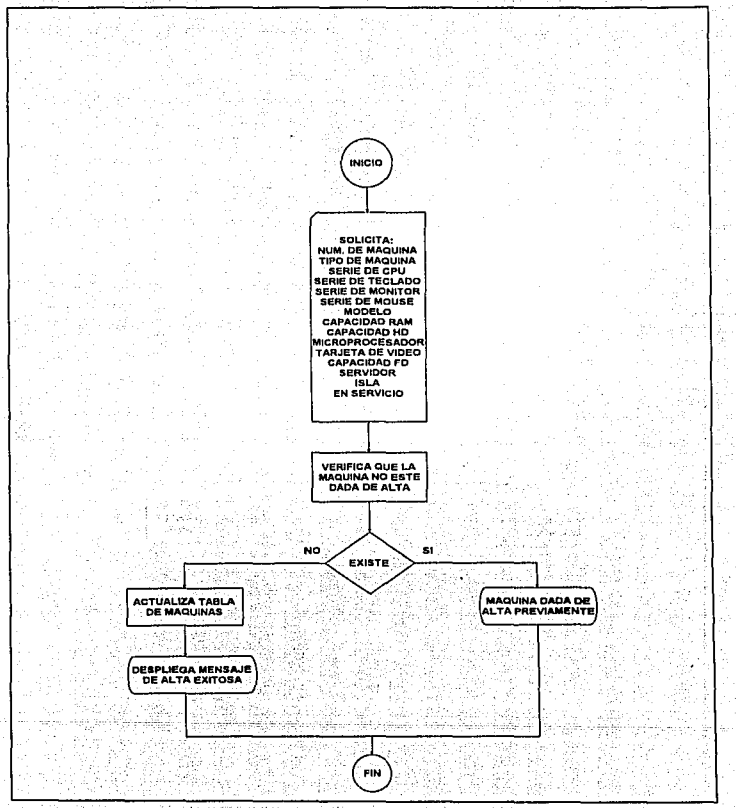


Figura IV.1.5.9 Mantenimiento de Máquinas altas

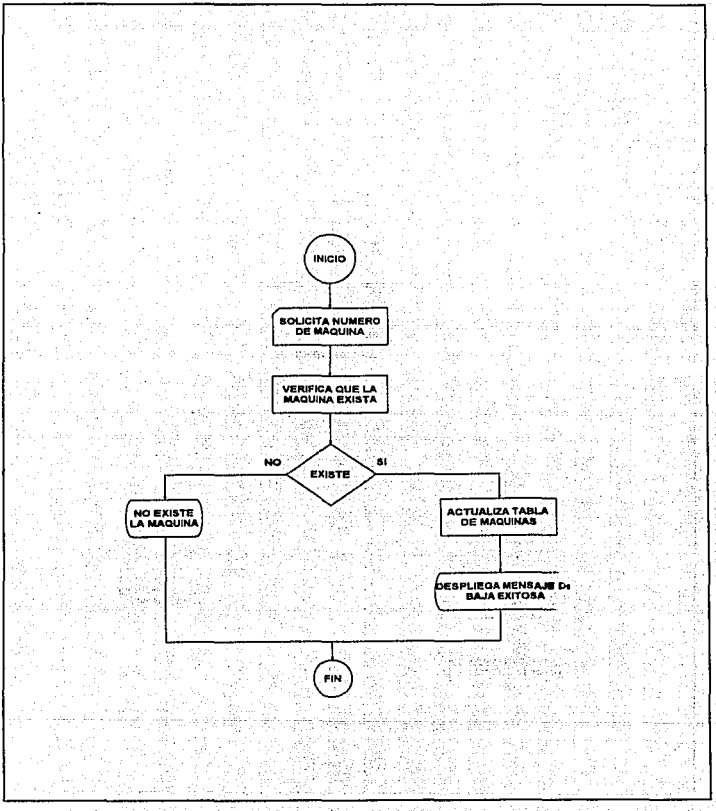


Figura IV.1.5.10 Mantenimiento de Máquinas Bajas

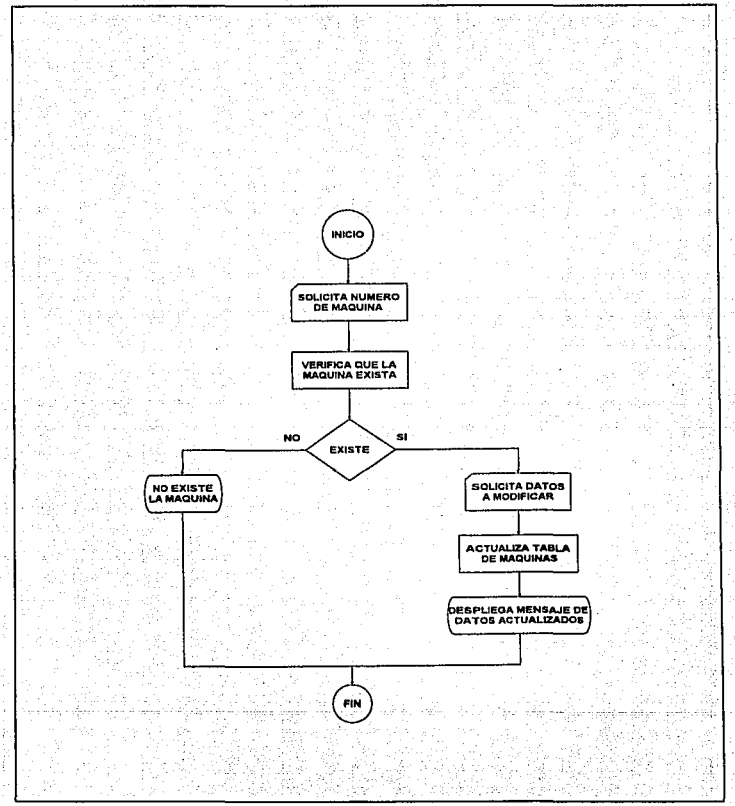


Figura IV.1.5.11 Mantenimiento de Máquinas Cambios

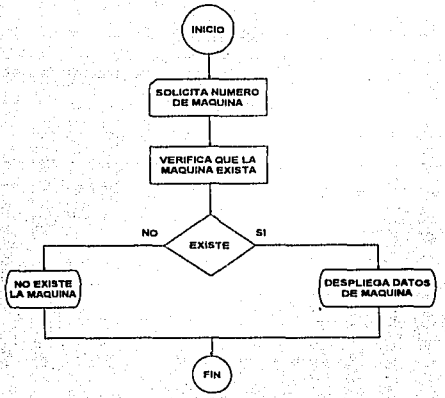


Figura IV.1.5.12 Mantenimiento de Máquinas Consultas

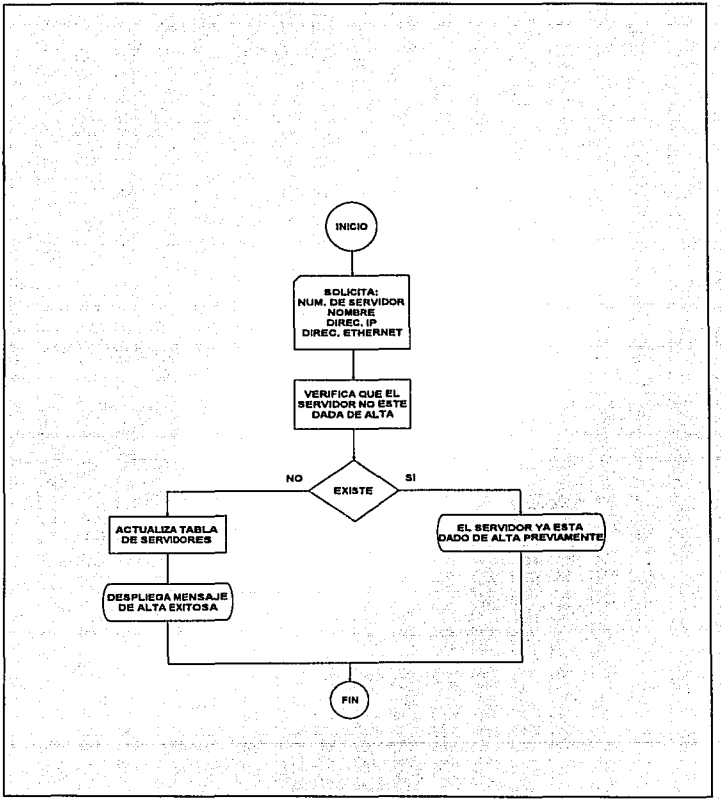


Figura IV.1.5.13 Mantenimiento de Servidores Altas

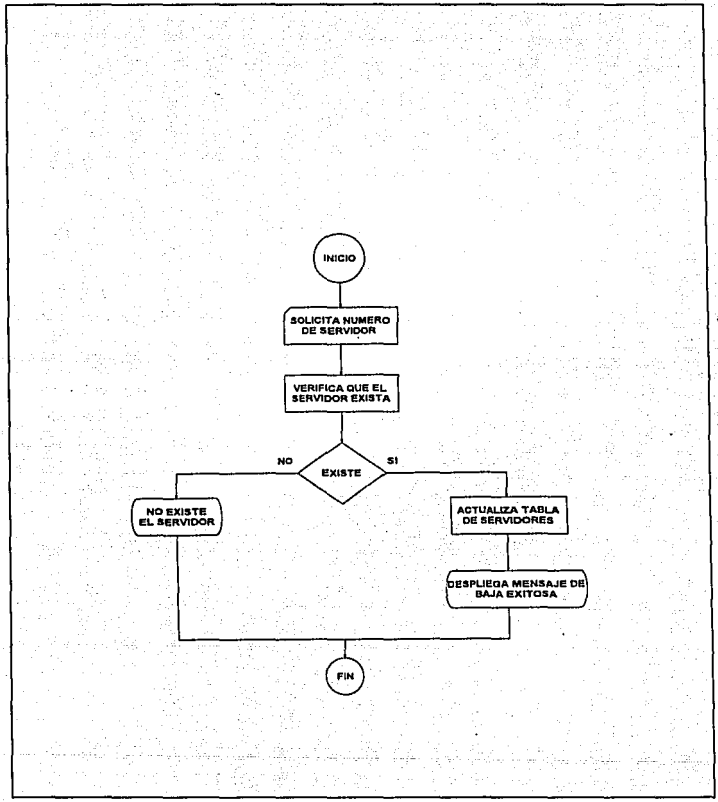


Figura IV.1.5.14 Mantenimiento de Servidores Bajas

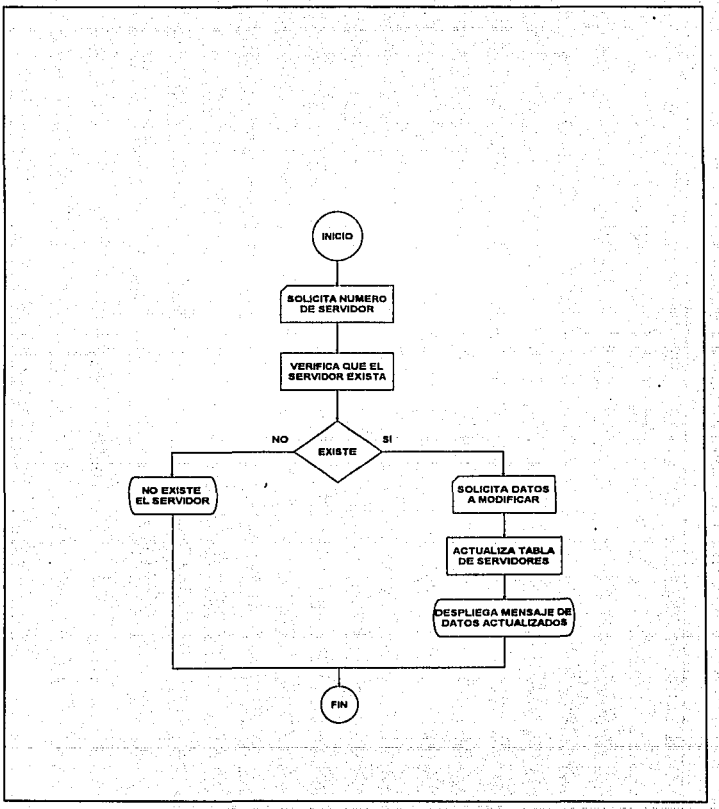


Figura IV.1.5.15 Mantenimiento de Servidores Cambios

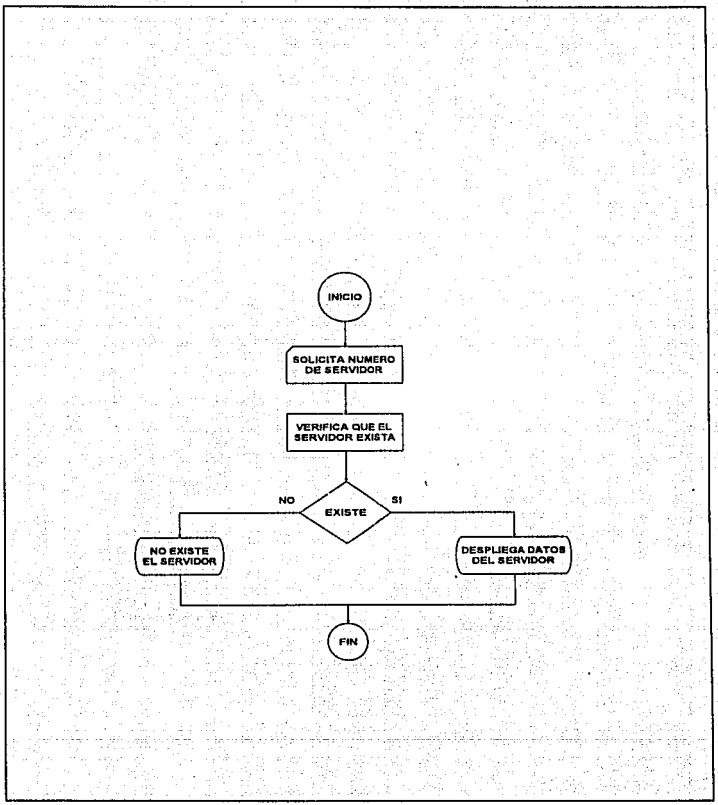


Figura IV.1.5.16 Mantenimiento de Servidores Consultas

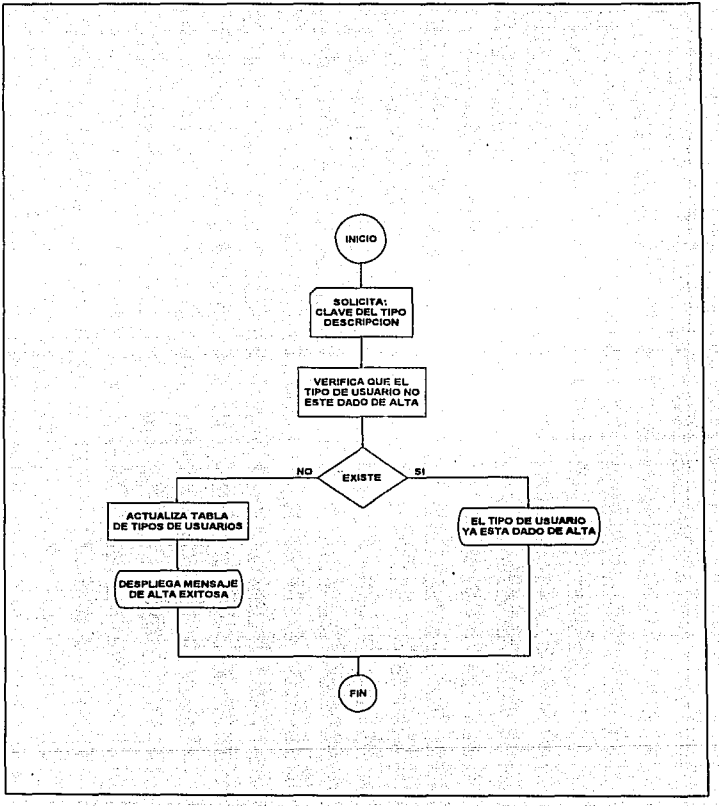


Figura IV.1.5.17 Mantenimiento de Tipos de usuarios Altas

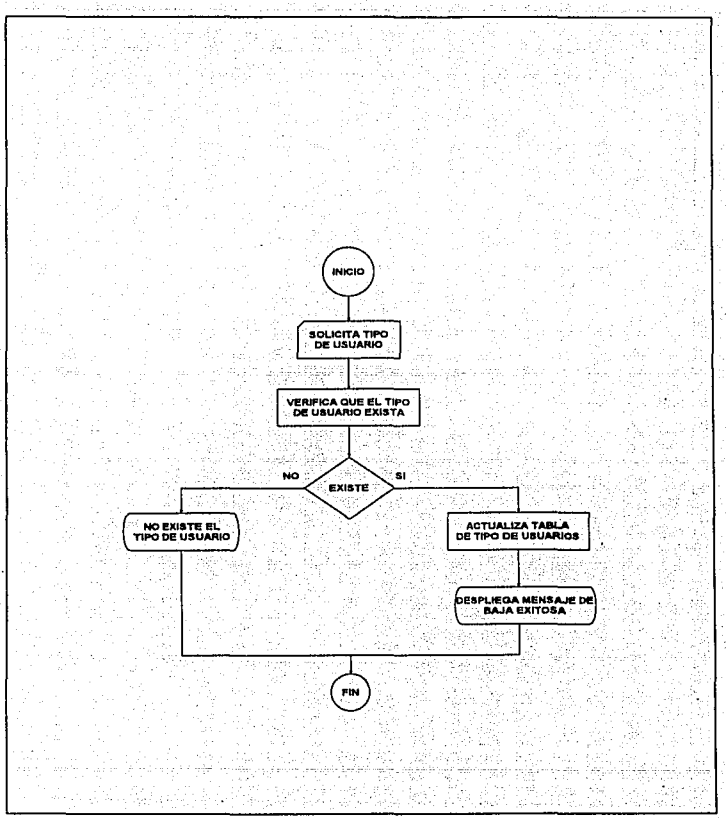


Figura IV.1.5.18 Mantenimiento de Tipos de usuarios Bajas

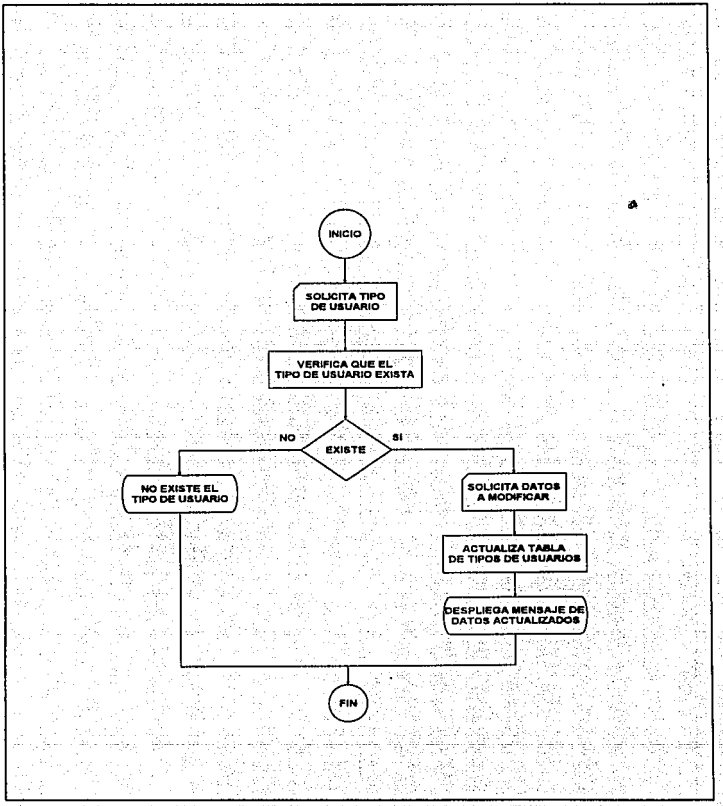


Figura IV.1.5.18a Mantenimiento de Tipos de usuarios Cambios

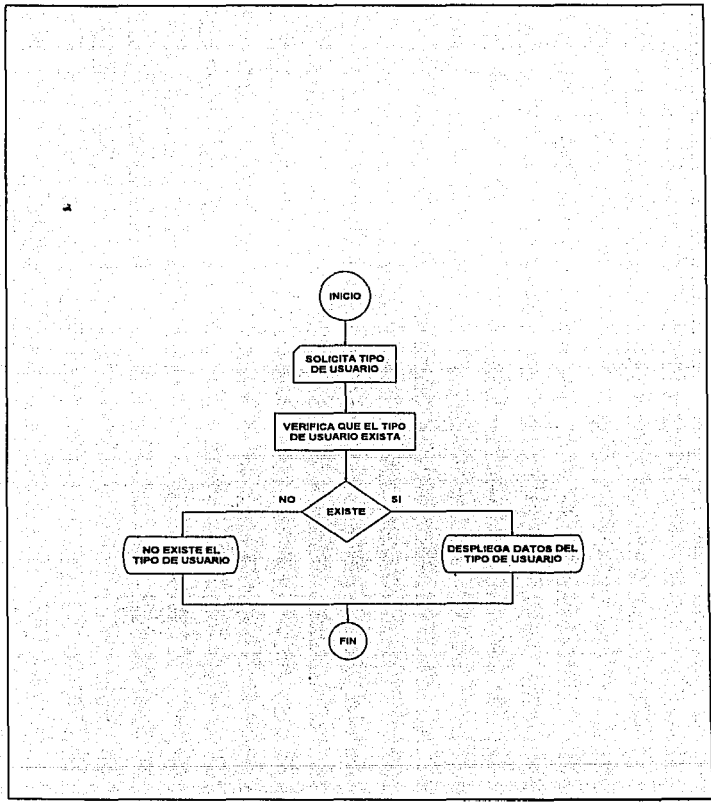


Figura IV.1.5.19 Mantenimiento de Tipos de usuarios Consultas

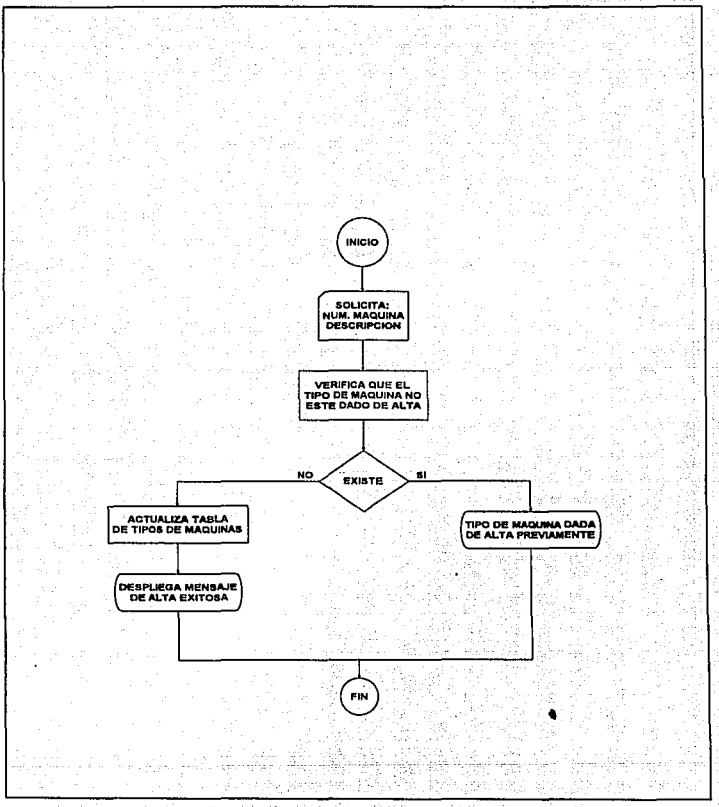


Figura IV.1.5.20 Mantenimiento de Tipos de máquinas Altas

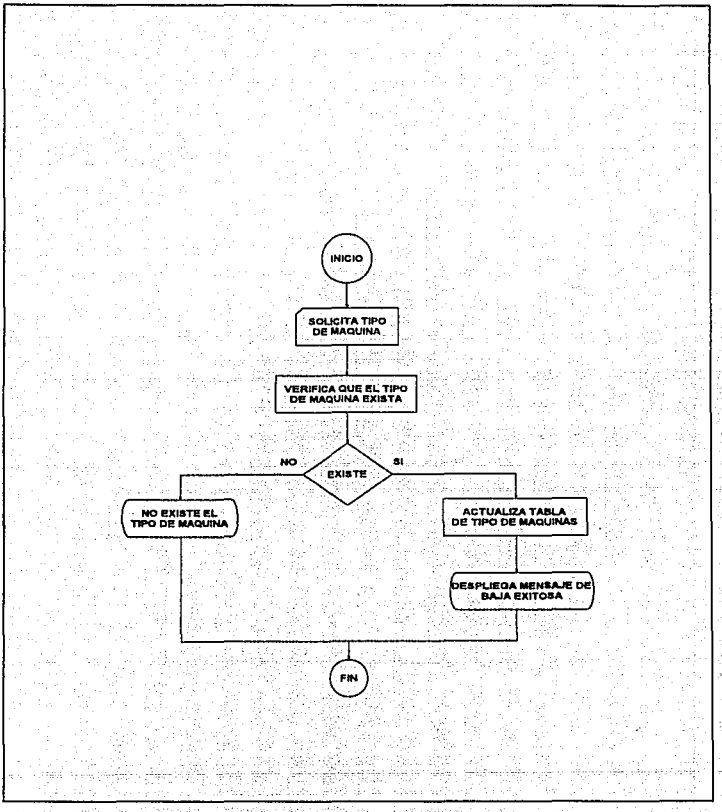


Figura IV.1.5.21 Mantenimiento de Tipos de máquinas Bajas

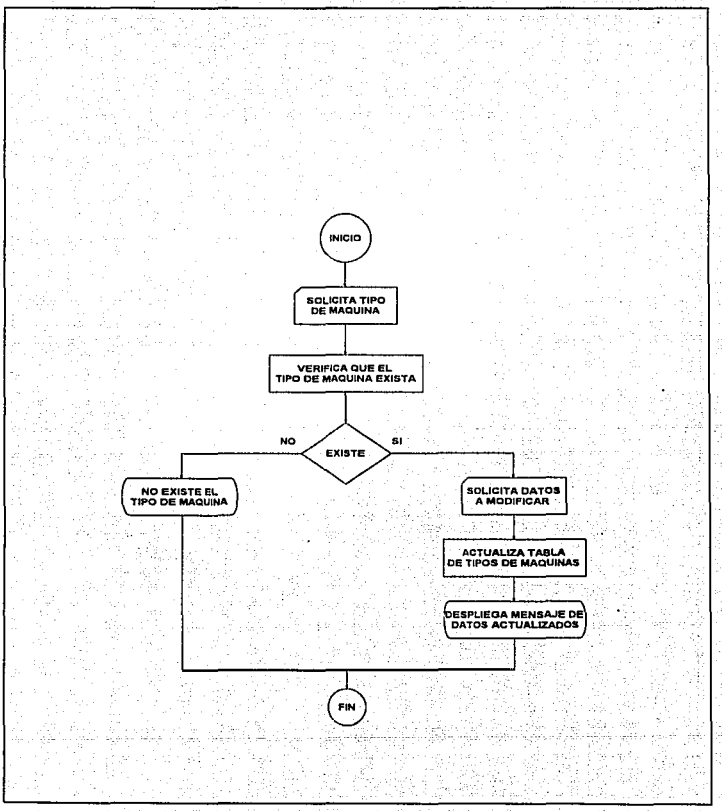


Figura IV.1.5.22 Mantenimiento de Tipos de máquinas Cambios

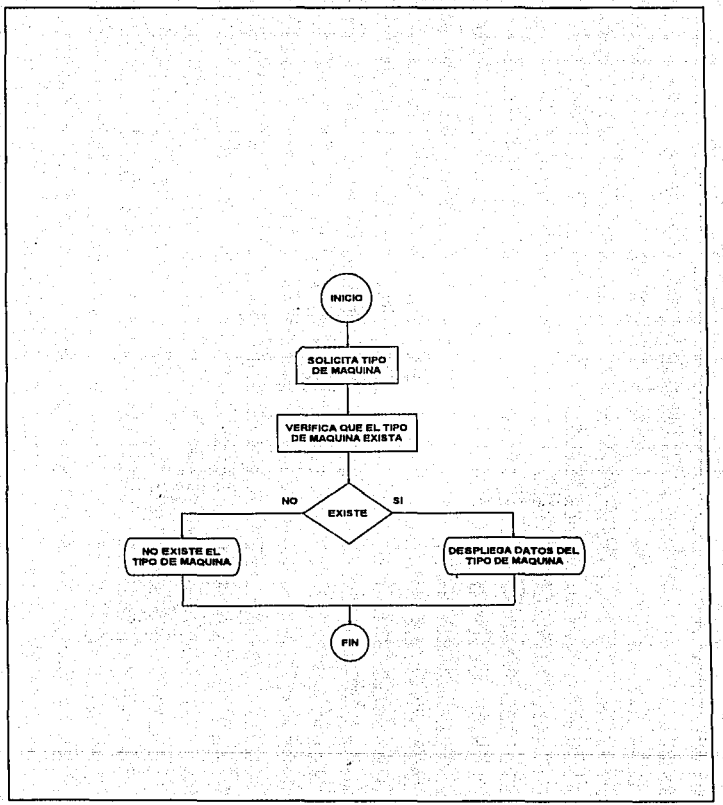


Figura IV.1.5.23 Mantenimiento de Tipos de máquinas Consultas

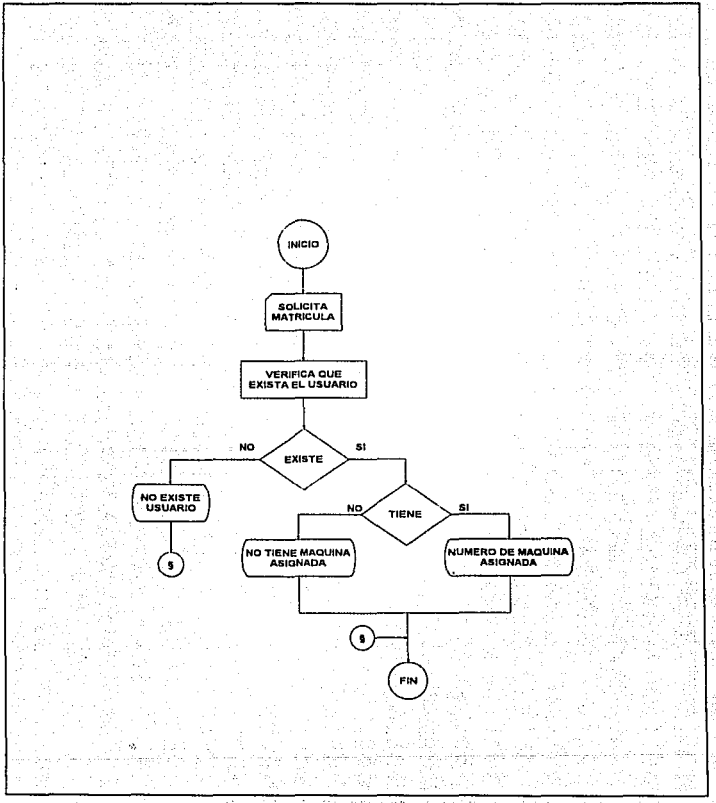


Figura IV.1.5.24 Consulta de usuarios

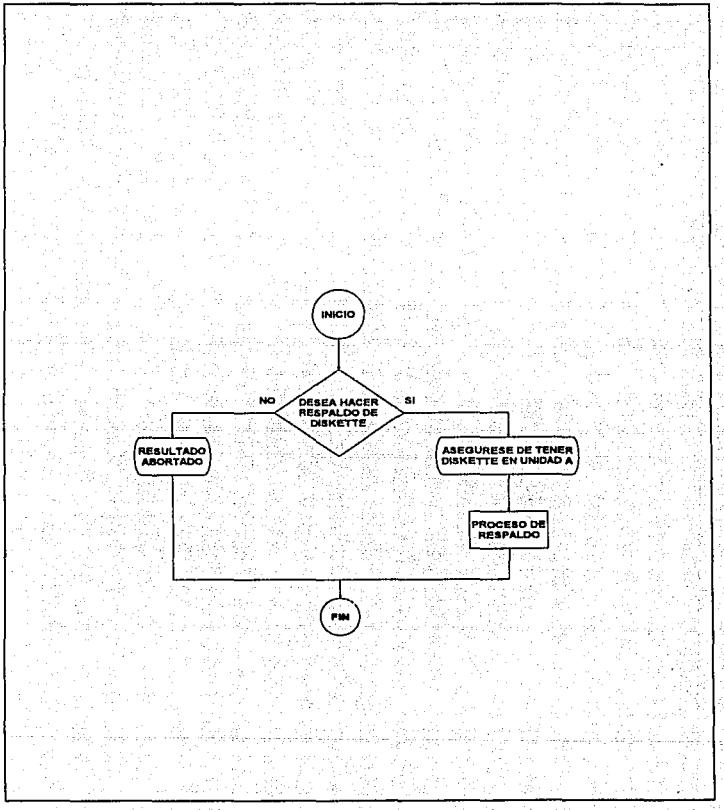


Figura IV.1.5.25 Respaldo de información

IV.1.6 DISEÑO E IMPLEMENTACION DE LOS DIVERSOS MODULOS DE SERVICIO DE LA BASE DE DATOS

Una vez que se tuvieron todos los elementos para la evaluación de los requisitos de diseño, fue posible elaborar detalladamente los distintos módulos que darían servicio a la base de datos y adicionalmente servirían para dar seguimiento a los reglamentos y solicitudes de los usuarios y ejecutivos involucrados.

Los módulos que se elaboraron fueron los siguientes:

- Apartados
- Mantenimiento
- Consulta
- Salida

Cada uno de los módulos, se subdivide en distintas partes que se explicarán conforme se mencione cada módulo.

Módulo de Apartados.

En ésta parte, se contempló ubicar la operación misma del sistema, ya que en ella se elaboran todos los procesos críticos del sistema y la operación cotidiana.

Las partes en que se divide éste módulo, son las siguientes:

- Apartados
- Cancelaciones
- Entradas
- Salidas

Apartados

La parte correspondiente a los apartados, tiene la función de elaborar los apartados de equipo con anticipación, por lo que sus funciones son las siguientes:

Solicitar matrícula del alumno que aparta la máquina.

Verificar que el usuario se encuentre dado de alta en el catálogo de usuarios.

Verificar que el usuario no se encuentre suspendido del servicio por alguna falta.

Verificar que el usuario no tenga un apartado a otra hora o se encuentre en el CEC en ese momento.

Solicitar el tipo de máquina que se desea apartar.

Solicitar la hora a la que se desea el apartado.

Verificar que se disponga de máquina a la hora y del tipo que se solicitó.

Realizar el apartado en el archivo de aparta.dat

Cancelaciones

La sección de cancelaciones realiza las siguientes funciones:

Solicita la matrícula del usuario que realiza la cancelación del apartado.

Verifica en el archivo de apartados, que el usuario tenga un apartado.

Borra del archivo de apartados, la información correspondiente a la matrícula que se indicó.

Entradas

La sección de entradas, realiza de manera general, las siguientes funciones:

Solicita la matrícula de la persona que entrará al CEC.

Verifica si la persona tiene un apartado.

Indica el número de máquina asignada si el usuario tiene apartado a la hora en que realiza la entrada.

Indica que el usuario no puede entrar si cuenta con un apartado a una hora distinta.

Si el usuario no tiene apartado previo, realiza el proceso de solicitud de datos idéntico al e un apartado, sin solicitar la hora del apartado, que se asume que es la actual. Indica igualmente, el número de máquina que se asigna para uso en esa hora.

Salidas.

En la sección de salidas, se realizan las siguientes funciones:

Solicita de la matrícula del usuario que desea registrar su salida.

Verifica que el usuario se encuentre en el CEC en ese momento.

Borra del archivo de ocupación al usuario si se encuentra en ese momento.

Indica si el usuario no se encuentra en el CEC en ese momento.

Módulo de Mantenimiento.

En el módulo de mantenimiento, se ofrece la posibilidad de dar el soporte a las tablas logísticas del sistema.

Cada una de las diversas opciones del menú de mantenimiento, cuentan con un submenú idéntico que contiene las opciones de Alta, Baja, Cambios y Consultas; las anteriores opciones, sirven para dar el mantenimiento a los diversos catálogos adicionando nuevos registros, borrando y alterando los ya existentes y visualizando la información de los mismos.

Los catálogos que se pueden manipular mediante el módulo de mantenimiento, son los siguientes:

Usuarios.

En esta sección, se manipula la información de los usuarios que emplean el CEC, los datos que se almacenan son Matrícula, Apellido paterno, Apellido Materno, Nombre(s), Tipo de usuario, Carrera y si el usuario puede apartar.

Máquinas.

Mediante la sección de mantenimiento de máquinas, se manipula la información concerniente a los equipos del CEC y los datos que se almacenan son: Número de máquina, Tipo de máquina, Número de serie del CPU (Unidad Central de Procesamiento), Número de serie del Teclado, Número de serie del Monitor, Número de Serie del Mouse, Modelo, Capacidad de RAM (Memoria de Acceso Aleatorio), Capacidad de HD (Disco Duro), Tipo de Microprocesador, Tarjeta de video, Capacidad de FD (Disco Flexible), Servidos al que se encuentra conectado, Isla en la que físicamente se encuentra el equipo y la bandera de si se encuentra en servicio o no.

Servidores.

En ésta sección, se maneja la información de los servidores existentes para conexión desde el CEC. Los datos importantes de cada servidor, son los siguientes: Número del servidor, Nombre del servidor, Dirección IP y Dirección Ethernet del mismo.

Tipos de Usuario.

Los tipos de usuario se manejan mediante esta sección, almacenando para ellos, la información de su clave y su descripción únicamente.

Tipos de Máquina.

Los distintos tipos de máquinas, se pueden encontrar en ésta sección y los datos que se almacenen son: Clave del tipo y Descripción.

Módulo de Consulta.

En el módulo de consulta, se realizan operaciones menos ligadas con la operación solicitada del sistema, pero que son de vital importancia para la operación del mismo. Las opciones del módulo de consulta son:

- Usuario
- Respaldo

En la consulta del usuario, se realizan las siguientes operaciones:

Solicita la matrícula del usuario que se desea consultar.

Verifica que se trate de un usuario válido (que esté registrado, que pueda apartar etc.)

Verifica si el usuario tiene un apartado.

Verifica si el usuario se encuentra en el CEC.

Muestra la máquina que tiene asignada y la hora de la asignación.

En la sección de respaldo, se realizan las siguientes actividades:

Crea un directorio de trabajo en el disco duro con el nombre formado por la fecha del día del respaldo. Ej. 19950318 equivale al 19 de marzo de 1995.

Copia los archivos de estadísticas diarias al directorio creado para tal propósito.

Solicita que se inserte un disquete a la unidad de disco a:

Copia los archivos de estadísticas diarias al disco flexible colocado en a:

Todas las rutinas que se encuentran en los módulos anteriores, contemplan que se trabaje simultáneamente en la unidad local y en una unidad de disco de red; por consecuencia, en cada acceso a la base de datos de btrieve, se dispone de un acceso a cada unidad de disco en que se encuentren los archivos.

La razón por la que se hicieron por duplicado los accesos a la base de datos, es que si por alguna circunstancia, se pierde el contacto con la red, los archivos en el cliente, se inhiben, pero la máquina que se encuentra situada físicamente en el CEC, continúa dando servicio hasta que se recupera la red, momento en el que se realiza una copia en el disco de la red de los archivos del disco duro, para restablecer el servicio remoto.

IV.1.7 DISEÑO DE LAS PANTALLAS DE CONSULTA BAJO UN AMBIENTE GRAFICO

La aplicación se desarrolló bajo ambiente Windows con Visual Basic 3.0 como herramienta para presentar un ambiente gráfico y fácil de interactuar con el usuario. Una de las grandes ventajas de un interface gráfico de usuario estriba en que el usuario puede interacturar con un conjunto estándar de objetos, como ventanas, botones y barras de desplazamiento. Las aplicaciones que utilizan estos objetos se comportan de una manera estándar, haciendo que las aplicaciones sean fáciles de aprender.

En un entorno GUI (graphical user interface, interface gráfico de usuario), el usuario interactúa con los objetos que hay en pantalla para iniciar los eventos (abrir una ventana, hacer clic en un icono, seleccionar un elemento de un menú), y haciendo eso controla la aplicación. Visual Basic traduce un evento iniciado por el usuario en una actividad programada, llamando a un procedimiento que está asociado con ese evento. El código que se proporciona para ese evento implementa la respuesta adecuada a la interacción del usuario con el objeto.

En el desarrollo de cualquier aplicación con Visual Basic encontramos las siguientes fases:

1. Establecer las propiedades del formulario
   - Ventana de formulario[1] y Ventana de propiedades[2]
2. Añadir objetos
3. Elegir nombres (este paso no es obligatorio)
4. Completar la ventana principal
5. Escribir el código
6. Procedimientos para eventos
   - Declaraciones de procedimiento
   - Declaraciones de variables
7. Ejecución del programa
8. Revisar diseño
9. Compilación del programa
10. Guardar el programa

Una vez que se tiene presente las fases que comprenden el desarrollo de una aplicación, Visual Basic contiene una serie de herramientas las cuales nos permiten realizar cada una de las fases anteriormente descritas para crear una aplicación gráfica. La descripción de las herramientas más útiles son las siguientes:

[1] Un formulario es una zona de visualización que corresponde a una ventana que se despliega cuando la aplicación esté funcionando. Cuando empieza un proyecto nuevo, Visual Basic crea un formulario vacío y le da el título Form1. A medida que se diseña la aplicación, el formulario sirve como un lienzo en el que se puede dibujar diversas partes de la aplicación. Los componentes de la aplicación que se colocan en elformulario se denominan *objetos* o *controles* -- cuadros de dibujo, botones de opciones y barras de desplazamiento, por ejemplo.

[2] Las propiedades de Visual Basic son mecanismos formales que sirven para describir los atributos de un objeto.

Formularios

Los formularios son lienzos en los que se puede crear la aplicación visualmente. Cada formulario se corresponde con una ventana cuando se ejecuta la aplicación.

Propiedades

Hay muchas propiedades que pueden afectar a la apariencia y comportamiento de un formulario cuando éste aparece en pantalla; a continuación se presentan las de uso más frecuente. Las propiedades pueden establecerse usando tanto la ventana Propiertes a medida que se realiza el diseño de una aplicación, como escribiendo el código del programa, para que se establezca las propiedades cuando el programa se esté ejecutando. La lista de propiedades es la siguiente:

- BorderStyle
- Caption
- ControlBox
- Enabled
- FontBold, FontItalic, FontStrikethru, FontUnderline
- FontName
- FontSize
- ForeColor

- Height, Width
- Icon
- MaxButton, MinButton
- Name
- Picture
- Top, Left
- Visible
- WindowState

Eventos

Son los eventos más comunes que procesa un formulario. Los eventos son:

- Click
- DbClick
- Load

Procedimientos y métodos

Los métodos más interesantes de un formulario son los métodos gráficos ver figura IV.1.7.1. La lista de métodos y procedimientos más comunes son los siguientes:

- Cls
- LoadPicture
- Print

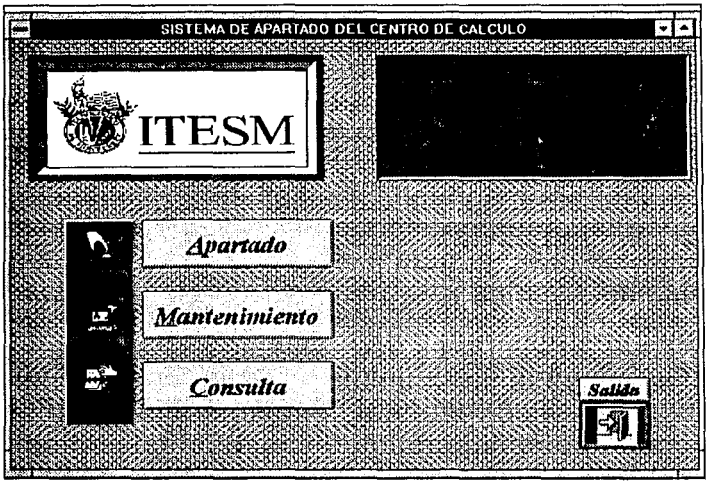


Figura IV.1.7.1 Formulario utilizando un procedimiento gráfico.

Cuadros de dibujos e imágenes

Los controles Picture Box (cuadro de dibujo) e Image (imagen) permiten situar información gráfica en una posición específica del formulario. El control cuadro de

dibujo es el más flexible de los dos y, consecuentemente requiere más memoria y tiempo cuando se utiliza. Los cuadros de dibujo son los más adecuados para entornos dinámicos - cuando se dibujan gráficos directamente en la pantalla mientras el programa está funcionando o cuando se anima un icono moviéndolo por la pantalla- los objetos imagen son mejores para situaciones estáticas. Ver figura IV.1.7.2

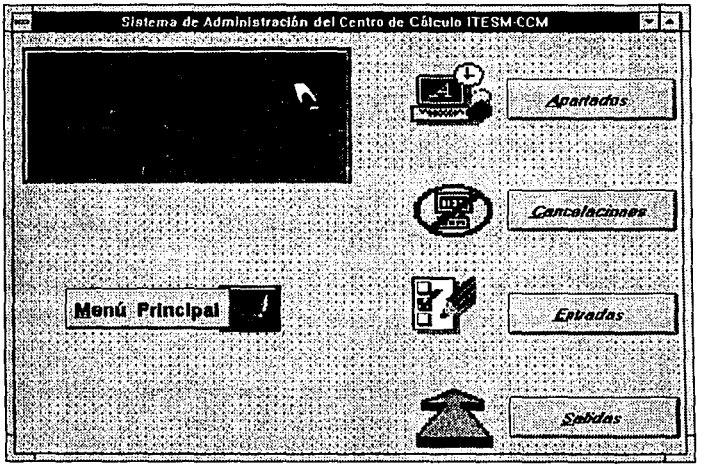


Figura IV.1.7.2 Utilización de cuadros e imágenes.

Los controles cuadro de dibujo e imagen tienen las propiedades Enabled, Height, Left, Name, Picture, Top, Visible y Width. A excepción de las propiedades para

coordenadas como Heigth, Left, Top y Width se miden con respecto a la localización del objeto en el formulario, no en coordenadas absolutas de pantalla. (Esto es así para todos los controles creados en la caja de herramientas). La propiedad Picture puede establecerse para que muestre un bitmap o un icono: durante el diseño utiliza la ventana de propiedades y el cuadro de diálogo Load Picture.

Etiquetas

Una etiqueta proporciona un área donde se puede presentar texto que no pueda ser editado por el usuario. El contenido se establece modificando la propiedad Caption de la etiqueta. No se puede imprimir ni dibujar en una etiqueta. Las etiquetas se crean con la herramienta Label de la caja de herramientas de Visual Basic.

Cuadros

Los cuadros de texto crean un área de pantalla en la cual el usuario puede introducir texto.

Marcos

Los marcos existen para separar grupos de otros objetos en la pantalla. Los marcos proporcionan una separación visual así como la posibilidad de activarlos y

desactivarlos en grupo. Para poner un marco se utiliza la herramienta Frame de la caja de herramientas de Visual Basic.

Cuadros de lista

Los cuadros de lista permiten ofrecer al usuario una serie de opciones para que elija. El cuadro de lista muestra las opciones que están disponibles, y el usuario selecciona un elemento, o entrada de la lista, haciendo clic sobre él.

Cuadros combinados

Los cuadros combinados se crean mediante la herramienta Combo Box de la caja de herramientas de Visual Basic. El cuadro combinado se llama así porque puede combinar un cuadro de texto y un cuadro de lista en un solo control.

A continuación se muestra la Fig. IV.1.7.3 como ejemplo de algunos de los objetos anteriormente descritos.

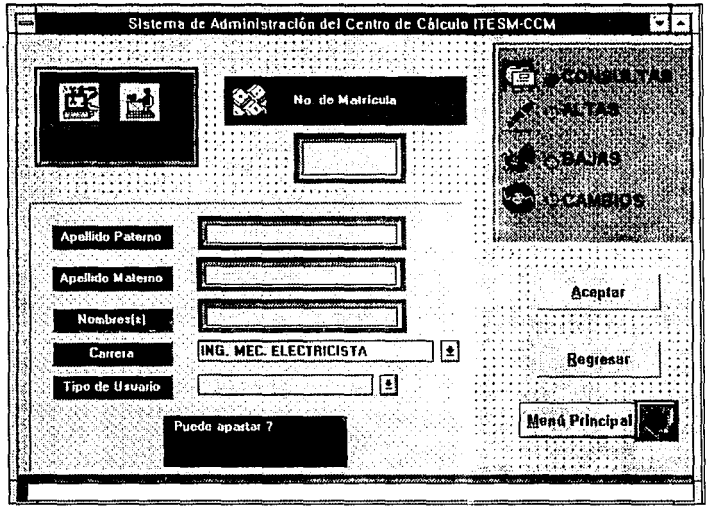
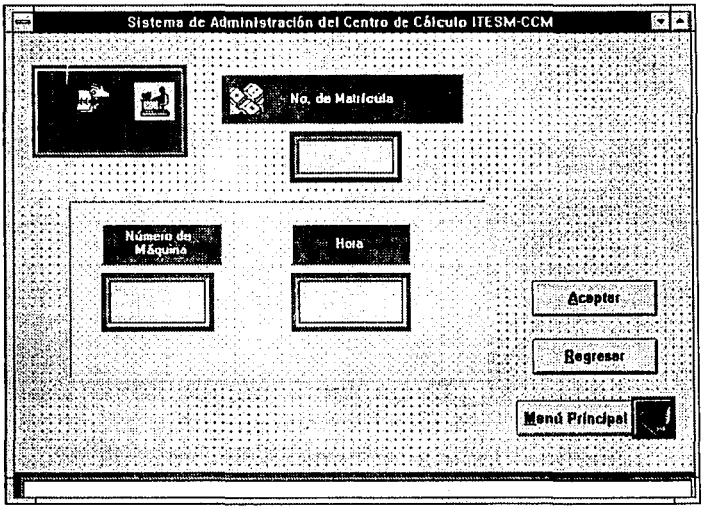


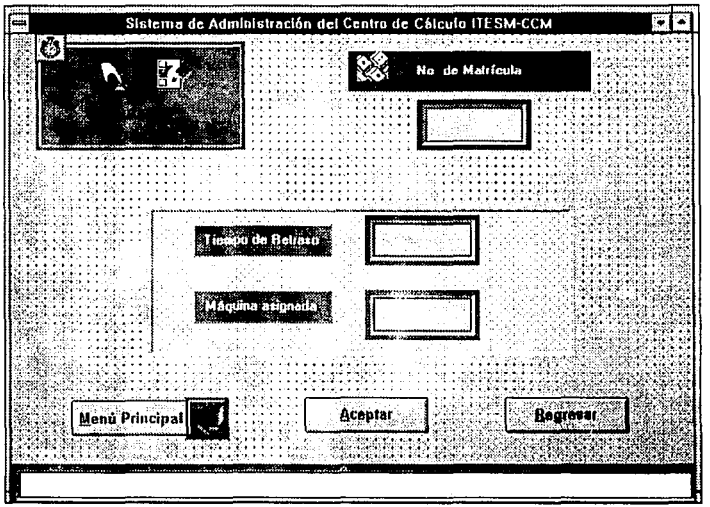
Figura IV.1.7.3 Ejemplo de objetos como: Etiquetas y Cuadros de Texto.

Pantallas generadas

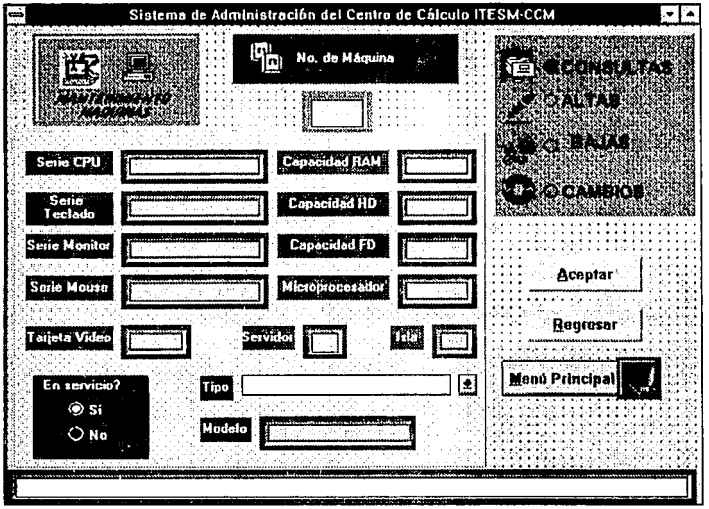
El resto de las pantallas generadas con las herramientas de Visual Basic para la Aplicación del Sistema de Apartados del Centro de Cálculo son las siguientes, se presentan las principales:



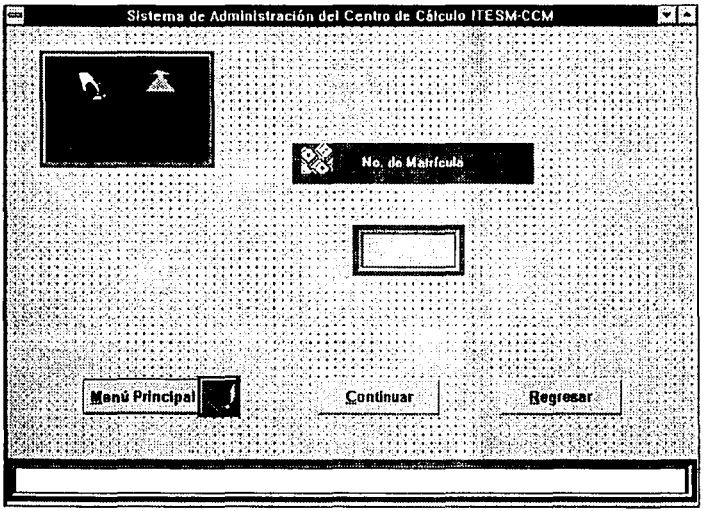
IV.1.7.4 Consultas de usuario



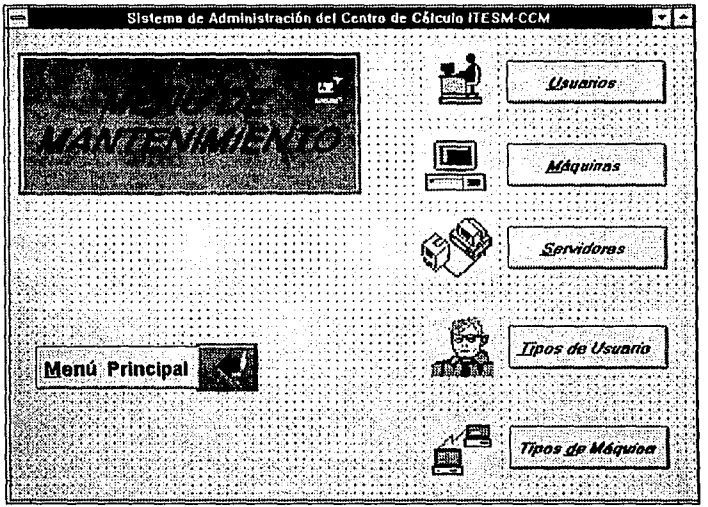
IV.1.7.5 Entradas de usuario



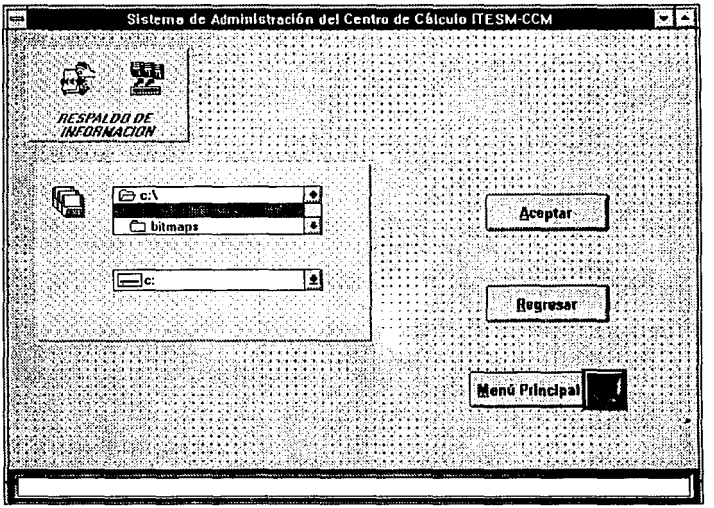
IV.7.6 Mantenimiento de máquinas



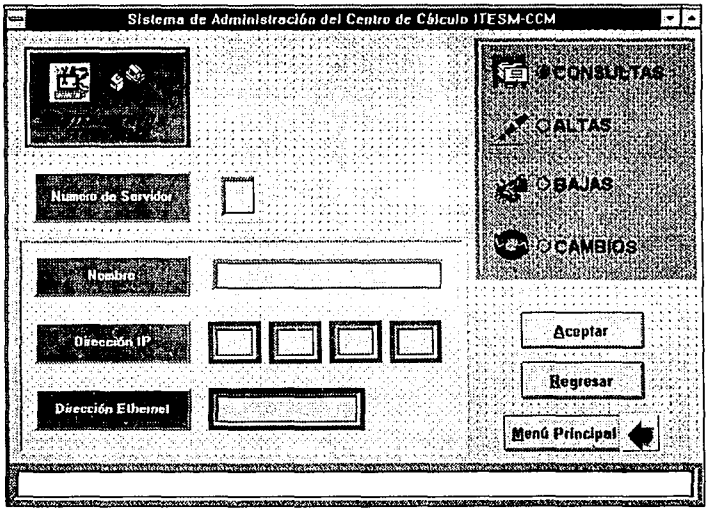
IV.1.7.7 Salidas de apartado



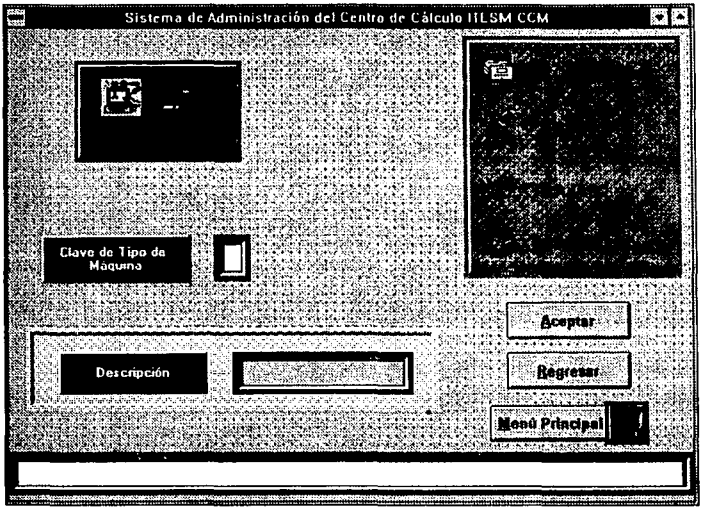
IV.1.7.8 Menú de mantenimiento



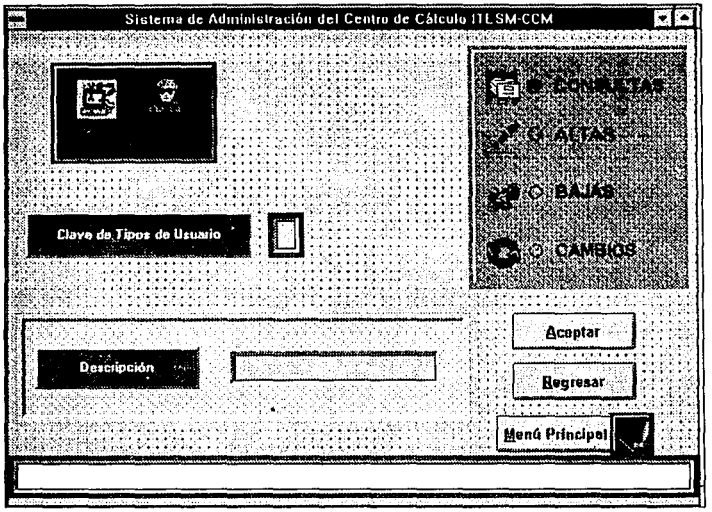
IV.1.7.9 Respaldo



IV.1.7.10 Consulta de mantenimiento a servidores



IV.1.7.10 Mantenimiento de tipos de máquina, consultas



IV.1.7.11 Mantenimiento a tipos de usuarios, consultas

IV.1.8 OPERACION EN RED.

Durante las etapas primeras del desarrollo del sistema, la operación en red, fue uno de los factores que más peso tuvieron en la toma de decisiones, ya que se trata de uno de los puntos álgidos para el desarrollo de cualquier sistema y adicionalmente, se presentó como uno de los puntos principales en los requerimientos de usuario.

Decididamente, se dedicó un importante esfuerzo a la confiabilidad del sistema en la corrida bajo un ambiente de red.

Para comprender cuales son los requerimientos de usuario, es importante observar la forma en que se encuentra el ambiente operativo del sistema, que es la red misma del campus. A continuación se presenta un bosquejo de la topología general de dicha red.

El primer punto a considerar, es la topología física de la red. El campus cuenta con un **back bone** (cable principal) que se encuentra cableado físicamente en fibra óptica, lo que permite una enorme confiabilidad en los enlaces además de presentar un soporte en velocidad muy alto.

El cableado del back bone, está distribuido siguiendo la posición de los edificios, que es una espiral, por lo que comúnmente se conoce a la red de fibra óptica como el anillo

de fibra óptica, sin que la denominación de anillo, tenga relación con ninguna topología de red.

Figura IV.1.8.1 Distribución de edificios del ITESM-CCM

Las transmisiones en general, se llevan a cabo en el anillo de fibra óptica al cual se conectan diversos ruteadores marca CISCO, que se encargan de interfasar distintos modos de cableado y distintos protocolos.

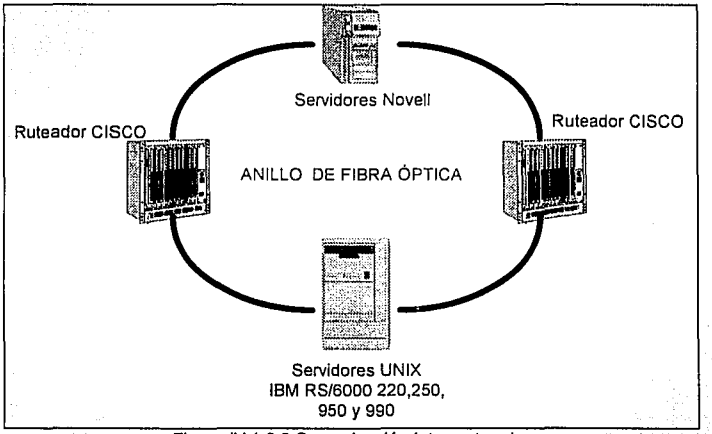


Figura IV.1.8.2 Comunicación Interprotocolo.

Los equipos que se encuentran en el campus, se dividen en tres grandes rubros, que son a saber:

- Equipo Unix
- Equipo Apple
- Equipo PC

Los equipos Unix, se conforma con servidores Risc RS/6000 de IBM, los cuales se encuentran operando bajo la versión de UNIX de IBM, que es AIX y que tiene como plataforma el protocolo de comunicaciones TCP/IP. Se cuenta con cuatro servidores de distintos modelos.

Unix también se emplea para dar salida al campus a la red mundial InterNet, que trabaja en TCP/IP. La comunicación a dicha red, se elabora mediante un enlace de RDI (Red Digital de servicios Integrados) que ofrece Telmex en México, con el campus Estado de México. El campus Estado de México, se comunica vía satélite, con el campus Monterrey y éste a su vez, con una estación ubicada en Dallas, Texas perteneciente a la compañía telefónica *Sprint*.

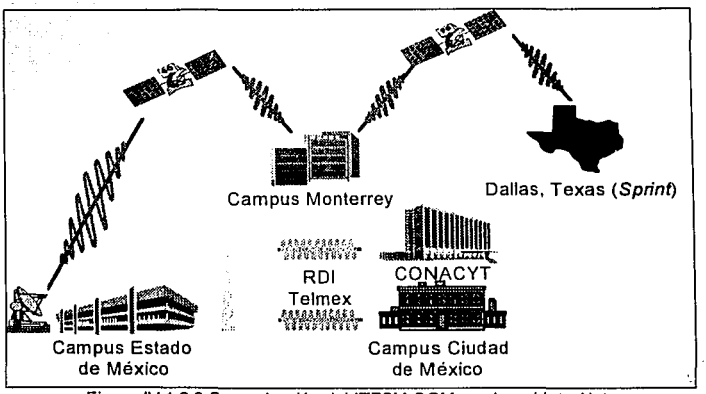


Figura IV.1.8.3 Comunicación del ITESM-CCM con la red InterNet.

El equipo Apple, está conformado por toda la gama de equipos de la marca Macintosh (LC, LCII, LCIII, IIvi, IIvx, Quadra 650, Quadra 800, Centris 610, Centris 650, Power PC 6100/66, 7100/60 y 8100) de los cuales todos se encuentran conectados a red sin que se disponga hasta el momento, de un servidor de aplicaciones de Mac, por lo que el acceso a red, únicamente se refiere al servicio de impresión.

Mediante el cableado de red, los equipos Mac, se comunican a la impresora que deseen, e imprimen en ella para ahorro de recursos. El protocolo de comunicación que se emplea, es el AppleTalk.

El equipo PC, se encuentra dispuesto en red Ethernet, conectado a servidores Novell manejando el protocolo IPX, el cual es nativo de Novell y permite la disposición de servidores de aplicaciones Novell (Windows, Office, etc.). Adicionalmente se encuentran conectadas a colas de Novell, las impresoras que dan servicios al C.E.C.

La interconexión de los distintos protocolos, se realiza en los ruteadores CISCO que tienen la posibilidad de recibir señales de los distintos protocolos y hacer el salto de uno a otro.

Por supuesto, la parte medular de la red, se encuentra en los ruteadores que son los que permiten la compartición de recursos entre máquinas de distintos protocolos así como la interconexión de todo el equipo en el campus.

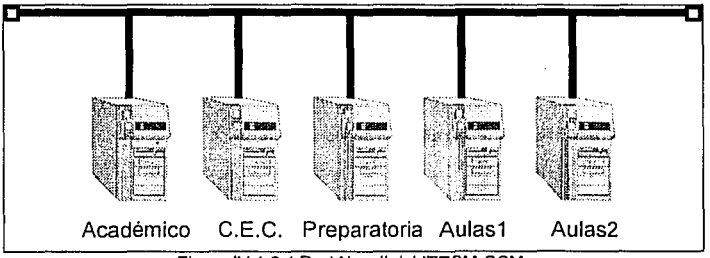


Figura IV.1.8.4 Red Novell del ITESM-CCM

La solicitud básica del sistema de apartados, fue la de operar correctamente en red, lo que implicaba la decisión de cual de los protocolos se debía emplear.

En un principio, la privacía en los sistemas de Mac, y la poca disponibilidad de desarrolladores de equipo Mac, descartó inmediatamente cualquier intento de elaborar el proyecto en Mac.

Asimismo, el alto costo de los desarrolladores de equipo mayor como los servidores RS/6000 aunado a la falta de presupuesto del proyecto, descartó también la posibilidad de desarrollar bajo un ambiente Unix.

Ante la necesidad de desarrollar el sistema en IPX, la tarea fundamental fue la de implementar un sistema de alta confiabilidad en Novell. Por supuesto, como ya se ha

mencionado en los capítulos anteriores, el manejador de registros Btrieve, que es el que Novell emplea para su control interno de archivos, surgió como la mejor opción.

Btrieve saltó a la vista, por muchas circunstancias. Dentro de ellas se encuentra el bajo costo (se disponía de él dado que se contaba con redes Novell), y la facilidad de programación.

Es importante hacer mención que la facilidad de programación es relativa, ya que teniendo en mente que Btrieve es un manejador de registros, no puede compararse con manejadores de bases de datos que cuentan con servicios integrales de mantenimiento a la base de datos. Sin embargo, Btrieve es el mejor en rendimiento en un ambiente de manejadores de registros.

Las implicaciones de tener un manejador de registros y no uno de base de datos, son muy importantes, ya que al tener la obligación de programar en un lenguaje de propósito general como lo fue Pascal.

Por supuesto, la principal ventaja aún independientemente de la velocidad de los acceso, fue el soporte a la operación en red.

Btrieve tiene la peculiaridad de que internamente controla los índices de los archivos (evitando así la molesta y tardada indexación y reindexación) y proporciona una

arquitectura interna de árboles binarios mejorados, lo que permite accesos muy rápidos a los datos.

Adicionalmente, Btrieve controla también internamente, el *semaforeo* de los accesos a las tablas ya sea a nivel de registro o de archivo inteligentemente.

Por obviedad, las restricciones lógicas de una programación en red, se convirtieron en las complicaciones de programar btrieve que se encargaría de la regulación de los accesos en red.

Evidentemente, programar un manejador de registros, representó un enorme problema ya que se deben programar directamente todos los querys (consultas) complejos a las tablas.

Para solidificar la operación en red, se pensaron todos los casos de contingencia posibles que son básicamente una caída en la red (pérdida de la comunicación o daño en el servidor) o una falla de energía eléctrica.

Dado que el punto más importante de operación del sistema de apartados es el mostrador mismo del C.E.C., se pensó que ese era el sitio en el cual se debería garantizar el servicio independientemente de fallas de la red.

Por ello se implementó en la programación, un sistema de *espejo* en el cual se realizan todas las operaciones simultáneamente en el drive (unidad de disco) de red y el disco duro local de la computadora del mostrador del C.E.C.

Con ello se pretendió lograr que si bien se pueden presentar fallas en la red que impidan la operación de los nodos conectados al sistema, el servicio estará garantizado en el C.E.C.

Para ello, el sistema instalado en el C.E.C., tiene los archivos residiendo además de en la red, en el disco duro del nodo, de modo que al detectar la ausencia de los archivos en el drive de red (pérdida de la comunicación), continúa trabajando en disco duro local.

Constantemente entonces, sensará si la red se ha restablecido (en el caso en que retoma la sesión), en caso de detectar nuevamente la comunicación, se realiza una copia de los archivos del disco duro local al drive de red con lo que las nuevas sesiones de las máquinas remotas, tendrán siempre una imagen consistente del sistema.

Es importante hacer notar que si bien ésta metodología se sigue para la instalación de la máquina del mostrador, la misma filosofía se siguió para las máquinas remotas, donde el programa detecta la ausencia de los archivos en un disco duro local (que

puede incluso no tener) por lo que trabajará únicamente el drive de red y se comporta como una máquina esclava y dependiente de la comunicación. Esto permite que se tenga la suficiente seguridad en la máquina remota que no tendrá en la red, privilegios de hacer ningún cambio que desee.

IV.2 INTEGRACION DEL SISTEMA

Una **estrategia de prueba de software** integra las técnicas de diseño de casos de prueba en pasos bien planificados con el fin de construir correctamente el software.

Una estrategia de prueba debe planificar la prueba, diseñar casos de prueba y recolectar y evaluar los resultados. Todas las estrategias tienen las siguientes características generales:

- La prueba comienza en el nivel de módulo y trabaja "hacia afuera" hacia la integración de todo el sistema.
- En diferentes etapas se utilizan distintas técnicas de prueba.
- La prueba la lleva a cabo el que desarrolla el software y a veces se hace necesario un grupo de prueba independiente (sobre todo en proyectos grandes).
- La prueba y la depuración son actividades diferentes, pero la depuración puede entrar en cualquier estrategia de prueba.

La prueba de software es un elemento de un tema más amplio que a menudo se refiere como **verificación y validación**:

- **La verificación** se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica.
- **La validación** se refiere a un conjunto diferente de actividades que aseguran que el software construido se ajusta a los requerimientos del cliente.

El que desarrolla el software es responsable de probar las unidades individuales (módulos) del programa, asegurándose de que cada una lleva a cabo la función para la que fue diseñada. En muchos casos también se encargara de **la prueba de integración** - el paso de prueba que lleva a la construcción y prueba de la estructura total del sistema.

Sólo una vez que la arquitectura del software esté completa entra en juego un grupo independiente de prueba. El papel del grupo independiente de prueba es eliminar los problemas inherentes asociados con el hecho de permitir al constructor que pruebe lo que ha construido.

Una prueba independiente elimina el conflicto de intereses que de otro modo estará presente.

ESTRATEGIA DE PRUEBA DE SOFTWARE

La prueba, en el contexto de la ingeniería de software, consiste en cuatro pasos:

1. Prueba de unidad
2. Prueba de integración
3. Prueba de validación
4. Prueba del sistema

Como se muestra en la figura IV.2.1.

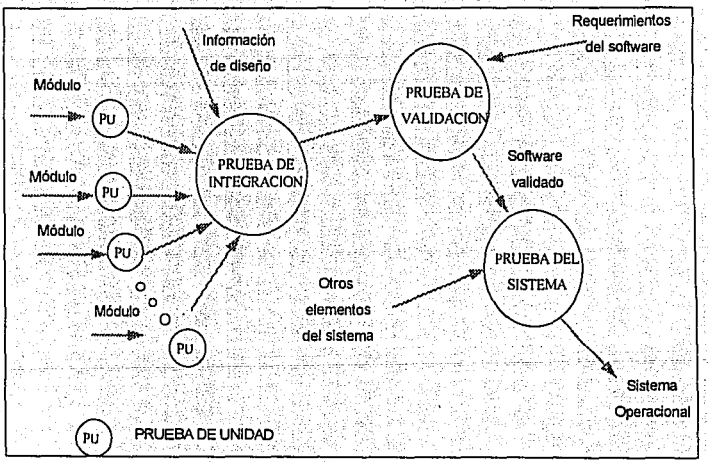


Figura IV.2.1 Pasos de prueba de software

LA PRUEBA DE LA UNIDAD

Esta prueba se centra en la menor unidad del diseño del software, es decir el módulo, en las implementaciones en código fuente. Realiza un uso intensivo de ejercicios de caminos especificos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores dentro del módulo.

Esta prueba se llevó a cabo en paralelo con otras pruebas del módulo, es decir mientras unos verificaban los módulos de apartados y mantenimiento otros probaban el módulo de consultas y salida del sistema.

Posteriormente se integraron los módulos para formar el paquete de software completo.

PRUEBAS PARA CADA MODULO DEL SISTEMA DE APARTADOS

El sistema de apartados esta compuesto por los módulos que se muestran en la figura IV.2.2.

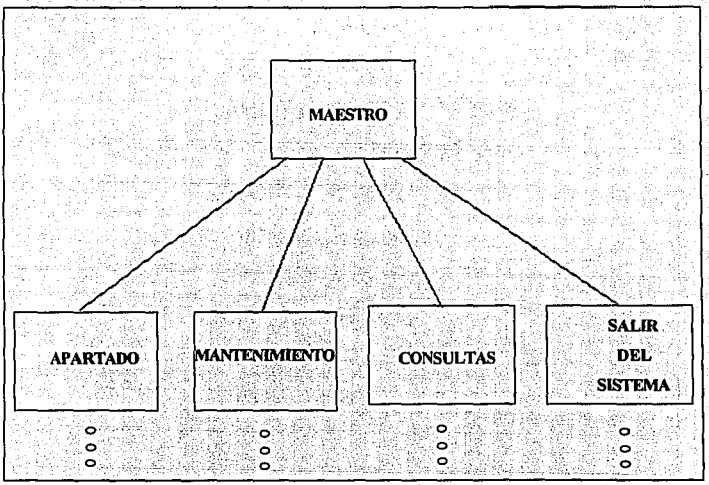


Figura IV.2.2. Módulos del sistema de apartados

El Modulo central o principal permite tener acceso al sistema mediante claves secretas de acceso (passwords) y una vez dentro, el usuario puede llamar a los módulos de apartado, mantenimiento, consultas y salir del sistema

Se probó que cada uno de los menúes de opciones y de las pantallas de captura cumpliera con los puntos que se muestran en la figura IV.2.3.

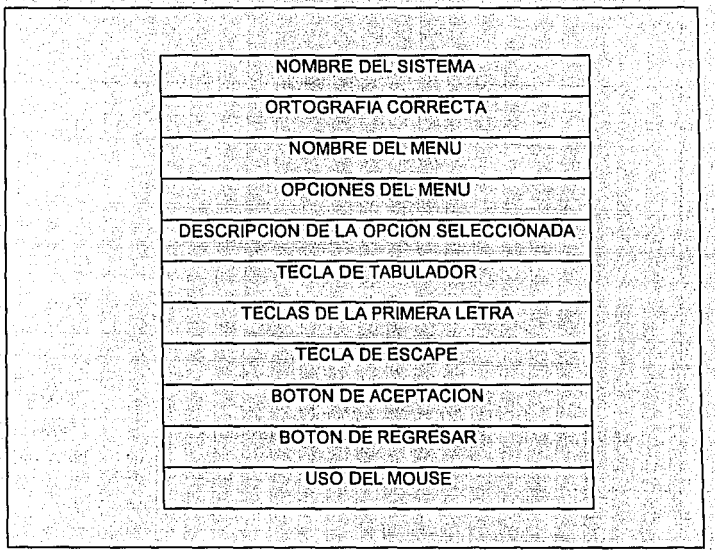


Figura IV.2.3. Pruebas que se aplicaron a los menúes

- **El Nombre del Sistema** siempre aparece en la parte superior de todas las pantallas como "Sistema de Administración del Centro de Cálculo ITESM-CCM".
- **Cumplir con Ortografía correcta,** todos los mensajes y pantallas.

- **Nombre del menú.** Cuando se selecciona un menú se mantiene el nombre de este en la parte superior izquierda de la pantalla y centrado con el fin de saber en todo momento en la opción en que nos encontramos.
- **Opciones del menú.** Se muestran todas las opciones que comprenden al menú seleccionado.
- **Descripción de la opción seleccionada.** Cuando el cursor esté posicionado en una opción deberá aparecer un descripción en la parte inferior de la pantalla que nos indica la función de dicha opción.
- **Tecla de tabulador.** Se podrá hacer uso de la tecla de tabulador para moverse a través de todos los botones, opciones o campos que se desplieguen en la pantalla
- **Tecla de entrar.** Durante la selección de una opción en los menúes se podrá hacer uso de esta tecla para activar un botón de aceptación, a un botón de cancelación o una selección,
- **Teclas de la primera letra.** Se podrá seleccionar una opción mediante la primera letra del nombre de la opción.
- **Botón de aceptación.** Se podrá hacer uso de un botón para indicar que estamos de acuerdo con la selección que se escogió o con los datos introducidos que nos solicite alguna de las opciones y se ejecuta el módulo del programa que le corresponde.
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

- **Tecla de Escape.** Cuando aparece un mensaje se podrá eliminar de la pantalla con la tecla de escape.
- **Uso del mouse.** Se podrá hacer uso del mouse oprimiendo el botón izquierdo con el fin de activar a un botón de aceptación, o un botón de cancelación o una selección, o para ubicarse en un campo en específico para introducir información.

PRUEBA DEL MODULO DE APARTADO

Este módulo está formado por los submódulos que se muestran en la Figura IV.2.4.

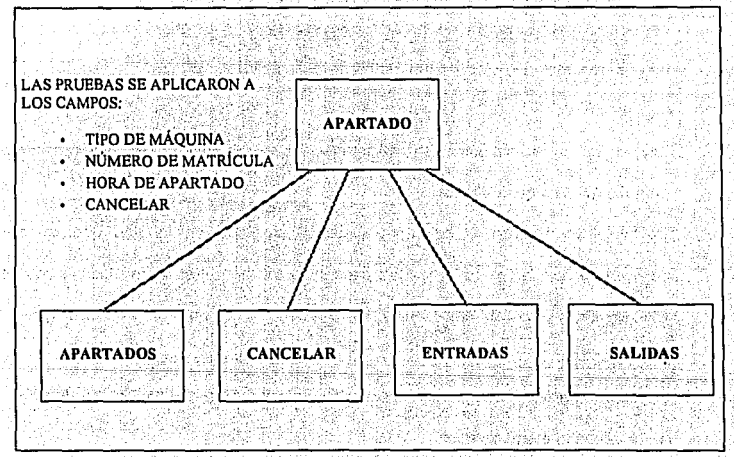


Figura IV.2.4 Prueba del módulo de apartado

PRUEBAS DEL SUBMODULO DE APARTADOS DEL MODULO DE APARTADO

En este submódulo se podrán hacer apartados de máquina en una hora determinada para los usuarios. Podrán ingresarse usuarios hasta que activemos un botón de regresar para volver al modulo principal de apartado. Los campos en los que se aplicaron las pruebas fueron:

- **Número de Matrícula.** Al proporcionar el número de matrícula del alumno el sistema valida si existe si no enviará un mensaje que el usuario no está dado de alta. Si el usuario ya tiene apartada una hora se enviará un mensaje de que el alumno ya tiene una hora apartada o que en ese momento se encuentra utilizándola.
- **Tipo de máquina.** Si el alumno tiene derecho a una hora, se podrá seleccionar el tipo de máquina que se desea
- **Hora de apartado.** Se selecciona de una ventana que muestra el catálogo de horas, desplazándose con el mouse. El sistema verifica si está desocupada el tipo de máquina en la hora seleccionada. Si existe proporciona el número de máquina en caso contrario se indica con un mensaje.

PRUEBAS DE LA OPCION CANCELAR DEL MODULO DE APARTADO

Si un usuario desea cancelar una hora el sistema solicita los siguientes datos:

- **Número de Matrícula.** Al proporcionar el número de matrícula del alumno el sistema valida si existe, si no enviará un mensaje que el usuario no está dado de alta. Si el usuario no tiene apartada una hora se enviará un mensaje que lo indica.
- **Botón de cancelar.** Se podrá activar un botón de "No" en caso de que se deseara abortar la cancelación del apartado.
- **Botón de aceptar.** Se podrá activar un botón de "SI", si se confirma la cancelación.

PRUEBAS DE LA OPCION ENTRADAS DEL MODULO DE APARTADO

Cuando el alumno ingresa al Centro de Cálculo deberá de registrar su entrada, el sistema solicita la información para el siguiente campo:

- **Matrícula.** Al proporcionar el número de matrícula del alumno, el sistema valida si tiene una hora, si no enviará un mensaje que el usuario no tiene hora de apartado. Se envía un mensaje del número de máquina que le fue asignada.

PRUEBAS DE LA OPCION SALIDAS DEL MODULO DE APARTADO

Con esta opción el sistema da salida automática cuando un usuario está haciendo uso de su hora. El sistema solicita los campos:

- **Matrícula.** Al proporcionar el número de matrícula del alumno el sistema valida si tiene una hora.

PRUEBA DEL MODULO DEL CATALOGO DE USUARIOS

Este módulo está formado por los submódulos que se muestran en la Figura IV.2.5.

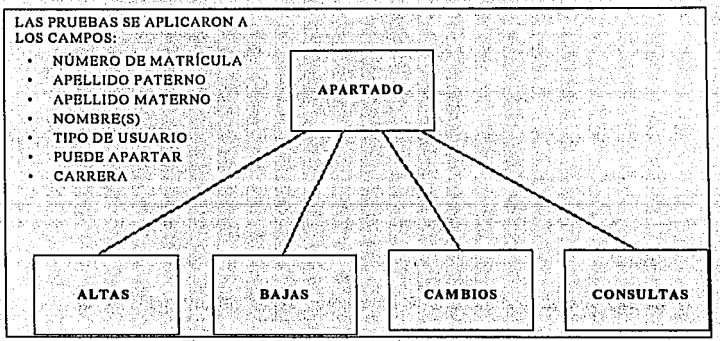


Figura IV.2.5 Prueba del módulo de usuarios

PRUEBAS DE LA OPCION ALTAS DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE USUARIOS

En esta opción se solicitará cada uno de los campos que formarán los datos que identifican al usuario, se probó que cada uno de los campos y botones cumpliera con:

- **Número de matrícula** Se proporciona el número de matrícula y el sistema validará si ya existe. Si ya existe el sistema lo indicara mediante un mensaje
- **Apellido paterno.** EL sistema asegura que el operador no deje en blanco el apellido paterno el cual es de máximo 20 caracteres.
- **Apellido materno.** EL sistema asegura que el operador no deje en blanco el apellido materno el cual es de máximo 20 caracteres.
- **Nombre(s).** EL sistema asegura que el operador no deje en blanco el nombre el cual también es de máximo 20 caracteres.
- **Tipo de Usuario.** Se proporciona una clave que identifica el tipo de usuario (Por ejemplo: **1**. alumno, **2**. profesor **3**. ex-alumno **4**. administrativo).
- **Puede apartar.** Es un campo que al teclear se determina si tiene derecho a apartar una hora de equipo.
- **Carrera.** Es un campo para teclear una clave de la carrera a la que pertenece el usuario.
- **Aceptar el alta.** Si se hace un Click al botón de "ACEPTAR" para confirmar el alta de los datos del usuario

PRUEBAS DE LA OPCION CAMBIOS DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE USUARIOS

En esta opción se podrán realizar cambios en cada uno de los campos que formarán los datos que identifican al usuario:

- **Matrícula.** Se proporciona el número de matrícula y el sistema validará si realmente existe. Si no existe el sistema lo indicara mediante un mensaje.
- **Apellido paterno.** En este campo se tecleará la modificación. El sistema asegura que el operador no deje en blanco el apellido paterno el cual es de máximo 20 caracteres.
- **Apellido Materno.** En este campo se modificará el apellido materno. El sistema asegura que el operador no deje en blanco el apellido materno el cual es de máximo 20 caracteres.
- **Nombre(s).** Si es el campo a modificar, en ese momento se deberá de modificar. El sistema asegura que el operador no deje en blanco el nombre el cual también es de máximo 20 caracteres.
- **Tipo de Usuario.** Si es el campo a modificar, en ese momento deberá modificarse. Se proporciona una clave que identifica el tipo de usuario (Por ejemplo: **1.** alumno, **2.** profesor **3.** ex-alumno **4.** administrativo).
- **Puede apartar.** Si es el campo a modificar, en ese momento deberá modificarse. Es un campo que al teclear se determina si tiene derecho a apartar una hora de equipo.

- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

PRUEBAS DE LA OPCION BAJAS DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE USUARIOS

En esta opción se puede dar de baja un usuario, se probó que cada uno de los campos y botones cumpliera con:

- **Matrícula.** Se proporciona el número de matrícula y el sistema validará si ya existe. Si no existe el sistema lo indicara mediante un mensaje
- **Aceptar la baja.** Si se hace un Click al botón de aceptar para confirmar la baja del usuario
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

- **Carrera.** Si es el campo a modificar, en ese momento deberá modificarse. Es un campo para teclear una clave de la carrera a la que pertenece el usuario
- **Aceptar los cambios.** Si se hace un Click al botón de "ACEPTAR" para confirmar la baja del usuario.
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

PRUEBAS DE LA OPCION CONSULTAS DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE USUARIOS

En esta opción se podrán consultar cada uno de los campos que formarán los datos que identifican al usuario, se probó que cada uno de los campos y botones cumpliera con:

- **Matrícula.** Se proporciona el número de matrícula y el sistema validará si existe. Si no existe el sistema lo indicara mediante un mensaje
- **Aceptar.** Si se hace un Click al botón de "ACEPTAR" para confirmar la consulta del usuario.
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

PRUEBA DEL MODULO DEL CATALOGO DE MAQUINAS

Este módulo está formado por los submódulos que se muestran en la Figura IV.2.6. .

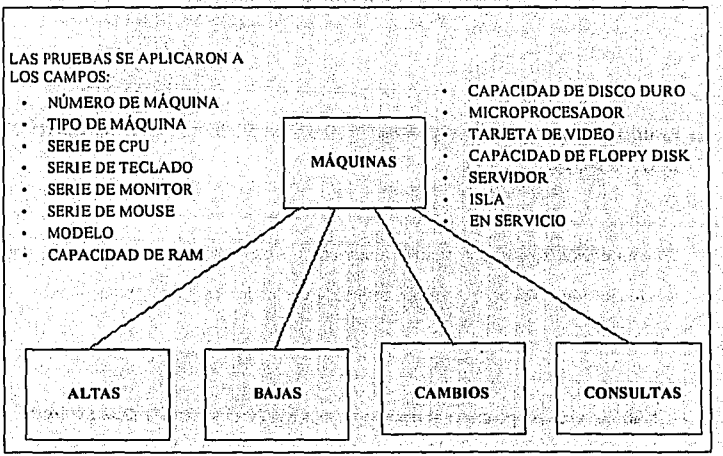


Figura IV.2.6 Prueba del módulo de máquinas.

PRUEBAS DE LAS OPCIONES DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE MAQUINAS.

En la opción de altas se solicitará cada uno de los campos que formarán los datos que identifican las características de las máquinas. En el caso de las bajas, cambios y consultas se mostró la información correspondiente al tipo de máquina. En el caso de cambios se permitió modificar los datos. Se probó que cada uno de los campos y botones cumpliera:

- **Número de máquina** Se deberá de proporcionar el número de máquina, la cual constituye su clave. En la opción de altas el sistema validará si ya existe y si existiera el sistema lo indicara mediante un mensaje. En el caso de las bajas cambios y consultas deberá de existir este número, también se indicará mediante un mensaje en caso de que no estuviera dada de alta. En la opción de cambios no podrá modificarse este campo.
- **Tipo de máquina.** EL sistema asegura que el operador no deje de proporcionar el tipo de máquina (**1.** MAC, **2.** IBM y **3.** PC). En la opción de cambios deberá de permitirse su modificación. En las opciones de bajas y consultas solo deberá de mostrar la información contenida.
- **Número de serie del CPU.** En la opción de altas se deberá de teclear un número de serie correspondiente a ésta máquina de máximo 15 caracteres. En la opción de

cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.

- **Número de serie del teclado.** En la opción de altas se deberá de teclear un número de serie correspondiente al teclado de ésta máquina, de máximo 15 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Número de serie del monitor.** En la opción de altas se deberá de teclear un número de serie correspondiente al monitor de ésta máquina de, máximo 15 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Número de serie del mouse.** En la opción de altas se deberá de teclear un número de serie correspondiente al mouse de ésta máquina, de máximo 15 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Modelo.** En la opción de altas se deberá de teclear el modelo de ésta máquina, de máximo 15 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Memoria RAM.** En la opción de altas se deberá de teclear el número correspondiente de cantidad de memoria RAM del equipo en Megabytes, para ello se cuenta con un campo de máximo 3 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.

- **HD.** En la opción de altas se deberá de teclear el número correspondiente de la capacidad del disco duro en Megabytes, en un campo de máximo 4 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Microprocesador.** En la opción de altas se deberá de teclear el tipo de microprocesador correspondiente a ésta máquina, el campo contará con 6 caracteres como máximo. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Video.** En la opción de altas se deberá de teclear el tipo de tarjeta de video de ésta máquina, el campo contará con máximo 5 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **FD.** En la opción de altas sólo podrá teclearse el número correspondiente de la capacidad del manejador del disco flexible en caso de que dicho equipo contara con el mismo, y deberá proporcionarse en kilobytes en un campo de 5 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Servidor.** En la opción de altas se deberá de teclear el número del servidor al que está conectada ésta máquina, el campo contará con 2 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Isla.** En la opción de altas se deberá de teclear el número de la isla (grupo de computadoras) en que está ubicado físicamente dicho equipo, el campo contará con

2 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.

- **En Servicio.** El sistema asegura que el operador no deje de proporcionar esta información. En la opción de altas se deberá de teclear si el equipo está en servicio o no. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Aceptar.** Si se hace un Click al botón de "ACEPTAR" para confirmar el alta, baja o cambio de los datos de la máquina.
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

PRUEBA DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE SERVIDORES.

Este módulo está formado por los submódulos que se muestran en la Figura IV.2.7.

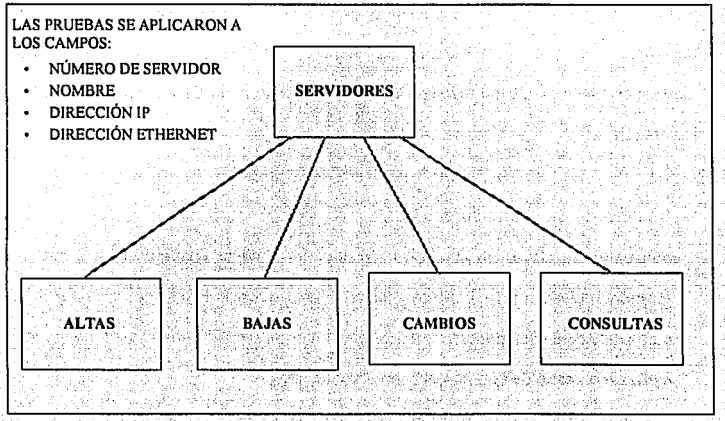


Figura IV.2.7 Prueba del módulo de servidores.

PRUEBAS DE LAS OPCIONES DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE SERVIDORES

En la opción de altas se solicitará cada uno de los campos que formarán los datos que identifican al servidor en el ambiente de red. En la opción de cambios podrán modificarse estos datos. En el caso de las bajas y consultas se mostró la información correspondiente al tipo de servidor. Se probó que cada uno de los campos y botones cumpliera:

- **Número de servidor.** Se deberá de proporcionar el número de servidor, la cual constituye su clave. En la opción de altas el sistema validará si ya existe y si existiera el sistema lo indicara mediante un mensaje. En el caso de las bajas cambios y consultas deberá de existir este número, también se indicará mediante un mensaje en caso de que no estuviera dado de alta.
- **Nombre.** En la opción de altas se deberá de teclear el nombre con el que fue bautizado el servidor, el campo contará con 20 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Dirección.** En la opción de altas se deberá de teclear el número de la dirección IP del servidor, necesaria para el protocolo de comunicaciones, el campo contará con 12 caracteres en grupos de 3 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.

- **Dirección Ethernet.** En la opción de altas se deberá de teclear el número de la dirección Ethernet del servidor, necesaria para compartir información a nivel mundial, el campo contará con 15 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará
- **Aceptar.** Si se hace un Click al botón de "ACEPTAR" para confirmar el alta, baja o cambio de los datos del servidor.
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

PRUEBA DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE TIPOS DE USUARIOS.

Este módulo está formado por los submódulos que se muestran en la Figura IV.2.8.

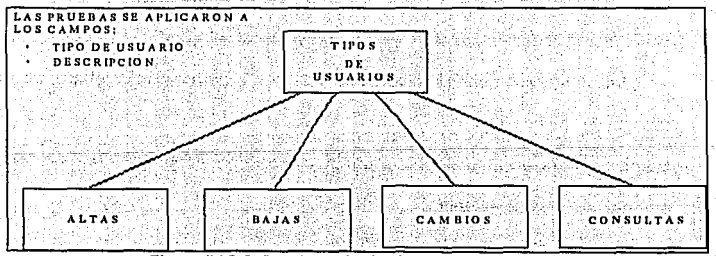


Figura IV.2.8. Prueba del módulo tipos de usuarios.

PRUEBAS DE LAS OPCIONES DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE TIPOS DE USUARIOS.

En la opción de altas se solicitará la información que formarán los datos que identifican a los tipos de usuarios. En la opción de cambios podrán modificarse estos datos. En el caso de las bajas y consultas se muestra la descripción del tipo de usuario. Se probó que cada uno de los campos y botones cumpliera con:

- **Clave tipo.** Se deberá de proporcionar el tipo de usuario (alumno, ex-alumno, visitante, etc.), la cual constituye su clave. En la opción de las altas el sistema validará si ya existe y si existiera el sistema lo indicara mediante un mensaje. En el caso de las bajas cambios y consultas deberá de existir este número, también se indicará mediante un mensaje en caso de que no estuviera dado de alta. El campo contará con 1 caracter.
- **Descripción.** En la opción de altas se deberá de teclear una breve descripción del tipo de usuario, el campo contará con 20 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Aceptar.** Si se hace un Click al botón de "ACEPTAR" para confirmar el alta, baja o cambio del tipo de usuario.
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

PRUEBA DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE TIPOS DE MAQUINAS.

Este módulo está formado por los submódulos que se muestran en la Figura IV.2.9.

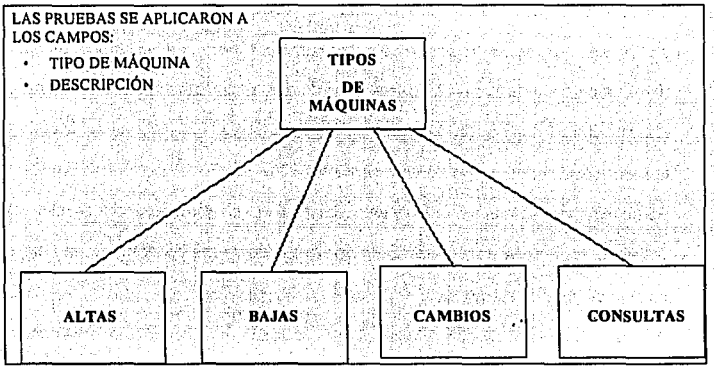


Figura IV.2.9. Prueba del módulo tipos de máquinas.

PRUEBAS DE LAS OPCIONES DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE TIPOS DE MAQUINAS.

En la opción de altas se solicitará la información que formarán los datos que identifican a los tipos de máquinas. En la opción de cambios podrán modificarse estos datos. En

el caso de las bajas y consultas se muestra la descripción del tipo de máquina. Se probó que cada uno de los campos y botones cumpliera con:

- **Clave tipo.** Se deberá de proporcionar el tipo de máquina (IBM, PC, MAC,... etc.) , la cual constituye su clave. En la opción de altas el sistema validará si ya existe y si existiera el sistema lo indicara mediante un mensaje. En el caso de las bajas cambios y consultas deberá de existir este número, también se indicará mediante un mensaje en caso de que no estuviera dado de alta. El campo contará con 1 caracter.
- **Descripción.** En la opción de altas se deberá de teclear una breve descripción del tipo de máquina, el campo contará con 20 caracteres. En la opción de cambios se podrá modificar esta información y en la opción de bajas y consultas se desplegará.
- **Aceptar.** Si se hace un Click al botón de "ACEPTAR" para confirmar el alta, baja o cambio del tipo de máquina.
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

PRUEBA DEL MODULO DE MANTENIMIENTO DEL CATALOGO DE TIPOS DE MAQUINAS.

Este módulo está formado por los submódulos que se muestran en la Figura IV.2.9.

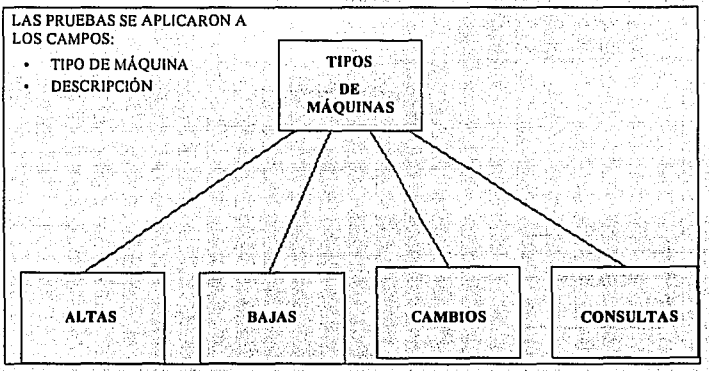


Figura IV.2.9. Prueba del módulo tipos de máquinas.

PRUEBA DEL MODULO DE CONSULTAS.

Este módulo está formado por los submódulos que se muestran en la Figura IV.2.10.

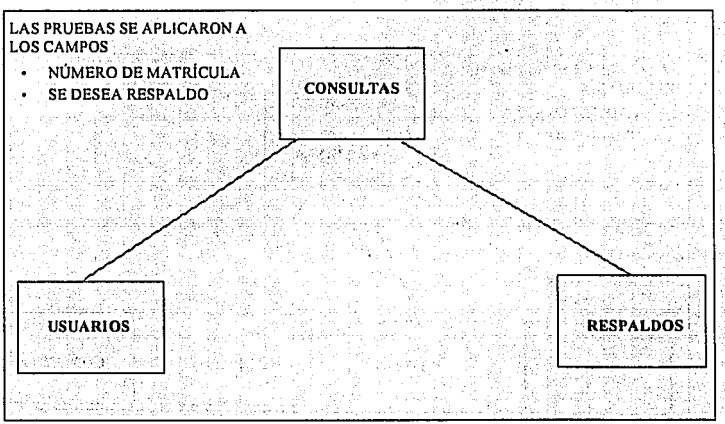


Figura IV.2.10. Prueba del módulo de consultas.

PRUEBAS DE LAS OPCIONES DEL MODULO DE CONSULTAS.

En la opción de consultas se solicitará información para consultar si un alumno tiene apartada una máquina a una hora determinada. Podrá respaldar información en disco flexible. Se probó que cada uno de los campos y botones cumpliera con:

- **Número de matrícula.** Se deberá de proporcionar el número de matrícula y el sistema validará si realmente existe. Si no existe el sistema lo indicara mediante un mensaje. Si el alumno tiene un apartado se desplegará un mensaje indicando la máquina asignada y el número de la misma. En el caso contrario se despliega un mensaje.
- **Se desea respaldo.** Se deberá de escoger la opción si se continua el respaldo de la información en disco flexible.
- **Aceptar.** Si se hace un Click al botón de "ACEPTAR" para confirmar la acción de consulta de apartado de un usuario o respaldo de la información.
- **Botón de regresar.** Se podrá hacer uso de un botón para abandonar la selección y regresarnos al menú anterior.

LA PRUEBA DE INTEGRACION

Durante esta prueba las actividades se centraron en el diseño y construcción de la arquitectura del software. Se verifica y se construye el programa como un todo. Durante la integración, las técnicas que más prevalecen son las de diseño de casos de prueba de la caja negra, aunque se pueden llevar a cabo algunas pruebas de la caja blanca con el fin de asegurar que se cubran los principales caminos de control.

- **Prueba de caja negra.** Es una prueba que asegura que todas las "piezas encajan" o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada; conociendo el funcionamiento del producto. Se llevan a cabo sobre la interface del software. Se pretende demostrar que las funciones del software son operativas, que la entrada se acepta en forma adecuada y que se produce una salida correcta , así como la integridad de la información externa se mantiene (p.ej., archivos de datos). Se examinan algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software.

- **Prueba de la caja blanca.** Es una prueba que demuestra que cada función es completamente operativa, conociendo la función específica para la que fue diseñado el producto. Se basa en el minucioso exámen de los detalles

procedurales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejercitan conjuntos específicos de condiciones y/o bucles.

PRUEBA DE INTEGRACION DEL SISTEMA DE APARTADOS

Se utilizó esta técnica para construir la estructura del programa y para realizar pruebas para detectar errores asociados con la interacción entre los módulos. Se tomaron los módulos probados en unidad y nos aseguramos que la estructura del programa estuviera de acuerdo con lo que dictaba el diseño.

Se utilizó la **integración incremental** en el cual el programa se construye y se prueban los módulos en los que los errores son más fáciles de aislar y de corregir con el fin de probar completamente todas las interfaces y aplicar una aproximación de prueba sistemática.

Existen dos estrategias de integración incremental:

- **Integración descendente**
- **Integración ascendente**

Empleamos la **integración descendente de forma primero-en-profundidad.** Integramos los módulos moviéndonos hacia abajo por la jerarquía de control, comenzando con el módulo de control principal (módulo maestro). Los módulos subordinados al módulo de control principal se van incorporando en la estructura de forma primero-en-profundidad.

Refiriéndonos a la Figura IV.2.11: se eligió el camino a mano Izquierda, se integraron los módulos M1, M2, y M3. A continuación será integrado M4. Acto seguido se construyen los caminos de control central y derecho.

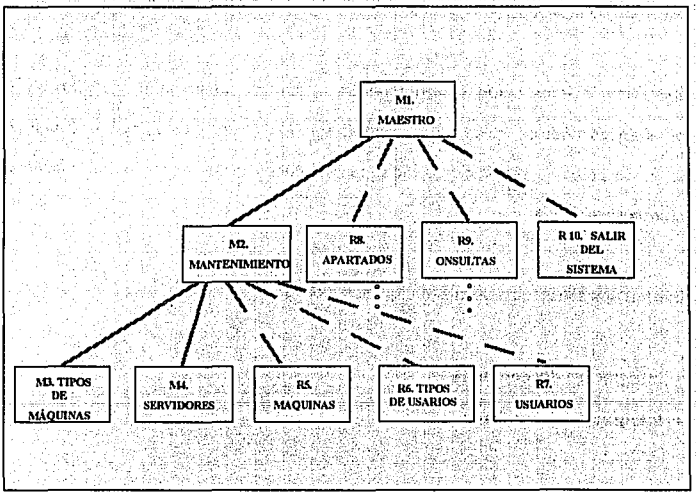


Figura IV.2.11. Integración descendente de forma primero en profundidad.

Llevamos a cabo el proceso de integración en cinco pasos:

Se usó el módulo principal como conductor de la prueba, disponiendo de los resguardos para todos los módulos directamente subordinados al módulo principal (los resguardos sirven para reemplazar módulos que están subordinados a el módulo a ser probado; un resguardo o "subprograma mudo" usa la interface del módulo subordinado, lleva a cabo la mínima manipulación de datos e imprime una verificación de la entrada y vuelve).

1. Se usa el módulo de control principal como conductor de la prueba, disponiendo resguardos para todos los módulos directamente subordinados al módulo de control principal.
2. Se fueron sustituyendo los resguardos subordinados uno a uno por los módulos reales
3. Se llevaron a cabo pruebas de caja negra cada vez que integramos un nuevo módulo.
4. Después de la pruebas reemplazamos otro resguardo con el módulo real.
5. Realizamos la prueba de regresión (o sea, todas las pruebas anteriores) para asegurarnos de que no se hayan introducido nuevos errores.

El proceso continuó desde el paso 2 hasta construir la estructura del programa entero. En cada reemplazamiento se llevaron a cabo pruebas para verificar la interface

La principal desventaja de la aproximación descendente es la necesidad de resguardos y las dificultades de prueba que pueden estar asociadas con ellos.

ESPECIFICACION DE LA PRUEBA DE INTEGRACION

Alcance de la prueba. Se probaron características tales como:

Fácil uso por parte del usuario.

Desplegado de ayuda

El fácil acceso utilizando los dispositivos de entrada: el mouse y el teclado.

Además que el número de módulos cumplieran con las especificaciones solicitadas.

Plan de prueba. La estrategia general para la integración se dividió en fases y subfases, dirigidas a características específicas funcionales del software:

Interacción con el usuario

-selección de órdenes

-representación visual

-procesamiento y representación de errores

Manipulación y análisis de datos

Procesamiento y generación de información visual

-Reportes y estadísticas

Estructura y contenido de la base de datos

En cada fase se siguieron los criterios con sus correspondientes pruebas:

- **Integridad de la interface.** Se probaron las interfaces internas y externas a medida que se incorporaron los módulos
- **Validez funcional.** Se llevaron a cabo pruebas diseñadas para descubrir errores funcionales.
- **Contención de información.** Pruebas para descubrir errores asociados con las estructuras globales y locales.
- **Rendimiento.** Pruebas para verificar los límites de rendimiento establecidos durante el diseño del software.

Los recursos empleados fueron una PC AT 486 con 240 MB en disco duro disponible, con conexión en red.

Procedimiento de prueba. Describimos el orden de integración y las pruebas correspondientes a cada fase de integración.

LA PRUEBA DE VALIDACION DEL SISTEMA DE APARTADOS

En una prueba de validación se deben comprobar los criterios de validación establecidos durante la fase de definición del sistema. Proporciona una seguridad final de que el software satisface todos los requerimientos funcionales y de rendimiento. Se usan exclusivamente técnicas de prueba de la caja negra.

Después de encontrar y corregir los errores de interfaces, comenzamos la prueba de validación. Con el fin de descubrir errores cuando el usuario utiliza el sistema, se llevaron a cabo las prueba alfa y beta, Figura IV.2.12.

- ***PRUEBA ALFA. En el lugar del desarrollo se invitó a operadores para probar el sistema, en presencia de los desarrolladores.***
- ***PRUEBA BETA. La prueba se realizó con operadores en el lugar físico donde estaría funcionando el sistema, ningún desarrollador estuvo presente.***

PRUEBA ALFA

PRUEBA BETA

Figura IV.2.12. Pruebas alfa y beta como pruebas de validación.

Prueba alfa. Para llevar a cabo la prueba alfa se invitó a un usuario en el lugar del desarrollo y en presencia del desarrollador, éste registró errores y problemas de uso. Es decir todo esto se llevó en un entorno controlado, en el mismo Instituto Tecnológico en el área de desarrollo. Al principio al utilizar el sistema, el operador tenía cierto temor y no se presentaba seguro. Los mensajes que proporcionaba el sistema algunos fueron modificados porque les faltaba claridad.

La principales deficiencias en esta prueba fue falta de familiaridad en el uso de menúes y el entender el procedimiento para manipular los datos.

En la prueba beta se llevó a cabo en el lugar donde estaría en funcionamiento el sistema, el encargado del desarrollo no estuvo presente

Sobre todo se presentaron problemas tales como la necesidad de una explicación sencilla de los errores y mensajes y además dudas en el instructivo.

PRUEBA DEL SISTEMA

Verifica que cada elemento del programa trabaja adecuadamente alcanzando la funcionalidad y el rendimiento del sistema total.

PRUEBA DEL SISTEMA DE APARTADOS

Entre los tipos de prueba del sistema que llevamos a cabo fueron:

Prueba de recuperación. Con el fin de probar su tolerancia a fallos y el período de tiempo de corrección de fallas Figura IV.2.13.

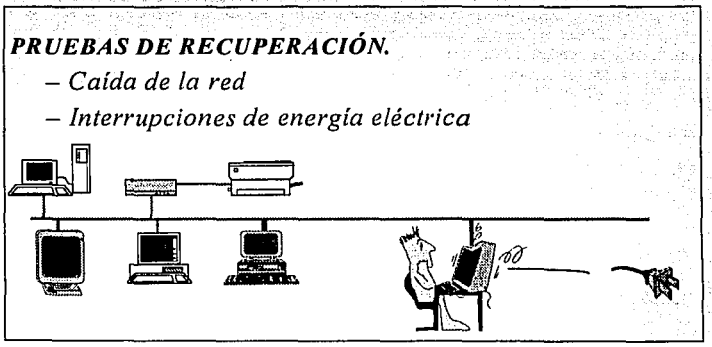


Figura IV.2.13. Pruebas del sistema

Para lo anterior forzamos el fallo del software mediante:

1. Caída de la Red.

2. Fallas de energía eléctrica.

En ambas la transacción en proceso era la que se perdía y con la implementación de un módulo de recuperación para ejecutarlo en éstos casos, se mantenía la integridad del resto de la información. La recuperación requiere la intervención humana, se evaluaron los tiempos medios de recuperación que no excedieron de 1 minuto y sólo se perdía el registro que estaba siendo actualizado en ese momento, determinándose que se encontraban dentro de los límites aceptables.

Prueba de seguridad. El mecanismo que evita que una persona que no tenga acceso al sistema pueda modificar cierta información, se evita con el uso de passwords, inclusive de diferentes niveles de acceso:

- Tiene acceso a todas las opciones del menú principal.
- Tiene acceso al módulo de Mantenimiento incluyendo todos los submenús correspondientes y al módulo de Salir del Sistema.
- Tiene acceso al módulo de Apartados incluyendo todos los submenús correspondientes, al módulo de Consultas y al módulo de Salir del Sistema.
- Tiene acceso sólo al submenú de apartados dentro del módulo de Apartados, al Módulo de Consultas y al Módulo de Salir del Sistema.

Al teclear alguno de los passwords antes descritos, se desplegará el menú correspondiente a cada uno de estos y las opciones a las cuales el usuario tendrá acceso.

El campo de passwords permite la combinación de cualquier caracter ASCII lo que aumenta la combinación de nombres y por lo tanto la seguridad.

Prueba de resistencia y de rendimiento. Entre las pruebas que se sometieron fueron las siguientes:

1. Se realizaron de tal manera que se sometió al sistema a una demanda de recursos en cantidad, y frecuencia anormales por medio del aumento de interrupciones al sistema por medio de 5 operadores en red que demandaban al mismo tiempo recursos al sistema, de los cuales sólo algunas eran de acceso normal al sistema.

2. Se probó casos que produjeran excesivas búsquedas de datos residentes en disco, por medio de localizar a un alumno al mismo tiempo por 5 operadores.

3. Se probó el tiempo de respuesta del sistema para los casos que requerían diferentes tamaños en la tabla de alumnos, desde 1 registro, pasando por 7,000 alumnos hasta 20,000.

4. Todo lo anterior debería poder realizarse respetando la ejecución de los procesos en los tiempos críticos en cada hora del sistema.

- **Del minuto 00 con 00 segundos al minuto 00 con 10 segundos.** Inicialización de variables.
- **Del minuto 00 con 11 segundos al minuto 00 con 31 segundos.** Copia del archivo Aparta.dat de la red al disco duro.
- **Del minuto 02 al 06.** Contadores por tipo de máquina disponibles en el CEC
- **Del minuto 10 al 14.** Contador total de apartados.
- **Del minuto 16 al minuto 49.** Contador total de entradas con y sin apartados.
- **Del minuto 50 al 59.** Da salidas automáticas, es decir, borra información de ocupa y apartados.
- **Del minuto 51 al 54.** Inserción de datos a la tabla por día (Lunes, Martes, Miércoles, Jueves, Viernes, Sábado), para generar el reporte de horas máquina por día.
- **Del minuto 55 al minuto 58.** Inserción de datos a la tabla de estadísticas por día (Estlun, Estmar, Estmie, Estjue, Estvie y Estsab), para generar el reporte de estadísticas del total de apartados, apartados sin entrar y entradas sin apartados por tipo de máquina.

OPERACION DEL SISTEMA

Después de lo anterior se hizo entrega del sistema para su operación en tiempo real, en la que se observó que la inversión de los recursos humanos y económicos se vieron reflejados en la mejora del tiempo de respuesta en la atención de todos los usuarios del Centro de Cálculo del Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Ciudad de México figura IV.12.14.

Figura IV.2.14. Operación del sistema.

IV.3. DOCUMENTACION

IV.3.1. MANUAL TECNICO

COMO ENTRAR AL SISTEMA

1.- Para entrar al sistema de apartados, se deberá escribir desde el prompt del sistema operativo (c:>), la palabra:

MAESTRO(Dará acceso al sistema para trabajar directamente en el disco duro de la máquina)

NOTA: Por no tener acceso a la red, desplegará en una ventana el siguiente mensaje: "NO SE PUDO ABRIR *nombre del archivo*".

nombre del archivo se refiere a las bases de datos en las que tiene que dar acceso, para lo cual será necesario teclear (ENTER) a cada uno de estos mensajes hasta que se despliegue el menú principal.

De la misma manera al indicar en el sistema la salida, aparecerá el siguiente mensaje: "NO SE PUDO CERRAR *nombre del archivo*", para lo cual será necesario teclear

(ENTER) a cada uno de estos mensajes hasta que indique el prompt del sistema operativo.

En caso de estar funcionando la conexión a la red, entonces se deberá escribir la palabra:

APARTADO (La cual hará directamente el enlace en la red para trabajar conjuntamente en el disco duro de la máquina y en la red).

2.- Teclear password correspondiente:

Existen cuatro password diferentes dependiendo del nivel de acceso al sistema de apartado:

1.- ****** Tiene acceso a todas las opciones del menú principal.

2.- ****** Tiene acceso al modulo de mantenimiento incluyendo todos los submenús correspondientes y al módulo de salir del sistema.

3.- ****** Tiene acceso al modulo de apartados incluyendo todos los submenus correspondientes, al módulo de consultas y al módulo de salir del sistema.

4.- ****** Tiene acceso solo a submenú de apartados dentro del módulo de apartados, al módulo de consultas y al módulo de salir del sistema.

Al teclear alguno de los passwords antes descritos, se desplegará el menú correspondiente a cada uno de estos y las opciones a las cuales el usuario tendrá acceso.

HORAS DE PROCESOS CRITICOS

☑ Del minuto 00 con 00 segundos al minuto 00 con 10 segundos.

Proceso a ejecutar: Inicialización de variables.

☑ Del minuto 00 con 11 segundos al minuto 00 con 31 segundos.

Proceso a ejecutar: Copia del archivo Aparta.dat de la red al disco duro.

☑ Del minuto 02 al minuto 06

Proceso a ejecutar: Contadores por tipo de máquina disponibles en el C.E.C.

☑ Del minuto 10 al minuto 14.

Proceso a ejecutar: Contar total de apartados.

☑ Del minuto 16 al minuto 49

Proceso a ejecutar: Busca apartados sin registrar entrada para borrarlos.

☑ Del minuto 47 al minuto 49

Proceso a ejecutar: Contador total de entradas con y sin apartados.

☑ Del minuto 50 al minuto 59

Proceso a ejecutar: Da salidas automáticas, es decir, borra información de ocupa y apartados.

☑ Del minuto 51 al minuto 54

Proceso a ejecutar: Inserción de datos a la tabla por día. (Lunes, Martes, Miércoles, Jueves, Sábado), para generar el reporte de horas/máquina por día.

☑ Del minuto 55 al minuto 58

Proceso a ejecutar: Inserción de datos a la tabla de estadística por día (Estlun, Estmar, Estmie, Estjue, Estvie, Estsab), para generar el reporte de estadísticas del total de apartados, apartados sin entrar, y entradas sin apartados por tipo de máquina.

MODULOS DEL SISTEMA

El sistema de apartados del Centro Electrónico de Cálculo cuenta con los siguientes módulos:

APARTADO	MANTENIMIENTO	CONSULTAS
Apartados	Usuarios	Usuario
Cancelaciones	Máquinas	
Entradas	Servidores	
Salida	Tipos de Usuarios	
	Tipos de máquina	

DESCRIPCION DETALLADA POR CADA MODULO

MODULO DE APARTADO

Apartados:

Proceso:

1.- Pide la matrícula del alumno.

2.- Pide el tipo de máquina por medio de una ventana que contiene el catálogo de tipos de máquina.

3.- Pide la hora para hacer el apartado, en la que presenta una ventana que contiene el catálogo de horas.

De esta manera el apartado quedará automáticamente registrado e indicará en una ventana de mensajes el número de máquina que le fué asignado y regresará a la ventana que pide el número de la matrícula para continuar con los apartados.

ADVERTENCIA:

- En caso de que al teclear el número de la matrícula aparezca el siguiente mensaje: " USUARIO NO DADO DE ALTA!! ", esto significa que el operador del sistema tiene que dar de alta al usuario en el módulo de mantenimiento en el submenú de usuarios.

- En caso de que al teclear el número de la matrícula aparezca el siguiente mensaje: " EL USUARIO ESTA DENTRO DEL C.E.C ", significa que el usuario tiene que esperar hasta que termine su hora para volver a realizar un apartado.

- Si el usuario esta dentro del C.E.C. y desea hacer un apartado para la siguiente hora inmediata, podrá hacerlo únicamente faltando diez minutos para que se termine la hora vigente en uso.

  Si el usuario está dentro del C.E.C. y desea hacer un apartado para otra hora que no sea la inmediata, podrá hacerlo siempre y cuando su hora en uso termine o registre su salida.

- El alumno puede realizar un apartado fuera del C.E.C. a cualquier hora.

Cancelaciones:

Proceso:

1.- Pide la matrícula del alumno.

2.- Se cancelará automáticamente el apartado realizado previamente y regresa al módulo de apartado.

ADVERTENCIA:

- En caso de que al teclear el número de matrícula aparezca el siguiente mensaje: "EL USUARIO NO TIENE APARTADOS", esto significa que el apartado para esa matrícula no fue elaborado.
- Sólo se podrá realizar una cancelación siempre y cuando la hora en que se haga sea menor que la hora registrada en el apartado.

Entradas:

Proceso: Existen dos procesos,

I.- CON APARTADO. Si el alumno realizó el apartado de una máquina previo a la hora de uso.

1.- Pide la matrícula del alumno

Despliega un mensaje indicando el tiempo de retraso con el que entró y el número de la máquina que le fué asignada.

De esta manera, queda registrada la entrada del usuario al C.E.C. y regresa a la ventana en la que pide el número de la matrícula para continuar dando Entradas a usuarios. En caso de querer regresar al módulo de apartado deberá teclear (ESC).

ADVERTENCIA:

- El alumno deberá registrar su entrada al C.E.C. en el transcurso del minuto cero hasta el minuto 15 de la hora en que realizó el apartado, debido a que el sistema internamente borrará a partir del minuto 16 todos los apartados que estén registrados a esa hora y sin entrada.
- En caso de que el alumno tenga un apartado a una hora e intente registrar entrada a otra hora, se desplegara en pantalla una ventana indicando el siguiente mensaje: " EL USUARIO TIENE APARTADO A OTRA HORA ", y no dará acceso hasta que la hora de entrada sea la misma que la hora del apartado. Si por disposición del alumno requiere entrar antes que la hora del apartado que tenía, el operador tendrá que dirigirse al módulo de apartado, al submenú de cancelaciones y cancelar el apartado. Una vez realizado este proceso, podrá registrar la entrada sin problemas.

II.- SIN APARTADO. Si el alumno no realizó el apartado de una máquina.

1.- Pide la matrícula del alumno

2.- Pide el tipo de máquina por medio del catálogo de tipos de máquinas.

Despliega un mensaje indicando el número de la máquina que le fué asignada.

De esta manera queda registrada la entrada del usuario al C.E.C. y regresa a la ventana en la que pide el número de la matrícula para continuar dando entradas a usuarios.

ADVERTENCIA:

- El alumno deberá registrar su entrada en el transcurso del minuto cero hasta el minuto 45 de la hora en que desee accesar al C.E.C , siempre y cuando no tengan ningún apartado o se encuentre dentro de las instalaciones del C.E.C.

La diferencia de registrar entrada con apartado es que el alumno dependerá de la disponibilidad de equipo con que se cuente en el momento en que entre al C.E.C.

Advertencia para ambos procesos:

- En caso de teclear una matrícula errónea o que no este dada de alta en el catálogo de usuarios, aparecerá el siguiente mensaje: "EL USUARIO NO PUEDE ENTRAR O NO HA SALIDO" por lo tanto no registrará entrada a esa matrícula.

- En caso de teclear una matrícula y aparezca el siguiente mensaje: "EL USUARIO ESTA DENTRO DEL C.E.C. " , quiere decir que un usuario no puede registrar entrada al C.E.C. más de una vez.

Salidas:

Proceso:

1.- Pide la matrícula del alumno

De esta forma da salida automáticamente al usuario de las instalaciones del C.E.C.

ADVERTENCIA:

- En caso de registrar salida antes del minuto 45 de la hora en uso, automáticamente queda liberado el equipo que fué utilizado y podrá ser ocupado por otro alumno hasta que se termine esa hora.

- Si el alumno no registra salida, el sistema de apartados libera y da salida automáticamente al minuto 49 de la hora en uso a todas aquellas matrículas que estén registradas en el sistema como entradas a esa hora.

MODULO DE MANTENIMIENTO

Usuarios:

Altas

Proceso

1.- Se pide la información del usuario mediante una pantalla para capturar los siguientes datos:

Matrícula

Apellido Paterno

Apellido Materno

Nombre (s)

Tipo de usuario

Puede apartar?

Carrera

Una vez capturada la información se deberá seleccionar el botón de continuar para actualizar la información y quede registrada en la base de datos de usuarios.

Si el proceso de altas fué exitoso, aparecerá el siguiente mensaje: "ALTA EXITOSA", y regresará a la opción de usuarios.

ADVERTENCIA:

- En caso de haber respondido por error que SI desea dar altas, aparecerá en la pantalla la ventana de captura de datos donde se deberá teclear (ESC), desplegándose el siguiente mensaje: "CAPTURA ABORTADA".

  En este caso, automáticamente regresará a la opción de usuarios del menú.

Bajas

Proceso:

1.- Pide matrícula del alumno

2.- Se despliega el siguiente mensaje: "USUARIO BORRADO", Y REGRESA AUTOMATICAMENTE A LA OPCION USUARIOS.

Cambios:

Proceso

1.- Pide la matrícula del alumno.

2.- Se despliega la pantalla de captura de datos, que permite realizar los cambios necesarios para ser actualizados. Para grabar dichos cambios en la base de datos, deberá seleccionar el botón de continuar y aparecerá el siguiente mensaje: "ACTUALIZACION EXITOSA" y regresa automáticamente a la opción de usuarios.

Consultas:

Proceso

1.- Pide matrícula del alumno.

2.- Se despliega una pantalla para la captura mostrando la información correspondiente. Para salir de esta ventana deberá teclear (ESC) y regresa automáticamente a la opción de usuarios.

Máquinas:

Altas:

Proceso

1.- Se pide la información de la máquina mediante una pantalla para capturar los siguientes datos:

No. de Máquina

Tipo de Máquina

Serie de CPU

Serie de Teclado

Serie de Monitor

Serie de Mouse

Modelo

Capacidad de RAM

Capacidad de HD

Microprocesador

Tarjeta de Video

Capacidad de FD

Servidor

Isla

En Servicio

Una vez capturada la información se deberá seleccionar el botón de continuar para actualizar la información y quede registrada en la base de datos máquinas.
Si el proceso de Altas fue exitoso, desplegará una ventana con el siguiente mensaje: "ALTA EXITOSA", y regresará a la opción de máquinas.

ADVERTENCIA

- En caso de haber respondido por error que SI desea dar altas, aparecerá en la pantalla de captura de datos donde se deberá teclear (ESC), desplegándose el siguiente mensaje: "CAPTURA ABORTADA". En este caso, automáticamente regresará a la opción de Máquinas del menú.

Bajas:

Proceso

1.- Pide la clave de la máquina (número de esta).
2.- Se despliega una ventana con el siguiente mensaje: "MAQUINA BORRADA", y regresa automáticamente a la opción de máquinas.

Cambios:

Proceso

1.- Pide la clave de la máquina.

2.- Se despliega la pantalla de captura de datos, donde se realizan los cambios necesarios para ser actualizados. Para grabar dichos cambios en la base de datos, se deberá teclear simultáneamente (CTRL)(ENTER) y aparece el siguiente mensaje: "ACTUALIZACION EXITOSA" y regresa automáticamente a la opción de máquinas.

ADVERTENCIA

- En caso de haber cambiado la opción de máquina **En servicio** y ésta se desactive, se lleva a cabo un proceso de reasignación de apartados de todos los usuarios que tengan asignada dicha máquina.

- El proceso consiste en asignar una nueva máquina a la hora del apartado y en caso de no tener disponibilidad al momento de que el usuario ingrese al C.E.C. se desplegará el siguiente mensaje: "APARTADO CANCELADO POR MANTENIMIENTO".

Consultas:

Proceso

1.- Pide la clave de la máquina.

2.- Se despliega la pantalla de captura de datos, mostrando la información correspondiente. Para salir de esta pantalla de selecciona el botón de regresar y regresa automáticamente a la opción de máquinas.

Servidores

Altas:

Proceso

1.- Se pide la información del servidor mediante una pantalla para capturar los siguientes datos:

No. de Servidor

Nombre

Dirección Ethernet

Una vez capturada la información se selecciona el botón de continuar para actualizar la información y quede registrada en la base de datos de servidores.

Si el proceso fue exitoso, se desplegará el siguiente mensaje "ALTA EXITOSA", y regresa a la opción de servidores.

ADVERTENCIA

- En caso de haber respondido por error SI, aparecerá la pantalla de captura de datos donde se deberá teclear (ESC), desplegándose el siguiente mensaje: "CAPTURA ABORTADA". Regresará automáticamente a la opción de servidores del menú.

Bajas:

Proceso

1.- Pide la clave del servidor.

2.- Se despliega el siguiente mensaje: "SERVIDOR BORRADO", y regresa a la opción de servidores.

Cambios:

Proceso

1.- Pide la clave del servidor

2.- Se despliega la pantalla de captura de datos, donde se realiza la actualización. Para grabar los cambios en la base de datos, se teclea simultáneamente (CTRL)(ENTER) y aparece el siguiente mensaje: "ACTUALIZACION EXITOSA" y regresa a la opción de servidores.

Tipos de usuarios

Altas:

Proceso

1.- Se pide la información del tipo de usuario mediante una pantalla para capturar los siguientes datos:

Clave de tipo

Descripción

Una vez capturada la información deberá seleccionar el botón de continuar para actualizar los datos en la base de datos de tipos de usuarios.

Si el proceso fue exitoso aparece el siguiente mensaje "ALTA EXITOSA" y regresa a la opción de tipos de usuarios.

ADVERTENCIA

- En caso de haber respondido por error SI, aparecerá la pantalla de captura donde se deberá teclear (ESC), se desplegará el siguiente mensaje "CAPTURA ABORTADA", y regresa a la opción tipos de usuarios del menú.

Bajas:

Proceso

1.- Pide la clave de tipo.

2.- Se despliega el siguiente mensaje: "TIPO DE USUARIO BORRADO", regresa automáticamente a la opción de tipos de usuarios.

Cambios:

Proceso

1.- Pide la clave de tipo.

2.- Se despliega la pantalla de captura de datos, donde se realiza la actualización. Para grabar los cambios en la base de datos, se selecciona el botón de continuar, enseguida aparece el siguiente mensaje: "ACTUALIZACION EXITOSA", regresa a la opción de tipos de usuarios.

Consultas:

Proceso

1.- Pide la clave de tipo.

2.- Se despliega en pantalla la información correspondiente. Para salir de esta opción se selecciona el botón de regresar y regresa a la opción de tipos de usuarios.

Tipos de Máquinas:

Altas:

Proceso

1.- Se pide la información del tipo de máquina mediante una pantalla para capturar los siguientes datos:

Clave Tipo

Descripción

Una vez capturada la información se selecciona el botón de continuar para actualizar la información y quede registrada en la base de datos tipos de máquinas.

Si el proceso de Altas fue exitoso aparece el siguiente mensaje "ALTA EXITOSA", y regresa a la opción de tipos de máquinas.

ADVERTENCIA

- En caso de haber respondido SI por error, aparecerá la pantalla de captura, donde se deberá teclear (ESC), apareciendo el siguiente mensaje "CAPTURA ABORTADA", y regresa a la opción de tipos de máquinas del menú.

Bajas:

Proceso

1.- Pide la clave del tipo de máquina.

2.- Se despliega el siguiente mensaje: "TIPO DE MAQUINA BORRADO", regresa automáticamente a la opción de tipos de MAQUINAS.

Cambios:

Proceso

1.- Pide la clave de tipo de máquina.

2.- Se despliega la pantalla de captura de datos, donde se realiza la actualización. Para grabar los cambios en la base de datos se selecciona el botón de continuar, enseguida aparece el siguiente mensaje: "ACTUALIZACION EXITOSA", regresa a la opción de tipos de máquina.

Consultas:

Proceso

1.- Pide la clave de tipo de máquina.

2.- Se despliega en pantalla la información correspondiente. Para salir de esta opción se selecciona el botón de regresar y regresa a la opción de tipos de máquinas.

Módulo de Consultas:

Usuario:

Proceso

1.- Pide la matrícula del alumno.

2.- Despliega una ventana indicando si tiene máquina asignada y el número de la misma, en caso contrario aparece el siguiente mensaje: "NO TIENE MAQUINA ASIGNADA".

QUE HACER EN CASO DE FALLA

I.- Proceso a seguir en caso de falla eléctrica

1.- Si se dió acceso al Sistema como MAESTRO, el proceso es el siguiente:

Desde el prompt del sistema operativo, teclear la siguiente instrucción:

c:\> b (enter)

Entrar al subdirectorio TP\BIN tecleando la siguiente instrucción:

c:\> cd\tp\bin (enter)

El prompt quedará de la siguiente manera:

C:\TP\BIN>

Desde el nuevo prompt correr las siguientes instricciones:

C:\TP\BIN> reco (enter)

C:\TP\BIN> cierra (enter)

C:\TP\BIN> bd (enter)

De esta forma, se recuperaron y cerraron las bases de datos en uso al momento de la falla. Si se desea volver a correr el sistema de Apartados, se deberá teclear MAESTRO para continuar trabajando.

II.- Si se dió acceso como APARTADO el proceso a seguir es el siguiente:

Desde el prompt del sistema operativo, teclear la siguiente instrucción:

c:\> b (enter)

Entrar al subdirectorio TP\BIN tecleando la siguiente instrucción:

c:\> cd\tp\bin (enter)

El prompt quedará de la siguiente manera:

C:\TP\BIN>

Desde el nuevo prompt correr las siguientes instrucciones:

C:\TP\BIN> reco (enter)

C:\TP\BIN> cierra (enter)

C:\TP\BIN> cd (enter)

Deberá accesar a la red desde el directorio raíz, de la siguiente manera:

c:\> red (enter)

Pregunta el login

F:\LOGIN> login ccm_cec / aparta

El prompt queda como:

F:\USER\APARTA>

Desde el prompt se deberá teclear las siguientes instrucciones:

F:\USER\APARTA> reco (enter)

F:\USER\APARTA> cierra (enter)

F:\USER\APARTA> copy aparta.dat c: (enter)

F:\USER\APARTA> c: (enter)

Estando en raíz, se debe teclear lo siguiente:

c:\> bd (enter)

De esta forma se recuperaron y cerraron las bases de datos en uso tanto en la red como en el disco duro al momento de la falla. Si se desea volver al Sistema de Apartados, se deberá teclear APARTADO para continuar trabajando.

II.- Proceso a seguir en caso de que no funcione la red:

Si esta trabjando en red y esta deja de funcionar, el sistema está diseñado para seguir trabajando en disco duro. Al momento en que la red se active nuevamente el sistema recobrará los archivos y permitirá trabajar normalmente.

IV.3.2 MANUAL DEL USUARIO

INTRODUCCION

El sistema de administración del Centro Electrónico de Cálculo del ITESM-CCM, fue diseñado y desarrollado con el propósito de optimizar el uso, apartado y disponibilidad de los equipos de cómputo con que se cuenta en el CEC.

Este sistema fue desarrollado en forma modular, teniendo un nivel de privilegio para cada combinación de módulos en particular para poder ser ejecutados.

Este esquema de seguridad permite tener un control en cuanto a la información a manejar por cada uno de los usuarios dependiendo del privilegio asignado.

La interfaz con el usuario fue desarrollado en un ambiente gráfico para Windows, por lo tanto la operación del sistema se basa en controles que encontramos en las aplicaciones típicas de Windows.

Listas desplegables

Despliegue la lista seleccionándola y después presione la tecla de flecha descendente para seleccionar el elemento que desee de la lista, o si se cuenta con un Mouse presione sobre el elemento elegido.

Botones

Con los botones se ejecutan comandos o también dan la posibilidad de abrir una pantalla consecuente. Seleccione el botón deseado y presione <ENTER> o haga clic en él para ejecutar cierta acción. El botón elegido aparece en negritas.

Uso del teclado

Tecla < TAB >

Sirve para moverse a través de todos los botones, opciones o campos que se despliegan en la pantalla.

Tecla < ENTER >

Se podrá hacer uso de esta tecla para activar un botón; de aceptación, de cancelación o una selección.

Teclas de primeras letras

Se podrá seleccionar una opción mediante su primer letra.

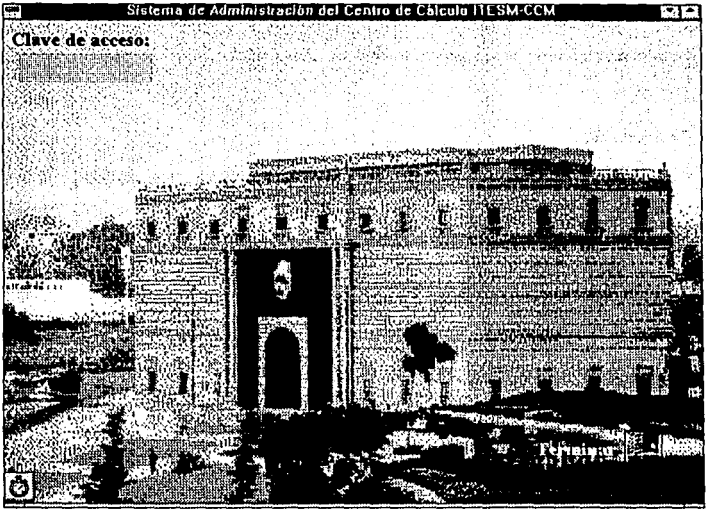
Tecla < ESC >

Con esta tecla se pueden eliminar los mensajes que aparecen en la pantalla,.

Por otra parte se cuenta con una serie de ventanas que se desplegarán a lo largo de la ejecución del sistema mostrando lo siguiente:

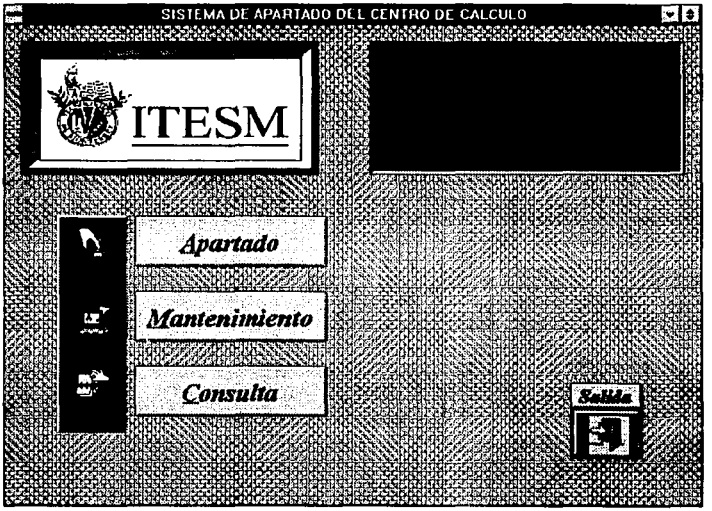
- Menúes
- Submenús
- Pantallas de Captura
- Catálogos de datos a Capturar
- Mensajes de error
- Mensajes de confirmación de operaciones

Al inicio de la sesión observara la siguiente pantalla.



Existen cuatro claves de acceso dependiendo del nivel de acceso del sistema de administración, al teclear alguna de éstas se desplegará el menú y las opciones a las cuales el usuario podrá tener acceso al sistema .

El sistema de Administración del Centro Electrónico de Cálculo consiste en tres módulos principales (Apartado, Mantenimiento y Consulta), pudiendo elegir cualquiera de ellos en la siguiente pantalla.



Se podrá elegir una opción del menú principal mediante dos maneras:

1.- Por medio de < TAB > hasta posicionarse en la opción deseada y oprimir <Enter> para que automáticamente despliegue los submenús correspondientes.

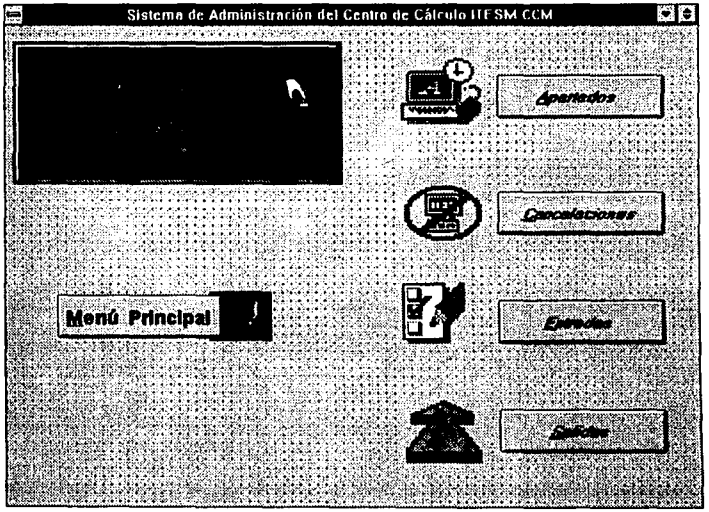
2.- Posicionándose por medio del Mouse y haciendo clic en la opción deseada.

Para abandonar un submenú y regresar al menú principal seleccione el ícono con la leyenda "Regresar al Menú principal", de ésta forma irá abandonando desde el proceso más interno al más externo.

Para terminar la ejecución del sistema deberá estar posicionado en el menú principal y seleccionar el ícono con la leyenda "Salir".

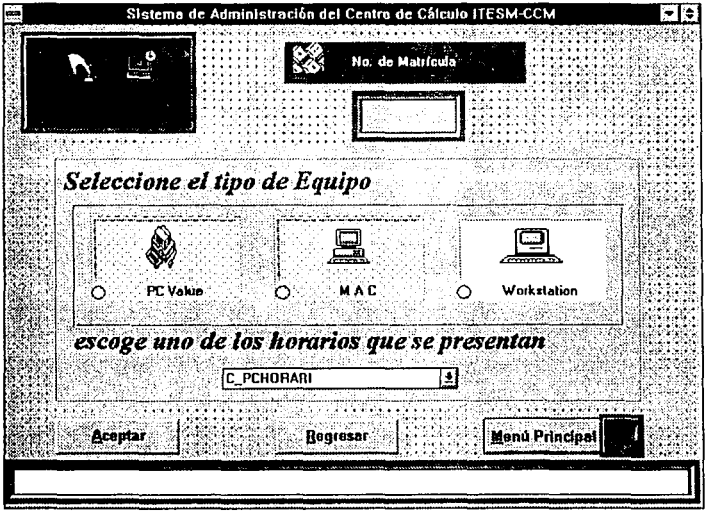
Descripción de cada módulo

Modulo de Apartado



Apartados

Permite realizar apartados de equipo, donde se determinan el tipo de máquina y hora a ser utilizado.



Al teclear el número de matrícula el sistema valida el número, si no existe enviará el siguiente mensaje: "Usuario no dado de alta"

Si ya se tiene apartado a una hora se enviará un mensaje advirtiéndolo.

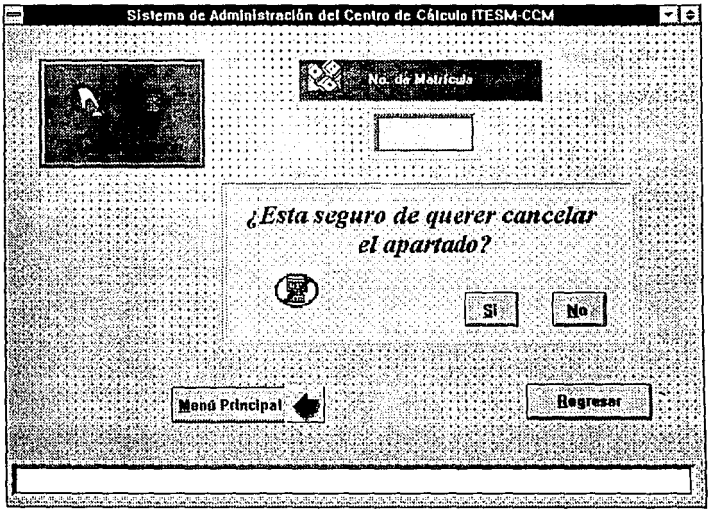
Posteriormente se muestran los tipos de máquina y horarios disponibles.

El sistema verifica si esta desocupado el tipo de máquina en la hora seleccionada.

De esta manera, el apartado queda automáticamente registrado y en una ventana de mensajes se indicará el número de máquina asignada, regresando al submenú.

Cancelaciones

Esta opción permite cancelar apartados realizados con anterioridad. Siendo éste su procedimiento :



En caso de que al teclear el número de matrícula aparezca el mensaje, "El usuario no tiene apartados" significa que el apartado para esa matrícula no fue elaborado.

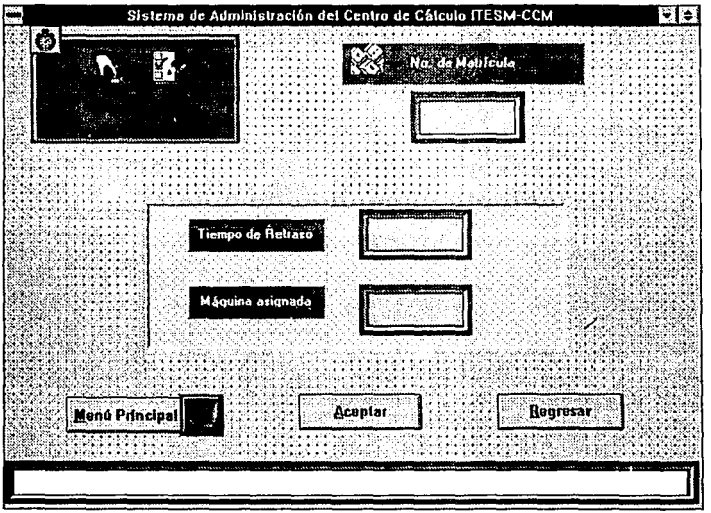
Si la respuesta es NO, despliega el mensaje "Cancelación abortada" y regresa al módulo de apartado.

Si la respuesta es SI, cancela automáticamente el apartado realizado previamente y regresa al módulo de apartados.

Solo se podrá realizar una cancelación siempre y cuando la hora en que éste se haga sea menor que la hora registrada en el apartado.

Entradas

Esta opción se utiliza para dar acceso a los usuarios al CEC.



Existen dos procesos:

a) Con apartado. Solicita la matrícula del usuario.

Despliega un mensaje indicando el tiempo de retraso con que entro y el número de máquina que fue asignada.

De esta manera queda registrada la entrada del usuario al CEC y regresa a la ventana en la que pide el número de matrícula para continuar dando entrada. En caso de querer regresar al modulo de apartado, se deberá teclear <ESC>.

Se debe registrar la entrada en el transcurso del minuto cero hasta el minuto quince de la hora en que se realizó el apartado. El sistema borra a partir del minuto 16 todos los apartados que están registrados sin entradas.

En caso de tener un apartado a una hora e intente registrar la entrada a otra, se desplegará en pantalla una ventana indicando el siguiente mensaje: "El usuario tiene apartado a otra hora", y no dará acceso hasta que la hora de entrada sea la misma que la hora del apartado.

Si un usuario desea entrar antes que la hora del apartado, el operador tendrá que dirigirse al módulo de apartado, al submenú de cancelaciones y cancelar el apartado.

Una vez realizado este proceso, podrá registrarse la entrada sin problema, advirtiendo al usuario que el apartado que tenía fue borrado para evitar confusiones.

b) Sin apartado.

Este proceso es para los usuarios que no realizaron el apartado de una máquina. Solicita la matrícula del usuario. Se despliega la pantalla para seleccionar el tipo de máquina y hora.

Despliega una ventana de mensajes indicando el número de máquina que fue asignada.

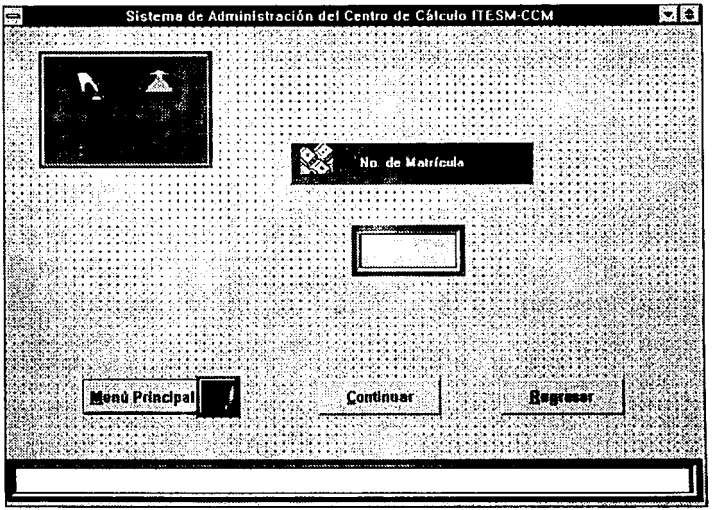
De esta manera, queda registrada la entrada del usuario al CEC y regresa a la ventana en que pide el número de matrícula para continuar dando entrada a usuarios. En caso de querer regresar al modulo de apartado se deberá teclear <ESC>.

La diferencia de estas dos opciones es que el usuario dependerá dela disponibilidad de equipo con que se cuente en el momento en que entre al CEC.

En ambos procesos pueden aparecer los siguientes mensajes: "El usuario no puede entrar o no ha salido"; en caso de haber tecleado una matrícula errónea o que no esté dado de alta en el catálogo de usuarios, por lo tanto no registrará entrada para esa matrícula "El usuario esta dentro del CEC"; significa que el usuario no puede registrar entrada al CEC más de una vez, en el transcurso de la hora.

Salidas

Se registra la salida del alumno del CEC. El sistema solicita el número de matrícula.

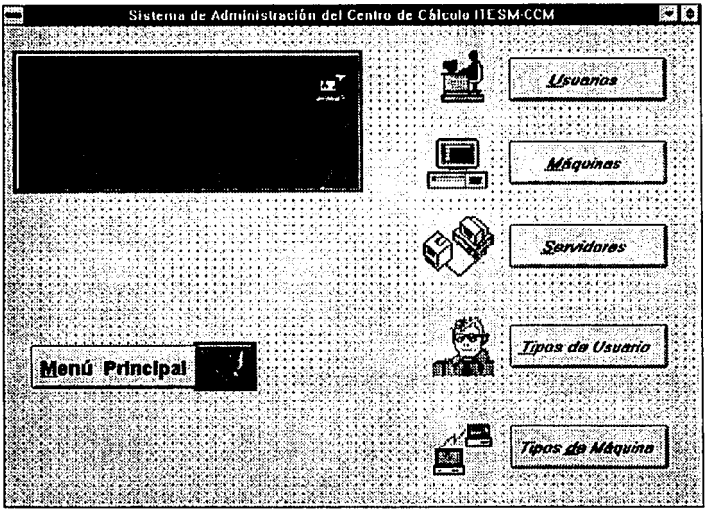


De esta forma da salida automáticamente al usuario de las instalaciones del CEC.

En caso de registrar salida antes del minuto 45 de la hora en uso, el equipo queda automáticamente liberado y podrá ser utilizado por otro usuario hasta el término de la hora.

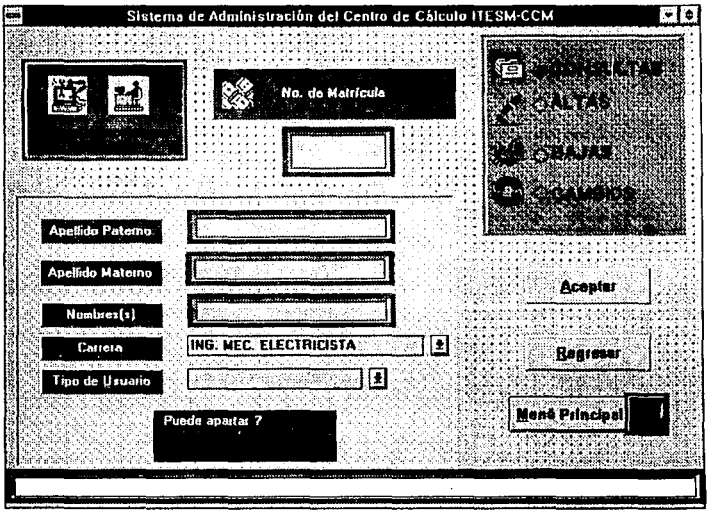
El sistema de administración libera y da salida automáticamente al minuto 49 de la hora en uso, a todas las matrículas que estén registradas como entradas a esa hora.

MODULO DE MANTENIMIENTO



Este módulo está formado por los submódulos usuarios, máquinas, servidores, tipos usuarios, tipos máquina. Su función es dar mantenimiento a los catálogos, por lo que cuenta con pantallas de captura a fin de facilitar la tarea de actualizar la información de las bases de datos. Cada uno de los submódulos se compone a su vez de altas, bajas, cambios, consultas.

Usuarios


Sistema de Administración del Centro de Cálculo ITESM-CCM
No. de Matrícula
CONSULTAS
ALTAS
BAJAS
CAMBIOS
Apellido Paterno
Apellido Materno
Nombres(s)
Carrera
ING. MEC. ELECTRICISTA
Tipo de Usuario
Puede apartar ?
Aceptar
Regresar
Menú Principal

Altas

En esta opción se solicitan los datos que identifican al usuario para hacer su ingreso a la base de datos.

Cada uno de los campos tienen las siguientes características:

- Número de matrícula. El sistema validará el número de matrícula, enviando un mensaje en caso de que ya exista en la base de datos.

- Apellido paterno. El sistema se asegurara que este campo no se deje en blanco. El número máximo de caracteres será de 20.

- Apellido Materno. El sistema se asegurara que este campo no se deje en blanco. El número máximo de caracteres será de 20.

- Nombre(s). El sistema se asegurara que este campo no se deje en blanco. El número máximo de caracteres será de 20.

-Tipo de usuario. Cada uno de estos tendrá una clave que los identificará individualmente por:

1. Alumno
2. Profesor
3. Ex-alumno
4. Administrativo

- Puede apartar. Al seleccionar este campo, se determina si tiene derecho de apartar una hora de equipo.

- Carrera. En este campo se teclea la clave de la carrera a la que pertenece el usuario.

Al terminar la captura de los datos, se hace clic en el botón ACEPTAR para confirmar el alta del usuario.

Si se desea volver al menú de mantenimiento, se deberá hacer clic en el botón regresar.

Bajas

Con esta opción se podrá dar de baja a un usuario con el siguiente proceso:

- Solicita la matrícula.- El sistema validará su existencia, en caso de encontrarla desplegara una ventana solicitando la confirmación de baja para ese usuario haciendo

clic en el botón ACEPTAR, en caso contrario, lo indicará mediante una ventana de mensaje.

Cambios

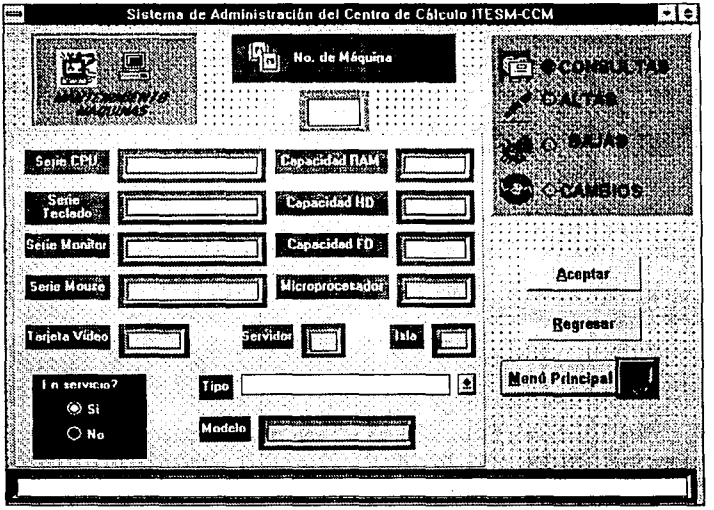
En esta opción se podrán realizar cambios a cualquier campo de los datos que identifican al usuario.

Se desplegará en la pantalla la ventana de captura de datos, donde se realizara la actualización de los datos.

Consultas

Esta opción permite consultar cada uno de los datos que identifican al usuario. Solicitando la matrícula desplegará una ventana de confirmación para la consulta de este usuario, haciendo clic en el botón de aceptar aparecerá la pantalla de captura mostrando la información correspondiente.

Máquinas



Altas

En esta opción se solicitan los datos que identifican a la máquina para hacer su ingreso a la base de datos.

Cada uno de los campos tienen las siguientes características:

- Número de máquina. El sistema validará el número de máquina, que constituye su clave. Si existe se desplegará una ventana indicándolo.

-Tipo de máquina. El sistema se asegurará que este campo no quede en blanco.

1. MAC
2. IBM
3. PC

- Número de serie del CPU. Se teclea el número de serie correspondiente con un máximo de 15 caracteres.

- Número de serie del Teclado Se teclea el número de serie correspondiente con un máximo de 15 caracteres.

- Número de serie del monitor. Se teclea el número de serie correspondiente con un máximo de 15 caracteres.

- Número de serie del Mouse. Se teclea el número de serie correspondiente con un máximo de 15 caracteres.

- Modelo. Se teclea el modelo de la máquina con un máximo de 15 caracteres.

- Memoria RAM. Teclear el número correspondiente a la cantidad de memoria RAM del equipo en megabytes, para ello se cuenta con un campo de tres caracteres.

- HD. Teclear el número correspondiente de la capacidad de disco duro en megabytes, en un campo de cuatro caracteres.

- Microprocesador. Teclear el tipo de microprocesador correspondiente. El campo cuenta con seis caracteres.

- Video. Se deberá teclear el tipo de tarjeta de video de esta máquina. El campo contara con cinco caracteres máximo.

-FD. Teclear el número correspondiente de la capacidad del manejador del disco flexible en caso de que la máquina contará con este, y deberá proporcionarse en kilobytes en un campo de 5 caracteres.

- Servidor. Teclear el número de servidor al que esta conectada la máquina. El campo cuenta con dos caracteres.

- Isla. Teclear el número de isla (Grupo de Computadoras) en que esta ubicado físicamente dicho equipo. El campo cuenta con dos caracteres.

- En servicio. Teclear si el equipo esta en servicio o no (S, en servicio y N, no en servicio).

Al terminar la captura de los datos, se hace clic en el botón ACEPTAR para confirmar el alta de la máquina.

Si se desea volver al menú de mantenimiento, se deberá hacer clic en el botón regresar.

Pendiente: botón terminar

Bajas

Con esta opción se podrá dar de baja a una máquina con el siguiente proceso:

- Solicita la clave de la máquina.- El sistema validará su existencia, en caso de encontrarla desplegara una ventana solicitando la confirmación de baja para esa máquina haciendo clic en el botón ACEPTAR, en caso contrario, lo indicará mediante una ventana de mensaje.

Cambios

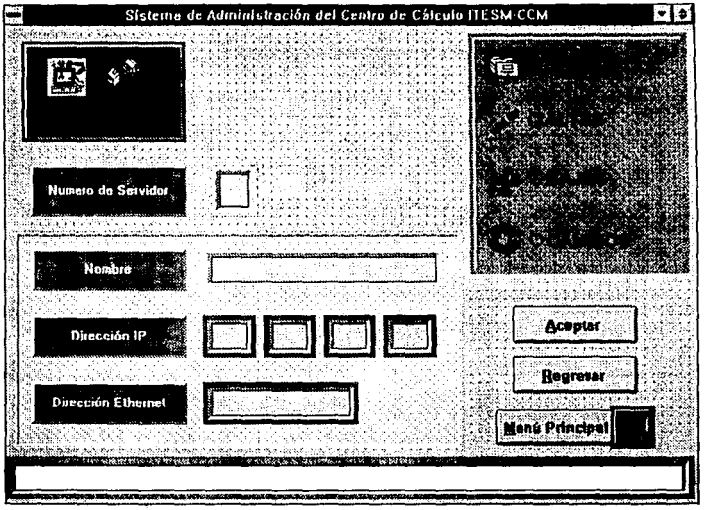
En esta opción se podrán realizar cambios a cualquier campo de los datos que identifican a la máquina.

Se desplegará en la pantalla la ventana de captura de datos, donde se realizara la actualización de los datos.

Consultas

Esta opción permite consultar cada uno de los datos que identifican a la máquina. Solicitando la clave de la máquina se desplegará una ventana de confirmación para la consulta de esta, haciendo clic en el botón de aceptar aparecerá la pantalla de captura mostrando la información correspondiente.

Servidores



Altas

En esta opción se solicitan los datos que identifican al servidor para hacer su ingreso a la base de datos.

Cada uno de los campos tienen las siguientes características:

- Número de servidor. El sistema validará el número de servidor, que constituye su clave. Si existe se desplegará una ventana indicándolo.

-Nombre. Se deberá teclear el nombre con el que fue bautizado el servidor con un máximo de 20 caracteres.

- Dirección. Teclear el numero de la dirección IP, necesaria para el protocolo de comunicaciones. El campo contará con 12 caracteres en grupos de 3.

- Dirección Ethernet. Teclear el número de la dirección Ethernet del servidor necesaria para compartir información a nivel mundial con un máximo de 15 caracteres.

Al terminar la captura de los datos, se hace clic en el botón ACEPTAR para confirmar el alta del servidor.

Si se desea volver al menú de mantenimiento, se deberá hacer clic en el botón regresar.

Bajas

Con esta opción se podrá dar de baja a un servidor con el siguiente proceso:

- Solicita la clave del servidor. El sistema validará su existencia, en caso de encontrarla desplegara una ventana solicitando la confirmación de baja para ese servidor haciendo clic en el botón ACEPTAR, en caso contrario, lo indicará mediante una ventana de mensaje.

Cambios

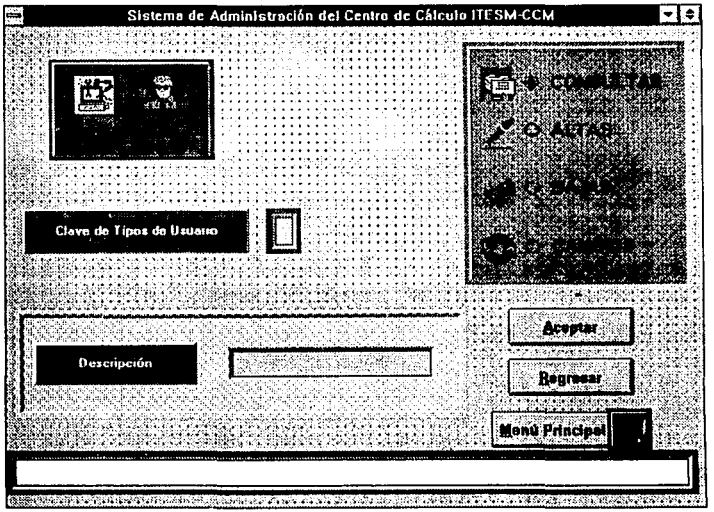
En esta opción se podrán realizar cambios a cualquier campo de los datos que identifican al servidor.

Se desplegará en la pantalla la ventana de captura de datos, donde se realizara la actualización de los datos.

Consultas

Esta opción permite consultar cada uno de los datos que identifican al servidor Solicitando la clave del servidor se desplegará una ventana de confirmación para la consulta de este, haciendo clic en el botón de aceptar aparecerá la pantalla de captura mostrando la información correspondiente.

Tipos de usuarios



Altas

En esta opción se solicitan los datos que identifican al tipo de usuario para hacer su ingreso a la base de datos.

Cada uno de los campos tienen las siguientes características:

- Clave tipo. Proporcionará el tipo de usuario (alumno, ex-alumno, administrativo, etc.) el cual constituye su clave. El sistema lo validará. Si existe se desplegará una ventana indicándolo. El campo contará con un carácter.

- Descripción. -Teclear una breve descripción del tipo de usuario. el campo contara con 20 caracteres.

Al terminar la captura de los datos, se hace clic en el botón ACEPTAR para confirmar el alta del tipo de usuario.

Si se desea volver al menú de mantenimiento, se deberá hacer clic en el botón regresar.

Con el botón terminar regresa a menú principal.

Bajas

Con esta opción se podrá dar de baja a un tipo de usuario con el siguiente proceso:

- Solicita la clave del tipo de usuario. El sistema validará su existencia, en caso de encontrarla desplegara una ventana solicitando la confirmación de baja para ese

tipo de usuario haciendo clic en el botón ACEPTAR, en caso contrario, lo indicará mediante una ventana de mensaje.

Cambios

En esta opción se podrán realizar cambios a cualquier campo de los datos que identifican al tipo de usuario.

Se desplegará en la pantalla la ventana de captura de datos, donde se realizara la actualización de los datos.

Consultas

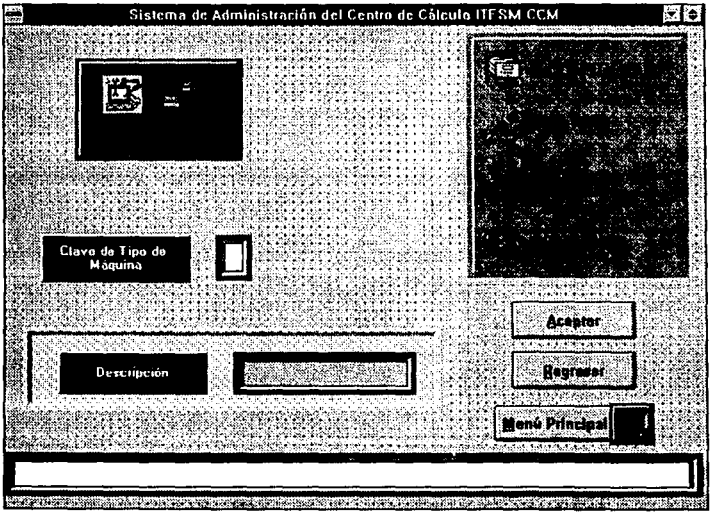
Esta opción permite consultar cada uno de los datos que identifican al tipo de usuario. Solicitando la clave del tipo de usuario se desplegará una ventana de confirmación para la consulta de este, haciendo clic en el botón de aceptar aparecerá la pantalla de captura mostrando la información correspondiente.

Al terminar la captura de los datos, se hace clic en el botón ACEPTAR para confirmar el alta del tipo de usuario.

Si se desea volver al menú de mantenimiento, se deberá hacer clic en el botón regresar.

Con el botón terminar regresa al menú principal.

Tipos de Máquinas



Altas

En esta opción se solicitan los datos que identifican al tipo de máquina para hacer su ingreso a la base de datos.

Cada uno de los campos tienen las siguientes características:

- Clave tipo. Proporcionará el tipo de máquina (IBM, PC, MAC, etc.) el cual constituye su clave. El sistema lo validará. Si existe se desplegará una ventana indicándolo. El campo contará con un carácter.

- Descripción. -Teclear una breve descripción del tipo de máquina. el campo contara con 20 caracteres.

Bajas

Con esta opción se podrá dar de baja a un tipo de máquina con el siguiente proceso:

- Solicita la clave del tipo de máquina. El sistema validará su existencia, en caso de encontrarla desplegara una ventana solicitando la confirmación de baja para ese tipo de máquina haciendo clic en el botón ACEPTAR, en caso contrario, lo indicará mediante una ventana de mensaje.

Cambios

En esta opción se podrán realizar cambios a cualquier campo de los datos que identifican al tipo de máquina.

Se desplegará en la pantalla la ventana de captura de datos, donde se realizara la actualización de los datos.

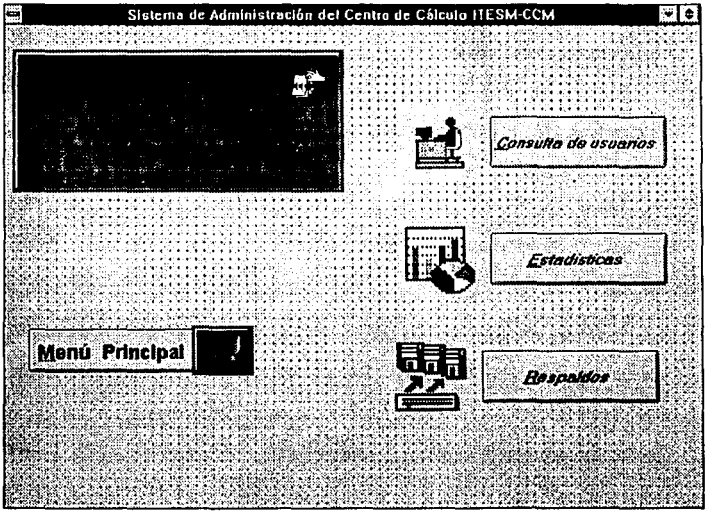
Consultas

Esta opción permite consultar cada uno de los datos que identifican al tipo de máquina Solicitando la clave del tipo de máquina se desplegará una ventana de confirmación para la consulta de este, haciendo clic en el botón de aceptar aparecerá la pantalla de captura mostrando la información correspondiente.

Al terminar la captura de los datos, se hace clic en el botón ACEPTAR para confirmar el alta del tipo de máquina.
Si se desea volver al menú de mantenimiento, se deberá hacer clic en el botón regresar.

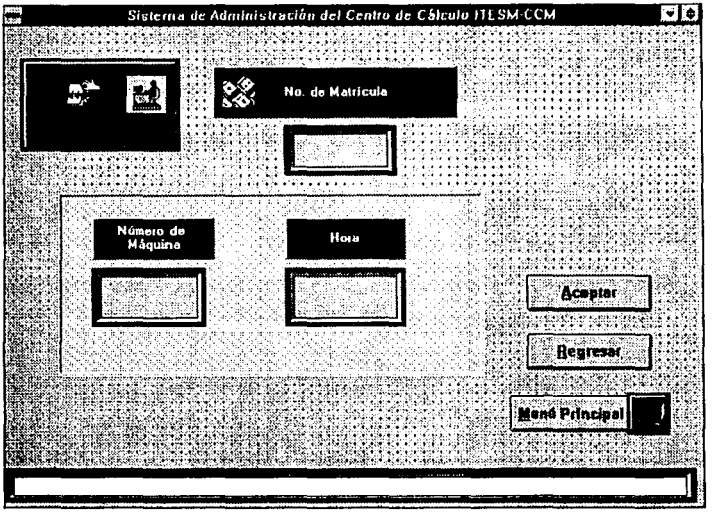
Con el botón terminar regresa al menú principal.

MODULO DE CONSULTAS



Este módulo está formado por los submódulos consulta de usuarios y respaldo. Su función es permitir consultar los apartados de un usuario y respaldar la información de la base de datos en disco flexible.

Consultas

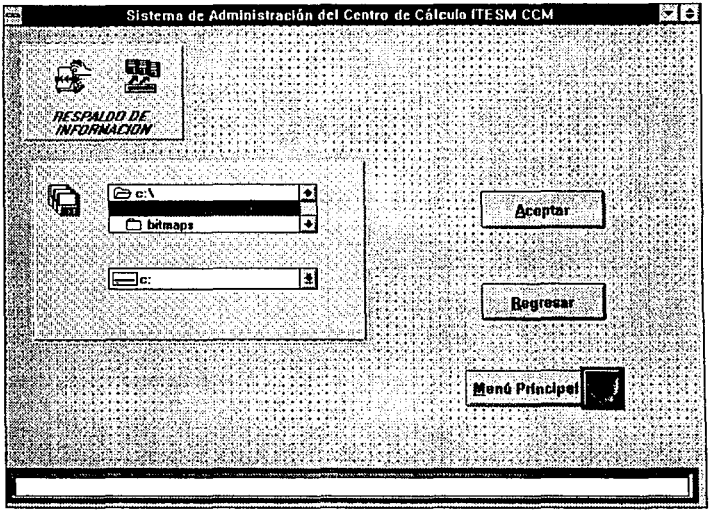


Permite consultar si un usuario tiene apartada una máquina a una hora determinada. y la opción de respaldos realiza el respaldo de la información en disco flexible.

Los campos tienen las siguientes características:

Número de matrícula. El sistema validará el número de matrícula, enviando un mensaje en caso de que ya exista en la base de datos.

Respaldo



Si desea respaldo. Se deberá escoger la opción si continua el respaldo de la información en disco flexible.

Al terminar la captura de los datos, se hace clic en el botón ACEPTAR para confirmar el alta de la acción de consulta de apartado de un usuario o respaldo de la información.

Si se desea volver al menú de consultas, se deberá hacer clic en el botón regresar.

Conclusiones

CONCLUSIONES

La evaluación más fidedigna del resultado de un sistema, es la repercusión que tiene y la utilidad que ofrece. Bajo este contexto, el presente trabajo de tesis, ha sido elaborado considerando todos y cada uno de los bemoles que se distinguieron en las etapas de diseño y desarrollo.

Es importante considerar que el sistema se encuentra funcionando ahora mismo en las instalaciones del ITESM-CCM, con resultados muy importantes en la consecución de los objetivos planteados al inicio del sistema.

La retroalimentación del usuario permitió que en etapas tempranas del diseño, se consideraran todos los factores necesarios de manera que el producto final cumple ampliamente con las expectativas y cubre cabalmente los requerimientos iniciales.

La utilización de las actuales técnicas de implementación de sistemas, nos ayudará a tener una mayor oportunidad de controlar y depurar nuestros proyectos. Su flexibilidad permiten aplicarse muy fácilmente a cualquier tipo y caso de estudio, convirtiéndose así en una excelente herramienta de trabajo, la cual aunada al gran crecimiento en el poder de cómputo de los equipos personales y en el surgimiento de programas auxiliares para el control de los proyectos, nos permitirán un aprovechamiento al

máximo de las mismas; logrando como resultado un sistema de muy alta calidad, en un tiempo más cercano al estimado y cumpliendo con las especificaciones establecidas y requerimientos reales del usuario.

La elaboración del presente trabajo requirió de la combinación de bases teóricas firmes y un sentido práctico, que a su vez nos permitieron formalizar y expandir nuevos conocimientos.

Comprobamos que la aplicación de métodos estructurados de desarrollo de sistemas de cómputo permiten resolver eficientemente las necesidades de trabajo de quienes los requieren. Tales métodos son considerablemente consistentes y son una herramienta eficiente, desde el proceso de análisis hasta la implantación e integración de todas la soluciones posibles.

Bibliografía

BIBLIOGAFIA

Carlo Bantini, Stefano Ceri

Conceptual Data Base Design

Benjamin/Cummings

Date

Introducción a los sistemas de bases de datos

Addison Wesley

Doug Bierer, Charles Hatch, Dee Anne Higley

Netware 4 for professionals

De. New Riders Publishing

Fairley

Ingeniería de Software

McGraw Hill

Feldman

Using Visual Basic 3

QUE

FitzGeral

Fundamentos de Análisis de Sistemas

CECSA

F. Korth, Henry, Silver Schatz, Abraham

Database Concepts

McGraw-Hill

González Sainz, Nestor

Comunicaciones y redes de procesamiento de datos

McGraw Hill

Kruse

Estructura de Datos y Diseño de Programas

Prentice Hall

Novell Netware 3.12

Brieve Installation and operation

Network Computing Products

Professional Development Series

Btrieve for Dos: Installation and operation

Scott, D.F.

Visual Basic by Example

QUE

S. Pressman, Roger

Software Engineering

MCGraw Hill

Tanenbaum , Andrew S.

Redes de ordenadores

Editorial Prentice Hall

Winsberg, Paul

Relational database design workshop

Sybase Inc.

Yourdon , Edward

Modern Structured Analysis

Yourdon Press computing

Apéndices

APENDICE 1

A continuación se presenta el código correspondiente a las rutinas que se emplean para el manejo de la base de datos mediante el lenguaje de programación Turbo Pascal v7.0.

```
Program apa(input,output,Aparta);
uses
  btr;
Type
  arr14  = array [1..14] of char;
  arr128 = array [1..128] of char;
  arr20  = array [1..20] of char;
  arr6   = array [1..6] of char;
  arr15  = array [1..15] of char;
  arr3   = array [1..3] of char;
  arr2   = array [1..2] of char;
  arr4   = array [1..4] of char;
  arr5   = array [1..5] of char;
  arr10  = array [1..10] of char;
  arr13  = array [1..13] of char;
  arr29  = array [1..29] of char;
  arr40  = array [1..40] of char;
  RegistroUsu = Record
                matricula : arr6;
                paterno : arr20;
                materno : arr20;
                nombre : arr20;
                tipo : char;
                habil : char;
                status : char;
                sanciones : arr2;
                carrera : arr6;
               end;
  RegistroMaq = Record
                claveMaq : arr3;
                claveTipo : char;
                serieCPU : arr15;
                serieTeclado : arr15;
                serieMonitor : arr15;
                serieMouse : arr15;
                modelo : arr15;
                RAM : arr3;
                HD : arr4;
                micro : arr6;
                video : arr5;
                FD : arr5;
                servidor : arr2;
                isla : arr2;
                enServicio : char;
               end;
  RegistroApa = Record
                matr_ap : arr6;
                cve_hora_ap : arr2;
                cve_maq_ap : arr3;
                cve_tipo_ap : char;
```

```
        end;
RegistroOcu = Record
        matricula : arr6;
        hora : arr2;
        maquina : arr3;
        Tipo_maq : char;
      end;
RegLlaveUsu1= Record
      paterno : arr20;
      materno : arr20;
      nombre : arr20;
      end;
RegLlaveUsu2= Record
      paterno  : arr20;
      materno  : arr20;
      nombre   : arr20;
      matricula : arr6;
      end;
RegLlaveUsu3= Record
      matricula : arr6;
      carrera : arr6;
      end;
RegLlaveMaq1= Record
      ClaveTipo  : char;
      EnServicio : char;
      end;
RegLlaveApa1= Record
      clave_Hora_ap : arr2;
      clave_Maq_ap : arr3;
      clave_Tipo_ap : char;
      end;
var
 Apart    : Text;
 matricula,
 arr_matri_apa : arr6;
 reg_apa   : RegistroApa;
 reg_ocu   : RegistroOcu;
 reg_usu   : RegistroUsu;
 reg_maq   : RegistroMaq;
 long_reg_apa,
 long_reg_ocu,
 long_reg_Usu,
 long_reg_Maq,
 estado_bt1,
 estado_bt2,
 estado_bt3 : Integer;
 bp_apa,
 bp_ocu,
 bp_maq,
 bp_usu    : arr128;
 arch_apa,
 arch_ocu,
 arch_maq,
 arch_usu  : arr29;
 llaveApa0,
 llaveOcu0 : arr6;
 llaveApa1 : RegLlaveApa1;
 llaveApa2 : arr2;
 llaveApa4 : arr3;
 llaveOcu1 : arr3;
 llaveOcu2 : arr2;
 llaveOcu3 : char;
 llaveUsu0 : arr6;
 llaveUsu1 : RegLlaveUsu1;
 llaveUsu2 : RegLlaveUsu2;
 llaveMaq0 : arr3;
 llaveMaq1 : RegLlaveMaq1;
 FDA       : boolean; {fin de archivo}
 CveTma    : char;
```

```
{*********************************************}
Procedure AbreArchivos;
 begin
  long_reg_apa := sizeof(reg_apa);
  arch_apa := 'f:\user\aparta\aparta.DAT   ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  long_reg_ocu := sizeof(reg_ocu);
  arch_ocu := 'f:\user\aparta\ocupa.DAT   ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
  long_reg_usu := sizeof(reg_usu);
  arch_usu := 'f:\user\aparta\usuario.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_usu,reg_usu,long_reg_usu,arch_usu,0);
  long_reg_ocu := sizeof(reg_maq);
  arch_maq := 'f:\user\aparta\maquina.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_maq,reg_maq,long_reg_maq,arch_maq,0);
 end;
{ end procedure AbreArchivos}
{*********************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_usu,reg_usu,long_reg_usu,arch_usu,0);
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_maq,reg_maq,long_reg_maq,arch_maq,0);
 end;
{ end procedure AbreArchivos}

{*********************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                    var arreglo_salida; { Arreglo de salida }
                    maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*********************************************}

function arreglo_a_cadena(  var arreglo_entrada; { Arreglo de entrada }
                    maxcar : byte) : string;
                    { Numero maximo de caracteres del arreglo fuente}
const
 MAX = 74;
var
 cadena_salida : string;
 long : byte absolute cadena_salida;
 arreglo : array[1..MAX] of char absolute arreglo_entrada;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to maxcar do cadena_salida[i] := arreglo[i];
 long := maxcar;

 { Se eliminan los blancos al final de la cadena }

 while (cadena_salida[long] = #32 {ESPACIO}) and (long > 0) do dec(long);
 arreglo_a_cadena := cadena_salida;
end; { arreglo_a_cadena }
```

```
(*****************************************************)

BEGIN
 {Codigo que se emplea para que se puedan hacer los accesos}
 long_reg_apa:=sizeof(Reg_Apa);
 long_reg_ocu:=sizeof(Reg_Ocu);
 long_reg_usu:=sizeof(Reg_Usu);
 long_reg_maq:=sizeof(Reg_Maq);
 assign(Apart,'f:\user\aparta\apart.txt');
 rewrite(Apart);
 AbreArchivos;

 {Codigo que se emplea para el apartado propiamente dicho}

 Cadena_a_Arreglo(ParamStr(1),arr_matri_apa,6);
 LlaveUsu0:=arr_matri_apa;
 estado_bt3:=btrv(REGISTRO_IGUAL,bp_usu,reg_usu,long_reg_usu,
              llaveUsu0,0);

 { verifica si es un usuario v lido }
 if estado_bt3<>0 then
  writeln(Apart,'1,USUARIO NO DADO DE ALTA!!')
 else
  if reg_usu.habil<>'S' then
   writeln(Apart,'2,USUARIO DESHABILITADO!!')
  else
   begin
    estado_bt2:=btrv(REGISTRO_IGUAL,bp_ocu,reg_ocu,long_reg_ocu,
                arr_matri_apa,0);
    estado_bt1:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,long_reg_apa,
                arr_matri_apa,0);
    { verifica si no tiene apartados }
    if (estado_bt1=0) or (estado_bt2<0) then
     writeln(apart,'3,EL USUARIO TIENE OTRO APARTADOA O ESTA EN EL C.E.C.')
    else
     begin
      Cadena_a_Arreglo(ParamStr(2),LlaveMaq1.ClaveTipo,1);
      LlaveMaq1.EnServicio:='S';
      estado_bt1:=btrv(REGISTRO_IGUAL,bp_maq,reg_maq,
                  long_reg_maq,LlaveMaq1,1);
      { verifica si el tipo de m quina es v lido }
      if estado_bt1<>0 Then
       writeln(Apart,'4,NO EXISTE ESE TIPO DE MAQUINA')
      else
       begin
        repeat
         with llaveApa1 do
          begin
           Cadena_a_Arreglo(ParamStr(3),Clave_Hora_ap,2);
           clave_maq_ap:=Reg_Maq.ClaveMaq;
           clave_tipo_ap:=Reg_Maq.claveTipo;
           CveTma:=clave_tipo_ap;
          end;
         { end with}
         { verifica si la m quina no est  apartada }
         estado_bt1:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,
                     long_reg_apa,LlaveApa1,1);
         if estado_bt1=0 Then {ya estaba apartada}
          begin
           estado_bt1:=btrv(REGISTRO_SIGUIENTE,bp_maq,
                       reg_maq,long_reg_maq,LlaveMaq1,1);
           if estado_bt1=9 then
            begin
             FDA:=True;
             writeln(Apart,'5,NO HAY MAQUINAS DISPONIBLES!!');
            end
           else
            if reg_maq.ClaveTipo<>CveTMa Then
```

```
                begin
                  FDA:=True;
                  writeln(Apart,'5,NO HAY MAQUINAS DISPONIBLES!!');
                end;
              { end if }
              estado_bt1:=99;
            end
          else  {no estaba apartada}
            begin
              with reg_apa do
                begin
                  matr_ap:=arr_matri_apa;
                  Cadena_a_Arreglo(ParamStr(3),Cve_Hora_ap,2);
                  cve_maq_ap:=Reg_Maq.ClaveMaq;
                  cve_tipo_ap:=Reg_Maq.claveTipo;
                end;
              { end with }
              estado_bt1:=btrv(INSERTAR_REGISTRO,bp_apa,
                               reg_apa,long_reg_apa,LlaveApa1,0);
              if estado_bt1<>0 Then
                writeln(Apart,'6,NO SE PUDO INSERTAR EL APARTADO')
              else
                writeln(Apart,'0,LA MAQUINA ASIGNADA ES LA NUMERO '+
                        arreglo_a_cadena(reg_apa.cve_maq_ap,3));
              { end if }
            end;
          { end if }
        Until FDA or (estado_bt1=0);
        FDA:=FALSE;
      end;
    { end if }
  end;
 end;
{ end if }
{ end if }
Close(Apart);
CierraArchivos;
end.

Program cancel(input,output,cancela);
uses
 btr;
Type
 arr2  = array[1..2] of char;
 arr3  = array[1..3] of char;
 arr6  = array[1..6] of char;
 arr29 = array[1..29] of char;
 arr128 = array[1..128] of char;
 RegistroApa = Record
             matr_ap : arr6;
             cve_hora_ap : arr2;
             cve_maq_ap : arr3;
             cve_tipo_ap : char;
           end;

var
 cancela  : Text;
 matricula : arr6;
 reg_apa  : RegistroApa;
 long_reg_apa : integer;
 bp_apa   : arr128;
 arch_apa  : arr29;

{*************************************************}
Procedure AbreArchivos;
 begin
  long_reg_apa := sizeof(reg_apa);
  arch_apa := 'F:\USER\APARTA\APARTA.DAT   ';
```

```
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
 end;
{ end procedure AbreArchivos}
{*****************************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
 end;
{ end procedure AbreArchivos}

{*****************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                    var arreglo_salida; { Arreglo de salida }
                    maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*****************************************************}

Begin
 {Código que se emplea para que se puedan hacer los accesos}
 long_reg_apa:=sizeof(Reg_Apa);
 assign(Cancela,'F:\USER\APARTA\CANCELA.TXT');
 rewrite(Cancela);
 AbreArchivos;

 {Código que se emplea para el acceso propiamente dicho}

 cadena_a_arreglo(ParamStr(1),matricula,6);
 estado_bt:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,long_reg_apa,
                 matricula,0);
 if (estado_bt=0) then
  begin
   estado_bt:=btrv(BORRAR_REGISTRO,bp_apa,reg_apa,long_reg_apa,
             matricula,0);
   if estado_bt=0 then
    begin
     writeln(Cancela,'0,APARTADO CANCELADO EXITOSAMENTE!!!');
    end
   else
    begin
     writeln(cancela,'1,IMPOSIBLE CANCELAR APARTADO!!!');
    end
   {end if}
  end
 else
  writeln(cancela,'4,EL USUARIO NO TIENE APARTADO!!!');
 {end if}
 Close(Cancela);
 CierraArchivos;
end.

Program Tipo_Maquina(Input,Output,TMa);
uses
 btr;

{*****************************************************}
```

```
Type
 arr29  = array [1..29] of char;   {para el nombre del archivo btrieve}
 arr128 = array [1..128] of char;  {para el databuffer}
 arr20  = array [1..20] of char;   {para desc. de m quina}
 RegistroTma = Record              {registro del archivo de tipoMaq}
          ClaveTipoMaq   : char;
          DescripTipoMaq : arr20;
          end;
var
 FDA        : boolean;
 estado_bt,
 long_reg_TMa,
 codigo     : integer;
 aborto     : boolean;
 bp_TMa     : arr128;
 reg_Tma    : RegistroTma;
 llaveTma0  : char;
 arch_TMa   : arr29;
 TMa        : Text;
 Operacion  : byte;
{*****************************************************}
Procedure AbreArchivos;
 begin
  long_reg_Tma := sizeof(reg_Tma);
  arch_TMa := 'f:\user\aparta\TIPOMAQ.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_TMa,reg_Tma,long_reg_Tma,arch_TMa,0);
 end;
{ end procedure AbreArchivos}
{*****************************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_TMa,reg_Tma,long_reg_Tma,arch_TMa,0);
 end;
{ end procedure AbreArchivos}

{*****************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*****************************************************}

Begin
 {Código que se emplea para que se puedan hacer los accesos}
 long_reg_Tma:=sizeof(reg_Tma);
 assign(TMa,'f:\user\aparta\catamaq.txt');
 rewrite(TMa);
 AbreArchivos;

 {Código que se emplea para el acceso propiamente dicho}
 Estado_bt:= btrv(Primer_Registro,bp_TMa,reg_Tma,long_reg_Tma,
                  llaveTMa0,0);
 if estado_bt=0 then
   begin
     writeln(TMa,reg_Tma.ClaveTipoMaq+','+reg_tma.DescripTipoMaq);
```

```
    repeat
      Estado_bt:= btrv(Registro_Siguiente,bp_TMa,reg_Tma,long_reg_Tma,
                  ClaveTMa0,0);
      if estado_bt=0 then
        writeln(TMa,reg_Tma.ClaveTipoMaq+','+reg_tma.DescripTipoMaq)
      else
        FDA:=True;
      { end if }
    Until FDA;
  end;
 { end if }
 CierraArchivos;
 close(TMa);
End.

Program Tipo_Usuarios(Input,Output,TUs);
uses
 btr;

{*********************************************}
Type
 arr29  = array [1..29] of char;   {para el nombre del archivo btrieve}
 arr128 = array [1..128] of char;  {para el databuffer}
 arr20  = array [1..20] of char;   {para desc. del usuario}
 RegistroTUs = Record              {registro del archivo de tipoUsu}
          ClaveTipoUsu  : char;
          DescripTipoUsu : arr20;
          end;
var
 FDA        : boolean;
 estado_bt,
 long_reg_TUs,
 codigo     : integer;
 aborto     : boolean;
 bp_TUs     : arr128;
 reg_TUs    : RegistroTUs;
 llaveTUs0  : char;
 arch_TUs   : arr29;
 TUs        : Text;
 Operacion  : byte;
{*********************************************}
Procedure AbreArchivos;
 begin
  long_reg_TUs := sizeof(reg_TUs);
  arch_TUs := 'f:\user\aparta\TIPOUSU.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_TUs,reg_TUs,long_reg_TUs,arch_TUs,0);
 end;
{ end procedure AbreArchivos}
{*********************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_TUs,reg_TUs,long_reg_TUs,arch_TUs,0);
 end;
{ end procedure AbreArchivos}

{*********************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 I : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
```

```
  for I := 1 to long do arreglo[I] := cadena_entrada[I];
  for I := succ(long) to maxcar do arreglo[I] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*****************************************************}

Begin
 {Código que se emplea para que se puedan hacer los accesos}
 long_reg_TUs:=sizeof(reg_TUs);
 assign(TUs,'f:\user\apartal\catatus.txt');
 rewrite(TUs);
 AbreArchivos;

 {Código que se emplea para el acceso propiamente dicho}
 Estado_bt:= btrv(Primer_Registro,bp_TUs,reg_TUs,long_reg_TUs,
                  LlaveTUs0,0);
 if estado_bt=0 then
   begin
    writeln(TUs,reg_TUs.ClaveTipoUsu+','+reg_tus.DescripTipoUsu);
    repeat
      Estado_bt:= btrv(Registro_Siguiente,bp_TUs,reg_TUs,long_reg_TUs,
                  LlaveTUs0,0);
     if estado_bt=0 then
      writeln(TUs,reg_TUs.ClaveTipoUsu+','+reg_tus.DescripTipoUsu)
     else
      FDA:=True;
     { end if }
    Until FDA;
   end;
 { end if }
 CierraArchivos;
 close(TUs);
End.

Program cancel(input,output,Consulta);
uses
 btr;
Type
 arr2  = array[1..2] of char;
 arr3  = array[1..3] of char;
 arr6  = array[1..6] of char;
 arr29  = array[1..29] of char;
arr128 = array[1..128] of char;
 RegistroApa = Record
               matr_ap : arr6;
               cve_hora_ap : arr2;
               cve_maq_ap : arr3;
               cve_tipo_ap : char;
              end;
 RegistroOcu = Record
               matricula : arr6;
               hora : arr2;
               maquina : arr3;
               Tipo_maq : char;
              end;

var
 Consulta  : Text;
 matricula : arr6;
 reg_apa   : RegistroApa;
 reg_ocu   : RegistroOcu;
 long_reg_apa,
 long_reg_ocu : integer;
 bp_apa,
 bp_ocu    : arr128;
 arch_apa,
 arch_ocu  : arr29;
 llaveApa0,
```

```
  llaveOcu0 : arr6;
  llaveOcu1 : arr3;
  llaveOcu2 : arr2;
  llaveOcu3 : char;


{*******************************************************}
Procedure AbreArchivos;
 begin
  long_reg_apa := sizeof(reg_apa);
  arch_apa := 'f:\user\aparta\aparta.DAT   ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  long_reg_ocu := sizeof(reg_ocu);
  arch_ocu := 'f:\user\aparta\ocupa.DAT   ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
 end;
{ end procedure AbreArchivos}
{*******************************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
 end;
{ end procedure AbreArchivos}

{*******************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*************************************************}

function arreglo_a_cadena( var arreglo_entrada; { Arreglo de entrada }
                           maxcar : byte) : string;
                           { Numero maximo de caracteres del arreglo fuente}
const
 MAX = 74;
var
 cadena_salida : string;
 long : byte absolute cadena_salida;
 arreglo : array[1..MAX] of char absolute arreglo_entrada;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to maxcar do cadena_salida[i] := arreglo[i];
 long := maxcar;

 { Se eliminan los blancos al final de la cadena }

 while (cadena_salida[long] = #32 {ESPACIO}) and (long > 0) do dec(long);
 arreglo_a_cadena := cadena_salida;
end; { arreglo_a_cadena }
{*************************************************}

BEGIN
```

```
{Código que se emplea para que se puedan hacer los accesos}
long_reg_apa:=sizeof(Reg_Apa);
long_reg_ocu:=sizeof(Reg_Ocu);
assign(Consulta,'f:\user\aparta\Consulta.txt');
rewrite(Consulta);
AbreArchivos;

{Código que se emplea para el acceso propiamente dicho}

cadena_a_arreglo(ParamStr(1),matricula,6);
llaveApa0:=matricula;
estado_bt:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,long_reg_apa,
                llaveApa0,0);
If estado_bt = 0 then
  begin
   writeln(Consulta,'0,'+arreglo_a_cadena(reg_apa.cve_maq_ap,3)+','+
                arreglo_a_cadena(reg_apa.cve_hora_ap,2)+':00');
  end
else
  begin
   llaveOcu0:=matricula;
   estado_bt:=btrv(REGISTRO_IGUAL,bp_ocu,reg_ocu,long_reg_ocu,
            llaveOcu0,0);
   If estado_bt = 0 then
     begin
      writeln(Consulta,'0,'+arreglo_a_cadena(reg_ocu.maquina,3)+','+
                arreglo_a_cadena(reg_ocu.hora,2)+':00');
     end
   else
     writeln(consulta,'1,NO TIENE MAQUINA ASIGNADA',',0');
   { end if }
  end;
 { end if }
 Close(Consulta);
 CierraArchivos;
end.

Program ChecaSalida(input,output,checasal);
uses
 btr;
Type
 arr14  = array [1..14] of char;
 arr128 = array [1..128] of char;
 arr20  = array [1..20] of char;
 arr6   = array [1..6] of char;
 arr15  = array [1..15] of char;
 arr3   = array [1..3] of char;
 arr2   = array [1..2] of char;
 arr4   = array [1..4] of char;
 arr5   = array [1..5] of char;
 arr10  = array [1..10] of char;
 arr13  = array [1..13] of char;
 arr29  = array [1..29] of char;
 arr40  = array [1..40] of char;

 RegistroUsu = Record
             matricula : arr6;
             paterno : arr20;
             materno : arr20;
             nombre : arr20;
             tipo : char;
             habil : char;
             status : char;
             sanciones : arr2;
             carrera : arr6;
            end;
 RegistroMaq = Record
             claveMaq : arr3;
             claveTipo : char;
```

```
            serieCPU : arr15;
            serieTeclado :arr15;
            serieMonitor : arr15;
            serieMouse : arr15;
            modelo : arr15;
            RAM : arr3;
            HD : arr4;
            micro : arr6;
            video : arr5;
            FD : arr5;
            servidor : arr2;
            Isla : arr2;
            enServicio : char;
           end;
  RegistroApa = Record
            matr_ap : arr6;
            cve_hora_ap : arr2;
            cve_maq_ap : arr3;
            cve_tipo_ap : char;
           end;
  RegistroOcu = Record
            matricula : arr6;
            hora : arr2;
            maquina : arr3;
            Tipo_maq : char;
           end;
  RegLlaveUsu1 = Record
           paterno : arr20;
           materno : arr20;
           nombre : arr20;
           end;
  RegLlaveUsu2= Record
           paterno  : arr20;
           materno  : arr20;
           nombre   : arr20;
           matricula : arr6;
           end;
  RegLlaveUsu3= Record
           matricula : arr6;
           carrera : arr6;
           end;
  RegLlaveMaq1= Record
           ClaveTipo  : char;
           EnServicio : char;
           end;
  RegLlaveApa1= Record
           clave_Hora_ap : arr2;
           clave_Maq_ap : arr3;
           clave_Tipo_ap : char;
           end;
var
  ChecaSal  : Text;
  matricula,
  arr_matricula : arr6;
  reg_apa   : RegistroApa;
  reg_ocu   : RegistroOcu;
  reg_usu   : RegistroUsu;
  reg_maq   : RegistroMaq;
  clave_hora: arr2;
  long_reg_apa,
  long_reg_ocu,
  long_reg_Usu,
  long_reg_Maq,
  estado_bt1,
  estado_bt2,
  estado_bt3 : integer;
  bp_apa,
  bp_ocu,
  bp_maq,
```

```
{Código que se emplea para que se puedan hacer los accesos}
long_reg_apa:=sizeof(Reg_Apa);
long_reg_ocu:=sizeof(Reg_Ocu);
assign(Consulta,'f:\user\aparta\Consulta.txt');
rewrite(Consulta);
AbreArchivos;

{Código que se emplea para el acceso propiamente dicho}

cadena_a_arreglo(ParamStr(1),matricula,6);
llaveApa0:=matricula;
estado_bt:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,long_reg_apa,
                llaveApa0,0);
If estado_bt = 0 then
 begin
  writeln(Consulta,'0,'+arreglo_a_cadena(reg_apa.cve_maq_ap,3)+','+
                arreglo_a_cadena(reg_apa.cve_hora_ap,2)+':00');
 end
else
 begin
  llaveOcu0:=matricula;
  estado_bt:=btrv(REGISTRO_IGUAL,bp_ocu,reg_ocu,long_reg_ocu,
            llaveOcu0,0);
  If estado_bt = 0 then
   begin
    writeln(Consulta,'0,'+arreglo_a_cadena(reg_ocu.maquina,3)+','+
                arreglo_a_cadena(reg_ocu.hora,2)+':00');
   end
  else
   writeln(consulta,'1,NO TIENE MAQUINA ASIGNADA',',0');
  { end if }
 end;
{ end if }
Close(Consulta);
ClierraArchivos;
end.

Program ChecaSalida(input,output,checasal);
uses
 btr;
Type
 arr14  = array [1..14] of char;
 arr128 = array [1..128] of char;
 arr20  = array [1..20] of char;
 arr6   = array [1..6] of char;
 arr15  = array [1..15] of char;
 arr3   = array [1..3] of char;
 arr2   = array [1..2] of char;
 arr4   = array [1..4] of char;
 arr5   = array [1..5] of char;
 arr10  = array [1..10] of char;
 arr13  = array [1..13] of char;
 arr29  = array [1..29] of char;
 arr40  = array [1..40] of char;

 RegistroUsu = Record
             matricula : arr6;
             paterno : arr20;
             materno : arr20;
             nombre : arr20;
             tipo : char;
             habil : char;
             status : char;
             sanciones : arr2;
             carrera : arr6;
            end;
 RegistroMaq = Record
             claveMaq : arr3;
             claveTipo : char;
```

```
  bp_usu   : arr128;
  arch_apa,
  arch_ocu,
  arch_maq,
  arch_usu  : arr29;
  llaveApa0,
  llaveOcu0 : arr6;
  llaveApa1 : RegLlaveApa1;
  HoraSalida,
  llaveApa2 : arr2;
  llaveApa4 : arr3;
  llaveOcu1 : arr3;
  llaveOcu2 : arr2;
  llaveOcu3 : char;
  llaveUsu0 : arr6;
  llaveUsu1 : RegLlaveUsu1;
  llaveUsu2 : RegLlaveUsu2;
  llaveMaq0 : arr3;
  llaveMaq1 : RegLlaveMaq1;
  FDA       : boolean; {fin de archivo}
  CveTma    : char;

{*******************************************************}
Procedure AbreArchivos;
 begin
  long_reg_apa := sizeof(reg_apa);
  arch_apa := 'f:\user\aparta\aparta.DAT   ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  long_reg_ocu := sizeof(reg_ocu);
  arch_ocu := 'f:\user\aparta\ocupa.DAT   ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
 end;
{ end procedure AbreArchivos}
{*******************************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
 end;
{ end procedure CierraArchivos}

{*******************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 I: byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for I := 1 to long do arreglo[I] := cadena_entrada[I];
 for I := succ(long) to maxcar do arreglo[I] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*******************************************************}

function arreglo_a_cadena( var arreglo_entrada; { Arreglo de entrada }
                           maxcar : byte) : string;
                           { Numero maximo de caracteres del arreglo fuente}
const
 MAX = 74;
var
 cadena_salida : string;
```

```
  long : byte absolute cadena_salida;
  arreglo : array[1..MAX] of char absolute arreglo_entrada;
  i : byte;
begin
  if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
  for i := 1 to maxcar do cadena_salida[i] := arreglo[i];
  long := maxcar;

  { Se eliminan los blancos al final de la cadena }

  while (cadena_salida[long] = #32 {ESPACIO}) and (long > 0) do dec(long);
  arreglo_a_cadena := cadena_salida;
end; { arreglo_a_cadena }

{*****************************************************}

BEGIN
  {Codigo que se emplea para que se puedan hacer los accesos}
  long_reg_apa:=sizeof(Reg_Apa);
  long_reg_ocu:=sizeof(Reg_Ocu);
  AbreArchivos;

  {Codigo que se emplea para el apartado propiamente dicho}
  Cadena_a_Arreglo(ParamStr(1),LlaveApa2,2);
  HoraSalida:=LlaveApa2;
  FDA:=False;
  estado_bt:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,
                  long_reg_apa,LlaveApa2,2);
  if estado_bt=0 then
    begin
      repeat
        estado_bt:=btrv(REGISTRO_IGUAL,bp_ocu,reg_ocu,
                        long_reg_ocu,reg_apa.matr_ap,0);
        if estado_bt=4 then
          estado_bt:=btrv(BORRAR_REGISTRO,bp_apa,reg_apa,
                          long_reg_apa,LlaveApa2,2);
        { end if }
        estado_bt:=btrv(REGISTRO_SIGUIENTE,bp_apa,reg_apa,
                        long_reg_apa,LlaveApa2,2);
        if estado_bt=9 then
          FDA:=True
        else
          if HoraSalida<>reg_Apa.cve_hora_ap then
            FDA:=True
          else
            FDA:=False;
          { end if }
        { end if }
      Until FDA;
      FDA:=False;
    end;
  { end if }
  CierraArchivos;
end.

Program ChecaSalida(input,output,checasal);
uses
  btr,dos;
Type
  arr14  = array [1..14] of char;
  arr128 = array [1..128] of char;
  arr20  = array [1..20] of char;
  arr6   = array [1..6] of char;
  arr15  = array [1..15] of char;
  arr3   = array [1..3] of char;
  arr2   = array [1..2] of char;
  arr4   = array [1..4] of char;
  arr5   = array [1..5] of char;
  arr10  = array [1..10] of char;
```

```
arr13  = array [1..13] of char;
arr29  = array [1..29] of char;
arr40  = array [1..40] of char;

RegistroUsu = Record
            matricula : arr6;
            paterno : arr20;
            materno : arr20;
            nombre : arr20;
            tipo : char;
            habil : char;
            status : char;
            sanciones : arr2;
            carrera : arr6;
           end;
RegistroMaq = Record
            claveMaq : arr3;
            claveTipo : char;
            serieCPU : arr15;
            serieTeclado :arr15;
            serieMonitor : arr15;
            serieMouse : arr15;
            modelo : arr15;
            RAM : arr3;
            HD : arr4;
            micro : arr6;
            video : arr5;
            FD : arr5;
            servidor : arr2;
            isla : arr2;
            enServicio : char;
           end;
RegistroApa = Record
            matr_ap : arr6;
            cve_hora_ap : arr2;
            cve_maq_ap : arr3;
            cve_tipo_ap : char;
           end;
RegistroOcu = Record
            matricula : arr6;
            hora : arr2;
            maquina : arr3;
            Tipo_maq : char;
           end;
RegLlaveUsu1= Record
            paterno : arr20;
            materno : arr20;
            nombre : arr20;
            end;
RegLlaveUsu2= Record
            paterno  : arr20;
            materno  : arr20;
            nombre   : arr20;
            matricula : arr6;
            end;
RegLlaveUsu3= Record
            matricula : arr6;
            carrera : arr6;
            end;
RegLlaveMaq1= Record
            ClaveTipo  : char;
            EnServicio : char;
            end;
RegLlaveApa1= Record
            clave_Hora_ap : arr2;
            clave_Maq_ap : arr3;
            clave_Tipo_ap : char;
            end;
Var
```

```
  Aux1,
  Aux2,
  Aux3     : string;
  EstMaq   : Text;
  matricula,
  arr_matricula : arr6;
  reg_apa  : RegistroApa;
  reg_ocu  : RegistroOcu;
  reg_usu  : RegistroUsu;
  reg_maq  : RegistroMaq;
  year,month,
  day,dow  : word;
  year_arr : arr4;
  month_arr,
  day_arr  : arr2;
  clave_hora: arr2;
  long_reg_apa,
  long_reg_ocu,
  long_reg_Usu,
  long_reg_Maq,
  estado_bt1,
  estado_bt2,
  estado_bt3,
  cont_maq,
  codigo : integer;
  bp_apa,
  bp_ocu,
  bp_maq,
  bp_usu   : arr128;
  arch_apa,
  arch_ocu,
  arch_maq,
  arch_usu  : arr29;
  llaveApa0,
  llaveOcu0 : arr6;
  cont_arr  : arr3;
  llaveApa1 : RegLlaveApa1;
  dow_arr   : char;
  HoraSalida,
  llaveApa2 : arr2;
  llaveApa4 : arr3;
  llaveOcu1 : arr3;
  llaveOcu2 : arr2;
  llaveOcu3 : char;
  llaveUsu0 : arr6;
  llaveUsu1 : RegLlaveUsu1;
  llaveUsu2 : RegLlaveUsu2;
  llaveMaq0 : arr3;
  llaveMaq1 : RegLlaveMaq1;
  FDA      : boolean; {fin de archivo}
  CveTma   : char;

{*********************************************}
Procedure AbreArchivos;
 begin
  long_reg_apa := sizeof(reg_apa);
  arch_apa := 'f:\user\aparta\aparta.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  long_reg_ocu := sizeof(reg_ocu);
  arch_ocu := 'f:\user\aparta\ocupa.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
  long_reg_maq := sizeof(reg_maq);
  arch_maq := 'f:\user\aparta\maquina.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_maq,reg_maq,long_reg_maq,arch_maq,0);

 end;
{ end procedure AbreArchivos}
{*********************************************}
```

```
Procedure CierraArchivos;
 begin
  estado_bt:=blrv(CERRAR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  estado_bt:=blrv(CERRAR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
  estado_bt:=blrv(CERRAR_ARCHIVO,bp_maq,reg_maq,long_reg_maq,arch_maq,0);
 end;
{ end procedure CierraArchivos}

{*****************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*****************************************************}
procedure cadena_a_arreglo1( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := '0';
end;
{ cadena_a_arreglo }

{*****************************************************}

function arreglo_a_cadena( var arreglo_entrada; { Arreglo de entrada }
                           maxcar : byte) : string;
                           { Numero maximo de caracteres del arreglo fuente}
const
 MAX = 74;
var
 cadena_salida : string;
 long : byte absolute cadena_salida;
 arreglo : array[1..MAX] of char absolute arreglo_entrada;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to maxcar do cadena_salida[i] := arreglo[i];
 long := maxcar;

 { Se eliminan los blancos al final de la cadena }

 while (cadena_salida[long] = #32 {ESPACIO}) and (long > 0) do dec(long);
 arreglo_a_cadena := cadena_salida;
end; { arreglo_a_cadena }

{*****************************************************}

procedure entero_a_arreglo(num: longint; var arreglo; maxcar:byte);
```

```
 { Convierte un numero entero a un arreglo de maxcar caracteres }
var
 arr : array [1..11] of char absolute arreglo;
 cadnum : string[11];
 i : byte;
begin
 str(num:maxcar,cadnum);
 for i := 1 to maxcar do
 begin
  if cadnum[i] = ' ' then
   arr[i] := '0'
  else
   arr[i] := cadnum[i]
 end
end; { entero_a_arreglo }


{*****************************************************}

BEGIN
 {Codigo que se emplea para que se puedan hacer los accesos}
 long_reg_apa:=sizeof(Reg_Apa);
 long_reg_ocu:=sizeof(Reg_Ocu);
 long_reg_maq:=sizeof(Reg_Maq);
 AbreArchivos;

 {Codigo que se emplea para el acceso propiamente dicho}
 Cadena_a_Arreglo1(ParamStr(1),LlaveApa2,2);
 { busca en aparta por apartados a ,sta hora para borrarlos }
 HoraSalida:=LlaveApa2;
 estado_bt:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,
          long_reg_apa,llaveApa2,2);
 if estado_bt=0 then
  begin
   repeat
    estado_bt:=btrv(BORRAR_REGISTRO,bp_apa,reg_apa,
             long_reg_apa,llaveApa2,2);
    if estado_bt=0 then
     begin
      estado_bt:=btrv(REGISTRO_SIGUIENTE,bp_apa,reg_apa,
               long_reg_apa,llaveApa2,2);
      if estado_bt=9 then
       FDA:=True
      else
       if reg_apa.cve_hora_ap<>HoraSalida then
        FDA:=True
       else
        FDA:=False;
       { end if }
      { end if }
     end;
   { end if }
   Until FDA;
   FDA:=False;
  end;
 { end if }
 { ********** salida en ocupa ********** }
 { ***** rutina borra informacion de ocupa ***** }
 estado_bt:=btrv(PRIMER_REGISTRO,bp_ocu,reg_ocu,
          long_reg_ocu,LlaveOcu0,0);
 if estado_bt = 0 then
  begin
   repeat
    FDA:=False;
    estado_bt:=btrv(BORRAR_REGISTRO,bp_ocu,reg_ocu,
             long_reg_ocu,LlaveOcu0,0);
    if estado_bt=0 then
     begin
      estado_bt:=btrv(Registro_Siguiente,bp_ocu,
```

```
                reg_ocu,long_reg_ocu,LlaveOcu0,0);
        if estado_bt = 9 then
          begin
            FDA:=True;
          end;
        {end if}
      end;
    { end if }
  until FDA;
  FDA:=False;
 end;
{end if}

{ Inicia el contador de m quinas}
{ pide la fecha del sistema }

GetDate(year,month,day,dow);
str(year,aux1);
cadena_a_arreglo1(aux1,year_arr,4);
{ entero_a_arreglo(year,year_arr,4);}
str(month,aux2);
cadena_a_arreglo1(aux2,month_arr,2);
{ entero_a_arreglo(month,month,2);   }
str(day,aux3);
cadena_a_arreglo1(aux3,day_arr,2);
{ entero_a_arreglo(day,day_arr,2);}
cont_maq:=0;
estado_bt:=btrv(Primer_Registro,bp_maq,reg_maq,
          long_reg_maq,Llavemaq1,1);
if estado_bt = 0 then
 begin
  if (reg_maq.enServicio='S') then
    inc(cont_maq);
  {end if}
  repeat
    FDA:=False;
    estado_bt:=btrv(Registro_Siguiente,bp_maq,
              reg_maq,long_reg_maq,Llavemaq1,1);
    if estado_bt = 0 then
      begin
        if (reg_maq.enServicio='S') then
          inc(cont_maq);
        {end if}
      end
      else
        if estado_bt = 9 then
          begin
            FDA:=True;
          end;
        {end if}
      {end if}
    until FDA;
 end
 else
  begin
   cont_maq:=0;
  end;
{end if}
assign(EstMaq,'f:\user\aparla\'+day_arr+month_arr+year_arr[3]+year_arr[4]+'_O.txt');
entero_a_arreglo(cont_maq,cont_arr,3);
{checa primera hora entresemana}
if (HoraSalida='07') and (dow>=1) and (dow<6) then
 begin
  rewrite(EstMaq);
  writeln(EstMaq,HoraSalida+','+cont_arr);
 end
else
 {checa primera hora s bado}
 if (HoraSalida='08') and (dow=6) then
```

```
    begin
      rewrite(EstMaq);
      writeln(EstMaq,'07,0');
      writeln(EstMaq,HoraSalida+','+cont_arr);
    end
   else
     begin
       Append(EstMaq);
       writeln(EstMaq,HoraSalida+','+cont_arr);
     end;
  { end if }
 { end if }
 CierraArchivos;
 Close(EstMaq);
end.

Program cancel(input,output,entra);
uses
 btr;
Type
 arr14  = array [1..14] of char;
 arr128 = array [1..128] of char;
 arr20  = array [1..20] of char;
 arr6   = array [1..6] of char;
 arr15  = array [1..15] of char;
 arr3   = array [1..3] of char;
 arr2   = array [1..2] of char;
 arr4   = array [1..4] of char;
 arr5   = array [1..5] of char;
 arr10  = array [1..10] of char;
 arr13  = array [1..13] of char;
 arr29  = array [1..29] of char;
 arr40  = array [1..40] of char;

 RegistroUsu = Record
               matricula : arr6;
               paterno : arr20;
               materno : arr20;
               nombre : arr20;
               tipo : char;
               habil : char;
               status : char;
               sanciones : arr2;
               carrera : arr6;
             end;
 RegistroMaq = Record
               claveMaq : arr3;
               claveTipo : char;
               serieCPU : arr15;
               serieTeclado :arr15;
               serieMonitor : arr15;
               serieMouse : arr15;
               modelo : arr15;
               RAM : arr3;
               HD : arr4;
               micro : arr6;
               video : arr5;
               FD : arr5;
               servidor : arr2;
               isla : arr2;
               enServicio : char;
             end;
 RegistroApa = Record
               matr_ap : arr6;
               cve_hora_ap : arr2;
               cve_maq_ap : arr3;
               cve_tipo_ap : char;
             end;
 RegistroOcu = Record
```

```
            matricula : arr6;
            hora : arr2;
            maquina : arr3;
            Tipo_maq : char;
           end;
  RegLlaveUsu1= Record
           paterno : arr20;
           materno : arr20;
           nombre : arr20;
           end;
  RegLlaveUsu2= Record
           paterno  : arr20;
           materno  : arr20;
           nombre   : arr20;
           matricula : arr6;
           end;
  RegLlaveUsu3= Record
           matricula : arr6;
           carrera : arr6;
           end;
  RegLlaveMaq1= Record
           ClaveTipo  : char;
           EnServicio : char;
           end;
  RegLlaveApa1= Record
           clave_Hora_ap : arr2;
           clave_Maq_ap : arr3;
           clave_Tipo_ap : char;
           end;

var
  Entra1    : Text;
  matricula,
  arr_matricula : arr6;
  reg_apa   : RegistroApa;
  reg_ocu   : RegistroOcu;
  reg_usu   : RegistroUsu;
  reg_maq   : RegistroMaq;
  clave_hora: arr2;
  long_reg_apa,
  long_reg_ocu,
  long_reg_Usu,
  long_reg_Maq,
  estado_bt1,
  estado_bt2,
  estado_bt3 : Integer;
  bp_apa,
  bp_ocu,
  bp_maq,
  bp_usu    : arr128;
  arch_apa,
  arch_ocu,
  arch_maq,
  arch_usu  : arr29;
  llaveApa0,
  llaveOcu0 : arr6;
  llaveApa1 : RegLlaveApa1;
  llaveApa2 : arr2;
  llaveApa4 : arr3;
  llaveOcu1 : arr3;
  llaveOcu2 : arr2;
  llaveOcu3 : char;
  llaveUsu0 : arr6;
  llaveUsu1 : RegLlaveUsu1;
  llaveUsu2 : RegLlaveUsu2;
  llaveMaq0 : arr3;
  llaveMaq1 : RegLlaveMaq1;
  FDA       : boolean; {fin de archivo}
  CveTma    : char;
```

```
{*********************************************}
Procedure AbreArchivos;
 begin
  long_reg_apa := sizeof(reg_apa);
  arch_apa := 'f:\user\aparta\aparta.DAT   ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  long_reg_ocu := sizeof(reg_ocu);
  arch_ocu := 'f:\user\aparta\ocupa.DAT   ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
  long_reg_usu := sizeof(reg_usu);
  arch_usu := 'f:\user\aparta\usuario.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_usu,reg_usu,long_reg_usu,arch_usu,0);
 end;
{ end procedure AbreArchivos}
{*********************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_usu,reg_usu,long_reg_usu,arch_usu,0);
 end;
{ end procedure AbreArchivos}

{*********************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                var arreglo_salida; { Arreglo de salida }
                maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*********************************************}

function arreglo_a_cadena( var arreglo_entrada; { Arreglo de entrada }
                maxcar : byte) : string;
                { Numero maximo de caracteres del arreglo fuente}
const
 MAX = 74;
var
 cadena_salida : string;
 long : byte absolute cadena_salida;
 arreglo : array[1..MAX] of char absolute arreglo_entrada;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to maxcar do cadena_salida[i] := arreglo[i];
 long := maxcar;

 { Se eliminan los blancos al final de la cadena }

 while (cadena_salida[long] = #32 {ESPACIO}) and (long > 0) do dec(long);
 arreglo_a_cadena := cadena_salida;
end; { arreglo_a_cadena }

{*********************************************}
```

```
BEGIN
 {Codigo que se emplea para que se puedan hacer los accesos}
 long_reg_apa:=sizeof(Reg_Apa);
 long_reg_ocu:=sizeof(Reg_Ocu);
 long_reg_usu:=sizeof(Reg_Usu);
 assign(Entra1,'f:\user\aparta\entra1.txt');
 rewrite(Entra1);
 AbreArchivos;

 {Codigo que se emplea para el apartado propiamente dicho}
 Cadena_a_Arreglo(ParamStr(1),arr_matricula,6);
 Cadena_a_Arreglo(ParamStr(2),clave_hora,2);
 LlaveUsu0:=arr_matricula;

 estado_bt:=btrv(REGISTRO_IGUAL,bp_usu,reg_usu,
          long_reg_usu,arr_matricula,0);
 if (estado_bt=0) and (reg_usu.habil='S') then
  begin
   estado_bt:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,
            long_reg_apa,arr_matricula,0);
   if (estado_bt=0) then
    if reg_apa.cve_hora_ap=clave_hora then
     begin
      estado_bt:=btrv(INSERTAR_REGISTRO,bp_ocu,reg_apa,
               long_reg_ocu,LlaveOcu0,0);
      if estado_bt=0 then
       Writeln(entra1,'0,'+arreglo_a_cadena(reg_apa.cve_maq_ap,3))
       else
       Writeln(entra1,'1,IMPOSIBLE DAR ACCESO')
      { end if }
     end
    else
     { ya tiene apartado a otra hora }
     writeln(entra1,'2,EL USUARIO TIENE APARTADO A OTRA HORA')
    { end if }
   else
    writeln(entra1,'3,0');
   { end if }
  end
 else
  if (estado_bt=0) and (reg_usu.habil='N') then
   Writeln(entra1,'4,EL USUARIO ESTA DESHABILITADO!')
  else
   writeln(entra1,'5,USUARIO NO DADO DE ALTA!');
  { end if }
 { end if }
 Close(Entra1);
 CierraArchivos;
end.

Program cancel(input,output,entra);
uses
 btr;
Type
 arr14  = array [1..14] of char;
 arr128 = array [1..128] of char;
 arr20  = array [1..20] of char;
 arr6   = array [1..6] of char;
 arr15  = array [1..15] of char;
 arr3   = array [1..3] of char;
 arr2   = array [1..2] of char;
 arr4   = array [1..4] of char;
 arr5   = array [1..5] of char;
 arr10  = array [1..10] of char;
 arr13  = array [1..13] of char;
 arr29  = array [1..29] of char;
 arr40  = array [1..40] of char;

 RegistroUsu = Record
```

```
            matricula : arr6;
            paterno : arr20;
            materno : arr20;
            nombre : arr20;
            tipo : char;
            habil : char;
            status : char;
            sanciones : arr2;
            carrera : arr6;
          end;
  RegistroMaq = Record
            claveMaq : arr3;
            claveTipo : char;
            serieCPU : arr15;
            serieTeclado :arr15;
            serieMonitor : arr15;
            serieMouse : arr15;
            modelo : arr15;
            RAM : arr3;
            HD : arr4;
            micro : arr6;
            video : arr5;
            FD : arr5;
            servidor : arr2;
            isla : arr2;
            enServicio : char;
          end;
  RegistroApa = Record
            matr_ap : arr6;
            cve_hora_ap : arr2;
            cve_maq_ap : arr3;
            cve_tipo_ap : char;
          end;
  RegistroOcu = Record
            matricula : arr6;
            hora : arr2;
            maquina : arr3;
            Tipo_maq : char;
          end;
  RegLlaveUsu1 = Record
            paterno : arr20;
            materno : arr20;
            nombre : arr20;
            end;
  RegLlaveUsu2= Record
            paterno  : arr20;
            materno  : arr20;
            nombre   : arr20;
            matricula : arr6;
            end;
  RegLlaveUsu3= Record
            matricula : arr6;
            carrera : arr6;
            end;
  RegLlaveMaq1= Record
            ClaveTipo  : char;
            EnServicio : char;
          end;
  RegLlaveApa1= Record
            clave_Hora_ap : arr2;
            clave_Maq_ap : arr3;
            clave_Tipo_ap : char;
            end;

var
  Entra2   : Text;
  matricula,
  arr_matricula : arr6;
  reg_apa   : RegistroApa;
```

```
  reg_ocu   : RegistroOcu;
  reg_usu   : RegistroUsu;
  reg_maq   : RegistroMaq;
  clave_hora: arr2;
  long_reg_apa,
  long_reg_ocu,
  long_reg_Usu,
  long_reg_Maq,
  estado_bt1,
  estado_bt2,
  estado_bt3 : integer;
  bp_apa,
  bp_ocu,
  bp_maq,
  bp_usu    : arr128;
  arch_apa,
  arch_ocu,
  arch_maq,
  arch_usu  : arr29;
  llaveApa0,
  llaveOcu0 : arr6;
  llaveApa1 : RegLlaveApa1;
  llaveApa2 : arr2;
  llaveApa4 : arr3;
  llaveOcu1 : arr3;
  llaveOcu2 : arr2;
  llaveOcu3 : char;
  llaveUsu0 : arr6;
  llaveUsu1 : RegLlaveUsu1;
  llaveUsu2 : RegLlaveUsu2;
  llaveMaq0 : arr3;
  llaveMaq1 : RegLlaveMaq1;
  FDA       : boolean; {fin de archivo}
  Clave_Tipo: char;

{*****************************************************}
Procedure AbreArchivos;
  begin
    long_reg_apa := sizeof(reg_apa);
    arch_apa := 'f:\user\aparta\aparta.DAT   ';
    estado_bt:=btrv(ABRIR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
    long_reg_ocu := sizeof(reg_ocu);
    arch_ocu := 'f:\user\aparta\ocupa.DAT    ';
    estado_bt:=btrv(ABRIR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
    long_reg_usu := sizeof(reg_usu);
    arch_usu := 'f:\user\aparta\usuario.DAT  ';
    estado_bt:=btrv(ABRIR_ARCHIVO,bp_usu,reg_usu,long_reg_usu,arch_usu,0);
    long_reg_maq := sizeof(reg_maq);
    arch_maq := 'f:\user\aparta\maquina.DAT  ';
    estado_bt:=btrv(ABRIR_ARCHIVO,bp_maq,reg_maq,long_reg_maq,arch_maq,0);

  end;
{ end procedure AbreArchivos}
{*****************************************************}

Procedure CierraArchivos;
  begin
    estado_bt:=btrv(CERRAR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
    estado_bt:=btrv(CERRAR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
    estado_bt:=btrv(CERRAR_ARCHIVO,bp_usu,reg_usu,long_reg_usu,arch_usu,0);
    estado_bt:=btrv(CERRAR_ARCHIVO,bp_maq,reg_maq,long_reg_maq,arch_maq,0);
  end;
{ end procedure AbreArchivos}

{*****************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
```

```
const
  MAX = 74;
var
  long : byte absolute cadena_entrada;
  arreglo : array[1..MAX] of char absolute arreglo_salida;
  i : byte;
begin
  if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
  for i := 1 to long do arreglo[i] := cadena_entrada[i];
  for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*******************************************************}

function arreglo_a_cadena(  var arreglo_entrada; { Arreglo de entrada }
                            maxcar : byte) : string;
                            { Numero maximo de caracteres del arreglo fuente}
const
  MAX = 74;
var
  cadena_salida : string;
  long : byte absolute cadena_salida;
  arreglo : array[1..MAX] of char absolute arreglo_entrada;
  i : byte;
begin
  if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
  for i := 1 to maxcar do cadena_salida[i] := arreglo[i];
  long := maxcar;

  { Se eliminan los blancos al final de la cadena }

  while (cadena_salida[long] = #32 {ESPACIO}) and (long > 0) do dec(long);
  arreglo_a_cadena := cadena_salida;
end; { arreglo_a_cadena }


{*******************************************************}

BEGIN
  {Codigo que se emplea para que se puedan hacer los accesos}
  long_reg_apa:=sizeof(Reg_Apa);
  long_reg_ocu:=sizeof(Reg_Ocu);
  long_reg_usu:=sizeof(Reg_Usu);
  long_reg_maq:=sizeof(Reg_Maq);
  assign(Entra2,'f:\user\aparta\entra2.txt');
  rewrite(Entra2);
  AbreArchivos;

  {Codigo que se emplea para el apartado propiamente dicho}
  Cadena_a_Arreglo(ParamStr(1),arr_matricula,6);
  Cadena_a_Arreglo(ParamStr(2),clave_hora,2);
  Cadena_a_Arreglo(ParamStr(3),clave_tipo,1);
  LlaveUsu0:=arr_matricula;

  estado_bt:=btrv(REGISTRO_IGUAL,bp_usu,reg_usu,
            long_reg_usu,arr_matricula,0);
  if (estado_bt=0) and (reg_usu.habil='S') then
    {EL USUARIO ENTRARA SIN APARTADO}
    begin
      LlaveMaq1.ClaveTipo:=Clave_Tipo;
      LlaveMaq1.EnServicio:='S';
      Estado_bt:=btrv(REGISTRO_IGUAL,bp_maq,reg_maq,
                long_reg_maq,LlaveMaq1,1);
      if estado_bt<>0 Then
        Writeln(entra2,'1,NO EXISTE ESE TIPO DE MAQUINA')
      else
        begin
          repeat
```

```
    FDA:=False;
    with llaveApa1 do
      begin
       clave_hora_ap:=Clave_Hora;
       clave_maq_ap:=Reg_Maq.ClaveMaq;
       clave_tipo_ap:=Reg_Maq.claveTipo;
      end;
    { end with }
    { verifica si la m quina no est  apartada }
    estado_bt:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,
              long_reg_apa,LlaveApa1,1);
    { verifica que la m quina no est, ocupada }
    estado_bt1:=btrv(REGISTRO_IGUAL,bp_ocu,reg_ocu,
              long_reg_ocu,Reg_Maq.ClaveMaq,1);
    if (estado_bt=0) or (estado_bt1=0) Then
     {estaba ocupada o apartada}
     begin
      estado_bt:=btrv(REGISTRO_SIGUIENTE,bp_maq,
              reg_maq,long_reg_maq,LlaveMaq1,1);
      if estado_bt=9 then {ya se acabo el archivo}
       begin
        FDA:=True;
        writeln(entra2,'NO HAY MAQUINAS DISPONIBLES');
       end
      else
       begin
       if reg_maq.ClaveTipo<>Clave_Tipo Then
        begin
         FDA:=True;
         writeln(entra2,'NO HAY MAQUINAS DISPONIBLES');
        end;
       { end if }
       end;
      { end if }
      estado_bt:=99;
     end
    else
     begin
      with reg_ocu do
       begin
        matricula:=arr_matricula;
        hora:=Clave_Hora;
        maquina:=Reg_Maq.ClaveMaq;
        Tipo_maq:=Reg_Maq.claveTipo;
       end;
      { end with }
      estado_bt:=btrv(INSERTAR_REGISTRO,bp_ocu,reg_ocu,
              long_reg_ocu,LlaveOcu0,0);
      if estado_bt<>0 Then
       begin
        if estado_bt=5 then
         Writeln(ENTRA2,'2,EL USUARIO ESTA DENTRO DEL '+
                 'C.E.C.')
        else
         WRITELN(ENTRA2,'3,IMPOSIBLE DAR ACCESO, REINTENTE');
        { end if }
        FDA:=True;
       end
      else
       writeln(entra2,'0,'+arreglo_a_cadena(reg_ocu.maquina,3));
      { end if }
     end;
    { end if }
   Until FDA or (Estado_bt=0);
   end;
  { end if }
 end
else
 writeln(entra2,'3,EL USUARIO ESTA DESHABILITADO O NO EXISTE');
```

```
 { end if }
  Close(Entra2);
  CierraArchivos;
end.

Program Maquina(Input,Output,Maq);
uses
  btr;

{*****************************************************}
Type
 arr29  = array [1..29] of char;   {para el nombre del archivo btrieve}
 arr128 = array [1..128] of char;  {para el databuffer}
 arr3   = array [1..3] of char;    {para clavemaq y RAM}
 arr15  = array [1..15] of char;   {para serieCPU, serieTeclado,
                                    serieMonitor, serieMouse, serieTeclado,
                                    modelo}
 arr5   = array [1..5] of char;    {para video}
 arr4   = array [1..4] of char;    {para HD}
 arr6   = array [1..6] of char;    {para microprocesador}
 arr2   = array [1..2] of char;    {para servidor e isla}
 RegistroMaq = Record              {registro del archivo de usuarios}
          claveMaq        : arr3;
          claveTipo       : char;
          serieCPU        : arr15;
          serieTeclado    : arr15;
          serieMonitor    : arr15;
          serieMouse      : arr15;
          modelo          : arr15;
          RAM             : arr3;
          HD              : arr4;
          microprocesador : arr6;
          video           : arr5;
          FD              : arr5;
          servidor        : arr2;
          isla            : arr2;
          EnServicio      : char;
          end;
var
 estado_bt,
 long_reg_maq,
 codigo      : integer;
 aborto      : boolean;
 bp_maq      : arr128;
 reg_maq     : RegistroMaq;
 llaveMaq0   : arr2;
 arch_maq    : arr29;
 Maq         : Text;
 Operacion   : byte;
{*****************************************************}
Procedure AbreArchivos;
 begin
  long_reg_maq := sizeof(reg_maq);
  arch_maq := 'C:\TESIS\MAQUINA.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_maq,reg_maq,long_reg_maq,arch_maq,0);
 end;
{ end procedure AbreArchivos}
{*****************************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_maq,reg_maq,long_reg_maq,arch_maq,0);
 end;
{ end procedure AbreArchivos}

{*****************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                    var arreglo_salida; { Arreglo de salida }
```

```
                maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*****************************************************}

Begin
 {Código que se emplea para que se puedan hacer los accesos}
 long_reg_maq:=sizeof(reg_maq);
 assign(Maq,'C:\TESIS\maq.txt');
 rewrite(Maq);
 AbreArchivos;


 {Código que se emplea para el acceso propiamente dicho}
 Val(ParamStr(1),Operacion,codigo);
 Case Operacion of
  1: Begin
      cadena_a_arreglo(ParamStr(2),reg_maq.claveMaq,3);
      cadena_a_arreglo(ParamStr(3),reg_maq.claveTipo,1);
      cadena_a_arreglo(ParamStr(4),reg_maq.serieCPU,15);
      cadena_a_arreglo(ParamStr(5),reg_maq.serieTeclado,15);
      cadena_a_arreglo(ParamStr(6),reg_maq.serieMonitor,15);
      cadena_a_arreglo(ParamStr(7),reg_maq.serieMouse,15);
      cadena_a_arreglo(ParamStr(8),reg_maq.modelo,15);
      cadena_a_arreglo(ParamStr(9),reg_maq.RAM,3);
      cadena_a_arreglo(ParamStr(10),reg_maq.HD,4);
      cadena_a_arreglo(ParamStr(11),reg_maq.microprocesador,6);
      cadena_a_arreglo(ParamStr(12),reg_maq.video,5);
      cadena_a_arreglo(ParamStr(13),reg_maq.FD,5);
      cadena_a_arreglo(ParamStr(14),reg_maq.servidor,2);
      cadena_a_arreglo(ParamStr(15),reg_maq.isla,2);
      cadena_a_arreglo(ParamStr(16),reg_maq.EnServicio,1);
      Estado_bt:= btrv(INSERTAR_REGISTRO,bp_maq,reg_maq,long_reg_maq,
                  llaveMaq0,0);
      case estado_bt of
       0: writeln(Maq,'0,ALTA EXITOSA');
       5: writeln(Maq,'5,LA MAQUINA YA ESTA DADA DE ALTA');
      else
       writeln(Maq,'1,IMPOSIBLE HACER ALTA');
      end; { Case }
     End;

  2: Begin
      cadena_a_arreglo(ParamStr(2),llaveMaq0,3);
      Estado_bt:= btrv(REGISTRO_IGUAL,bp_maq,reg_maq,long_reg_maq,
                  llaveMaq0,0);
      If estado_bt=0 then
       begin
        Estado_bt:= btrv(BORRAR_REGISTRO,bp_maq,reg_maq,long_reg_maq,
                    llaveMaq0,0);
        If estado_bt=0 then
         Writeln(Maq,'0,MAQUINA BORRADA')
        else
         Writeln(Maq,'1,IMPOSIBLE HACER BAJA');
        { End If }
       End
      Else
       Writeln(Maq,'4,NO EXISTE ESTA MAQUINA');
```

```
    { End If }
   End;

 3: Begin
    cadena_a_arreglo(ParamStr(2),llaveMaq0,3);
    Estado_bt:= btrv(REGISTRO_IGUAL,bp_maq,reg_maq,long_reg_maq,
                llaveMaq0,0);
    If estado_bt=0 then
     begin
      cadena_a_arreglo(ParamStr(2),reg_maq.claveMaq,3);
      cadena_a_arreglo(ParamStr(3),reg_maq.claveTipo,1);
      cadena_a_arreglo(ParamStr(4),reg_maq.serieCPU,15);
      cadena_a_arreglo(ParamStr(5),reg_maq.serieTeclado,15);
      cadena_a_arreglo(ParamStr(6),reg_maq.serieMonitor,15);
      cadena_a_arreglo(ParamStr(7),reg_maq.serieMouse,15);
      cadena_a_arreglo(ParamStr(8),reg_maq.modelo,15);
      cadena_a_arreglo(ParamStr(9),reg_maq.RAM,3);
      cadena_a_arreglo(ParamStr(10),reg_maq.HD,4);
      cadena_a_arreglo(ParamStr(11),reg_maq.microprocesador,6);
      cadena_a_arreglo(ParamStr(12),reg_maq.video,5);
      cadena_a_arreglo(ParamStr(13),reg_maq.FD,5);
      cadena_a_arreglo(ParamStr(14),reg_maq.servidor,2);
      cadena_a_arreglo(ParamStr(15),reg_maq.isla,2);
      cadena_a_arreglo(ParamStr(16),reg_maq.EnServicio,1);
      Estado_bt:= btrv(ACTUALIZAR_REGISTRO,bp_maq,reg_maq,long_reg_maq,
                  llaveMaq0,0);
      if estado_bt=5 then
       Writeln(Maq,'5,YA ESTA DADA DE ALTA LA MAQUINA')
      else
       if estado_bt=0 then
        Writeln(Maq,'0,ACTUALIZACION EXITOSA')
       else
        Writeln(Maq,'1,IMPOSIBLE HACER ACTUALIZACION');
       { end if }
      { end if }
     end
     Else
      Writeln(Maq,'4,NO EXISTE ESTA MAQUINA');
     { End If }
    End;

 4: Begin
    cadena_a_arreglo(ParamStr(2),llaveMaq0,3);
    Estado_bt:= btrv(REGISTRO_IGUAL,bp_maq,reg_maq,long_reg_maq,
                llaveMaq0,0);
    If estado_bt=0 then
     Writeln(Maq,'0,'+reg_maq.claveMaq+','+
           reg_maq.claveTipo+','+
           reg_maq.serieCPU+','+
           reg_maq.serieTeclado+','+
           reg_maq.serieMonitor+','+
           reg_maq.serieMouse+','+
           reg_maq.modelo+','+
           reg_maq.RAM+','+
           reg_maq.HD+','+
           reg_maq.microprocesador+','+
           reg_maq.video+','+
           reg_maq.FD+','+
           reg_maq.servidor+','+
           reg_maq.isla+','+
           reg_maq.EnServicio)
    Else
     Writeln(Maq,'4,NO EXISTE ESTA MAQUINA'+',0,0,0,0,0,0,0,0,0,0,0,0,0,0');
    { End If }
   End;
 End; { Case }
 CierraArchivos;
 close(Maq);
End.
```

```
{*****************************************************}
procedure GeneraRespaldo;
var
  confirma : char;
  dire     : string[8];
begin
  dire:='        ';
  HazVentana1(16,9,66,15,'');
  Write('      Desea Hacer Respaldo a Disquette ?   ');
  Confirma:=SINo1('');
  BorraVentana;
  if confirma='S' then
    begin
      MensajeError('Asegurese de tener el disco en el drive a:');
      CierraArchivos;
      GetDate(year,month,day,dow);
      dire:=num_cad_ceros(year,4)+num_cad_ceros(month,2)+num_cad_ceros(day,2);
{*************** respaldo a disco duro **************}
      ProgramName2:='md  c:\'+dire;
      if ProgramName2 <> '' then
        begin
          ProgramName2 := '/C' + ProgramName2;
          SwapVectors;
          Exec(GetEnv('COMSPEC'),ProgramName2);
          SwapVectors;
        end;
      { end if }
      ProgramName2:='copy est*.dat c:\'+dire+'>null';
      if ProgramName2 <> '' then
        begin
          ProgramName2 := '/C' + ProgramName2;
          SwapVectors;
          Exec(GetEnv('COMSPEC'),ProgramName2);
          SwapVectors;
        end;
      { end if }
      ProgramName2:='copy lunes.dat c:\'+dire+'>null';
      if ProgramName2 <> '' then
        begin
          ProgramName2 := '/C' + ProgramName2;
          SwapVectors;
          Exec(GetEnv('COMSPEC'),ProgramName2);
          SwapVectors;
        end;
      { end if }
      ProgramName2:='copy marles.dat c:\'+dire+'>null';
      if ProgramName2 <> '' then
        begin
          ProgramName2 := '/C' + ProgramName2;
          SwapVectors;
          Exec(GetEnv('COMSPEC'),ProgramName2);
          SwapVectors;
        end;
      { end if }
      ProgramName2:='copy miercole.dat c:\'+dire+'>null';
      if ProgramName2 <> '' then
        begin
          ProgramName2 := '/C' + ProgramName2;
          SwapVectors;
          Exec(GetEnv('COMSPEC'),ProgramName2);
          SwapVectors;
        end;
      { end if }
      ProgramName2:='copy jueves.dat c:\'+dire+'>null';
      if ProgramName2 <> '' then
        begin
          ProgramName2 := '/C' + ProgramName2;
```

```
        SwapVectors;
        Exec(GetEnv('COMSPEC'),ProgramName2);
        SwapVectors;
      end;
    { end if }
    ProgramName2:='copy viernes.dat c:\'+dire+'>null';
    if ProgramName2 <> '' then
      begin
        ProgramName2 := '/C' + ProgramName2;
        SwapVectors;
        Exec(GetEnv('COMSPEC'),ProgramName2);
        SwapVectors;
      end;
    { end if }
    ProgramName2:='copy sabado.dat c:\'+dire+'>null';
    if ProgramName2 <> '' then
      begin
        ProgramName2 := '/C' + ProgramName2;
        SwapVectors;
        Exec(GetEnv('COMSPEC'),ProgramName2);
        SwapVectors;
      end;
    { end if }
{*************** /respaldo a drive A ************}
    ProgramName2:='copy est*.dat a: >null';
    if ProgramName2 <> '' then
      begin
        ProgramName2 := '/C' + ProgramName2;
        SwapVectors;
        Exec(GetEnv('COMSPEC'),ProgramName2);
        SwapVectors;
      end;
    { end if }
    ProgramName2:='copy lunes.dat a: >null';
    if ProgramName2 <> '' then
      begin
        ProgramName2 := '/C' + ProgramName2;
        SwapVectors;
        Exec(GetEnv('COMSPEC'),ProgramName2);
        SwapVectors;
      end;
    { end if }
    ProgramName2:='copy martes.dat a: >null';
    if ProgramName2 <> '' then
      begin
        ProgramName2 := '/C' + ProgramName2;
        SwapVectors;
        Exec(GetEnv('COMSPEC'),ProgramName2);
        SwapVectors;
      end;
    { end if }
    ProgramName2:='copy miercole.dat a: >null';
    if ProgramName2 <> '' then
      begin
        ProgramName2 := '/C' + ProgramName2;
        SwapVectors;
        Exec(GetEnv('COMSPEC'),ProgramName2);
        SwapVectors;
      end;
    { end if }
    ProgramName2:='copy jueves.dat a: >null';
    if ProgramName2 <> '' then
      begin
        ProgramName2 := '/C' + ProgramName2;
        SwapVectors;
        Exec(GetEnv('COMSPEC'),ProgramName2);
        SwapVectors;
      end;
    { end if }
```

```
ProgramName2:='copy viernes.dat a: >null';
if ProgramName2 <> '' then
  begin
    ProgramName2 := '/C' + ProgramName2;
    SwapVectors;
    Exec(GetEnv('COMSPEC'),ProgramName2);
    SwapVectors;
  end;
{ end if }
ProgramName2:='copy sabado.dat a: >null';
if ProgramName2 <> '' then
  begin
    ProgramName2 := '/C' + ProgramName2;
    SwapVectors;
    Exec(GetEnv('COMSPEC'),ProgramName2);
    SwapVectors;
  end;
{ end if }
AbreArchivos;
{ ***** rutina borra informaci¢n de aparta ***** }
OperacionBtrieve(Primer_Registro,bp_apa_f,bp_apa,reg_apa,long_reg_apa,
          LlaveApa0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_apa_f,bp_apa,reg_apa,long_reg_apa,
                         LlaveApa0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_apa_f,bp_apa,reg_apa,
                 long_reg_apa,LlaveApa0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
  end;
{end if}
{ ***** rutina borra informacion de ocupa ***** }
OperacionBtrieve(Primer_Registro,bp_ocu_f,bp_ocu,reg_ocu,long_reg_ocu,
          LlaveOcu0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_ocu_f,bp_ocu,reg_ocu,long_reg_ocu,
                         LlaveOcu0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_ocu_f,bp_ocu,reg_ocu,
                     long_reg_ocu,LlaveOcu0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
```

```
    end;
  {end if}
  { ***** rutina borra informacion de cancela ***** }
  OperacionBtrieve(Primer_Registro,bp_can_f,bp_can,reg_can,long_reg_can,LlaveCan0,0);
  if estado_bt = 0 then
    begin
      repeat
        begin
          FDA:=False;
          OperacionBtrieve(BORRAR_REGISTRO,bp_can_f,bp_can,reg_can,long_reg_can,
                           LlaveCan0,0);
          if estado_bt=0 then
            begin
              OperacionBtrieve(Registro_Siguiente,bp_can_f,bp_can,reg_can,
                               long_reg_can,LlaveCan0,0);
              if estado_bt = 9 then
                begin
                  FDA:=True;
                end;
              {end if}
            end;
          { end if }
        end;
      until FDA;
      FDA:=False;
    end;
  {end if}
  { ***** rutina borra informacion de lunes ***** }
  OperacionBtrieve(Primer_Registro,bp_lun_f,bp_lun,reg_lun,long_reg_lun,
                   LlaveLun0,0);
  if estado_bt = 0 then
    begin
      repeat
        begin
          FDA:=False;
          OperacionBtrieve(BORRAR_REGISTRO,bp_lun_f,bp_lun,reg_lun,long_reg_lun,
                           LlaveLun0,0);
          if estado_bt=0 then
            begin
              OperacionBtrieve(Registro_Siguiente,bp_lun_f,bp_lun,reg_lun,
                               long_reg_lun,LlaveLun0,0);
              if estado_bt = 9 then
                begin
                  FDA:=True;
                end;
              {end if}
            end;
          { end if }
        end;
      until FDA;
      FDA:=False;
    end;
  {end if}
  { ***** rutina borra informacion de martes ***** }
  OperacionBtrieve(Primer_Registro,bp_mar_f,bp_mar,reg_mar,long_reg_mar,
                   LlaveMar0,0);
  if estado_bt = 0 then
    begin
      repeat
        begin
          FDA:=False;
          OperacionBtrieve(BORRAR_REGISTRO,bp_mar_f,bp_mar,reg_mar,long_reg_mar,
                           LlaveMar0,0);
          if estado_bt=0 then
            begin
              OperacionBtrieve(Registro_Siguiente,bp_mar_f,bp_mar,reg_mar,
                               long_reg_mar,LlaveMar0,0);
              if estado_bt = 9 then
                begin
```

```
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
  end;
{end if}
{ ***** rutina borra informacion de miercoles ***** }
OperacionBtrieve(Primer_Registro,bp_mie_f,bp_mie,reg_mie,long_reg_mie,
          LlaveMie0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_mie_f,bp_mie,reg_mie,long_reg_mie,
                LlaveMie0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_mie_f,bp_mie,reg_mie,
                    long_reg_mie,LlaveMie0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
  end;
{end if}
{ ***** rutina borra informacion de jueves ***** }
OperacionBtrieve(Primer_Registro,bp_jue_f,bp_jue,reg_jue,long_reg_jue,
          LlaveJue0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_jue_f,bp_jue,reg_jue,long_reg_jue,
                LlaveJue0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_jue_f,bp_jue,reg_jue,
                    long_reg_jue,LlaveJue0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
  end;
{end if}
{ ***** rutina borra informacion de viernes ***** }
OperacionBtrieve(Primer_Registro,bp_vie_f,bp_vie,reg_vie,long_reg_vie,
          LlaveVie0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
```

```
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_vie_f,bp_vie,reg_vie,long_reg_vie,
                LlaveVie0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_vie_f,bp_vie,reg_vie,
                    long_reg_vie,LlaveVie0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
  end;
{end if}
{ ***** rutina borra informacion de sabado ***** }
OperacionBtrieve(Primer_Registro,bp_sab_f,bp_sab,reg_sab,long_reg_sab,
        LlaveSab0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_sab_f,bp_sab,reg_sab,long_reg_sab,
                LlaveSab0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_sab_f,bp_sab,reg_sab,
                    long_reg_sab,LlaveSab0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
  end;
{end if}
{ ***** rutina borra informacion de EstLun ***** }
OperacionBtrieve(Primer_Registro,bp_est_lun_f,bp_est_lun,reg_est_lun,long_reg_est_lun,
        LlaveEstLun0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_est_lun_f,bp_est_lun,reg_est_lun,
                long_reg_est_lun,LlaveEstLun0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_est_lun_f,bp_est_lun,reg_est_lun,
                    long_reg_est_lun,LlaveEstLun0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
```

```
  end;
{end if}
{ ***** rutina borra informacion de EstMar ***** }
OperacionBtrieve(Primer_Registro,bp_est_mar_f,bp_est_mar,reg_est_mar,
          long_reg_est_mar,LlaveEstMar0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_est_mar_f,bp_est_mar,
                  reg_est_mar,long_reg_est_mar,LlaveEstMar0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_est_mar_f,bp_est_mar,reg_est_mar,
                      long_reg_est_mar,LlaveEstMar0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
  end;
{end if}
{ ***** rutina borra informacion de EstMie ***** }
OperacionBtrieve(Primer_Registro,bp_est_mie_f,bp_est_mie,reg_est_mie,
          long_reg_est_mie,LlaveEstMie0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_est_mie_f,bp_est_mie,
                  reg_est_mie,long_reg_est_mie,LlaveEstMie0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_est_mie_f,bp_est_mie,
                      reg_est_mie,long_reg_est_mie,LlaveEstMie0,0);
            if estado_bt = 9 then
              begin
                FDA:=True;
              end;
            {end if}
          end;
        { end if }
      end;
    until FDA;
    FDA:=False;
  end;
{end if}
{ ***** rutina borra informacion de EstJue ***** }
OperacionBtrieve(Primer_Registro,bp_est_jue_f,bp_est_jue,reg_est_jue,
          long_reg_est_jue,LlaveEstJue0,0);
if estado_bt = 0 then
  begin
    repeat
      begin
        FDA:=False;
        OperacionBtrieve(BORRAR_REGISTRO,bp_est_jue_f,bp_est_jue,
                  reg_est_jue,long_reg_est_jue,LlaveEstJue0,0);
        if estado_bt=0 then
          begin
            OperacionBtrieve(Registro_Siguiente,bp_est_jue_f,bp_est_jue,
                      reg_est_jue,long_reg_est_jue,LlaveEstJue0,0);
            if estado_bt = 9 then
```

```
                    begin
                      FDA:=True;
                    end;
                  {end if}
                end;
              { end if }
            end;
          until FDA;
          FDA:=False;
        end;
      {end if}
      { ***** rutina borra informacion de EstVie ***** }
      OperacionBtrieve(Primer_Registro,bp_est_vie_f,bp_est_vie,reg_est_vie,
                long_reg_est_vie,LlaveEstVie0,0);
      if estado_bt = 0 then
        begin
          repeat
            begin
              FDA:=False;
              OperacionBtrieve(BORRAR_REGISTRO,bp_est_vie_f,bp_est_vie,
                        reg_est_vie,long_reg_est_vie,LlaveEstVie0,0);
              if estado_bt=0 then
                begin
                  OperacionBtrieve(Registro_Siguiente,bp_est_vie_f,bp_est_vie,
                            reg_est_vie,long_reg_est_vie,LlaveEstVie0,0);
                  if estado_bt = 9 then
                    begin
                      FDA:=True;
                    end;
                  {end if}
                end;
              { end if }
            end;
          until FDA;
          FDA:=False;
        end;
      {end if}
      { ***** rutina borra informacion de EstSab ***** }
      OperacionBtrieve(Primer_Registro,bp_est_sab_f,bp_est_sab,reg_est_sab,
                long_reg_est_sab,LlaveEstSab0,0);
      if estado_bt = 0 then
        begin
          repeat
            begin
              FDA:=False;
              OperacionBtrieve(BORRAR_REGISTRO,bp_est_sab_f,bp_est_sab,
                        reg_est_sab,long_reg_est_sab,LlaveEstSab0,0);
              if estado_bt=0 then
                begin
                  OperacionBtrieve(Registro_Siguiente,bp_est_sab_f,bp_est_sab,
                            reg_est_sab,long_reg_est_sab,LlaveEstSab0,0);
                  if estado_bt = 9 then
                    begin
                      FDA:=True;
                    end;
                  {end if}
                end;
              { end if }
            end;
          until FDA;
          FDA:=False;
        end;
      {end if}
      MensajeError(' Fin del Respaldo ');
    end
  else
    MensajeError(' Respaldo Abortado ');
  { end if }
end;
```

```
{*****************************************************}

Program SALID(input,output,SALIDA);
uses
  btr;
Type
  arr2   = array[1..2] of char;
  arr3   = array[1..3] of char;
  arr6   = array[1..6] of char;
  arr29  = array[1..29] of char;
  arr128 = array[1..128] of char;
  RegistroApa = Record
                  matr_ap : arr6;
                  cve_hora_ap : arr2;
                  cve_maq_ap : arr3;
                  cve_tipo_ap : char;
                end;
  RegistroOcu = Record
                  matricula : arr6;
                  hora : arr2;
                  maquina : arr3;
                  Tipo_maq : char;
                end;

var
  estado_bt,
  estado_bt1: Integer;
  Salida    : Text;
  matricula : arr6;
  reg_apa   : RegistroApa;
  reg_ocu   : RegistroOcu;
  long_reg_apa,
  long_reg_ocu : integer;
  bp_apa,
  bp_ocu    : arr128;
  arch_apa,
  arch_ocu  : arr29;
  llaveApa0,
  llaveOcu0 : arr6;
  llaveOcu1 : arr3;
  llaveOcu2 : arr2;
  llaveOcu3 : char;


{*****************************************************}
Procedure AbreArchivos;
  begin
    long_reg_apa := sizeof(reg_apa);
    arch_apa := 'f:\user\aparta\aparta.dat    ';
    estado_bt:=btrv(ABRIR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
    long_reg_ocu := sizeof(reg_ocu);
    arch_ocu := 'f:\user\aparta\ocupa.dat    ';
    estado_bt:=btrv(ABRIR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
  end;
{ end procedure AbreArchivos}
{*****************************************************}

Procedure CierraArchivos;
  begin
    estado_bt:=btrv(CERRAR_ARCHIVO,bp_apa,reg_apa,long_reg_apa,arch_apa,0);
    estado_bt:=btrv(CERRAR_ARCHIVO,bp_ocu,reg_ocu,long_reg_ocu,arch_ocu,0);
  end;
{ end procedure AbreArchivos}

{*****************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida: { Arreglo de salida }
```

```
                    maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
  MAX = 74;
var
  long : byte absolute cadena_entrada;
  arreglo : array[1..MAX] of char absolute arreglo_salida;
  i : byte;
begin
  if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
  for i := 1 to long do arreglo[i] := cadena_entrada[i];
  for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*********************************************}

function arreglo_a_cadena( var arreglo_entrada; { Arreglo de entrada }
                    maxcar : byte) : string;
                    { Numero maximo de caracteres del arreglo fuente}
const
  MAX = 74;
var
  cadena_salida : string;
  long : byte absolute cadena_salida;
  arreglo : array[1..MAX] of char absolute arreglo_entrada;
  i : byte;
begin
  if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
  for i := 1 to maxcar do cadena_salida[i] := arreglo[i];
  long := maxcar;

  { Se eliminan los blancos al final de la cadena }

  while (cadena_salida[long] = #32 {ESPACIO}) and (long > 0) do dec(long);
  arreglo_a_cadena := cadena_salida;
end; { arreglo_a_cadena }


{*********************************************}

BEGIN
  {Código que se emplea para que se puedan hacer los accesos}
  long_reg_apa:=sizeof(Reg_Apa);
  long_reg_ocu:=sizeof(Reg_Ocu);
  assign(Salida,'f:\user\aparta\salida.txt');
  rewrite(Salida);
  AbreArchivos;

  {Código que se emplea para el acceso propiamente dicho}

  cadena_a_arreglo(ParamStr(1),matricula,6);
  llaveApa0:=matricula;
  llaveOcu0:=matricula;

  estado_bt:=btrv(REGISTRO_IGUAL,bp_ocu,reg_ocu,
            long_reg_ocu,LlaveOcu0,0);
  if estado_bt=0 then
    begin
      estado_bt:=btrv(BORRAR_REGISTRO,bp_ocu,reg_ocu,
                long_reg_ocu,LlaveOcu0,0);
      if estado_bt<>0 then
        writeln(Salida,'1,ERROR BORRANDO OCUPA ')
      else
        begin
          estado_bt1:=btrv(REGISTRO_IGUAL,bp_apa,reg_apa,long_reg_apa,
                    LlaveApa0,0);
          if estado_bt1=0 then
            begin
              estado_bt1:=btrv(BORRAR_REGISTRO,bp_apa,reg_apa,long_reg_apa,
```

```
                    LlaveApa0,0);
          if estado_bt1 <>0 then
            writeln(Salida,'2,ERROR BORRANDO APARTA ')
          else
            writeln(Salida,'0,BORRADO EXITOSO');
          { end if }
        end;
      { end if }
    end;
  { end if }
 end
else
  writeln(Salida,'3,EL USUARIO NO ESTA EN EL C.E.C.');
{ end if }
Close(Salida);
ClierraArchivos;
end.
end.
{ end program Salida }

Program Servidor(Input,Output,Serv);
uses
 btr;

{*****************************************************}
Type
 arr_ser = array[1..29] of char;
 arr2 = array[1..2] of char;
 arr3 = array[1..3] of char;
 arr15 = array[1..15] of char;
 arr20 = array[1..20] of char;
 arr29 = array[1..29] of char;
 arr128 = array[1..128] of char;

 RegistroSer = Record
          clave_ser : arr2;
          nombre_Ser : arr20;
          dirIP1 : arr3;
          dirIP2 : arr3;
          dirIP3 : arr3;
          dirIP4 : arr3;
          dirEth : arr15;
          end;

var
 estado_bt,
 long_reg_ser,
 codigo      : integer;
 aborto      : boolean;
 bp_ser      : arr128;
 reg_ser     : RegistroSer;
 llaveSer0   : arr2;
 arch_ser    : arr29;
 Serv        : Text;
 Operacion   : byte;
{*****************************************************}
Procedure AbreArchivos;
 begin
  long_reg_ser := sizeof(reg_ser);
  arch_ser := 'F:\USER\APARTA\SERVIDOR.DAT ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_ser,reg_ser,long_reg_ser,arch_ser,0);
 end;
{ end procedure AbreArchivos}
{*****************************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_ser,reg_ser,long_reg_ser,arch_ser,0);
 end;
```

```
{ end procedure AbreArchivos}

{*************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
  MAX = 74;
var
  long : byte absolute cadena_entrada;
  arreglo : array[1..MAX] of char absolute arreglo_salida;
  i : byte;
begin
  if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
  for i := 1 to long do arreglo[i] := cadena_entrada[i];
  for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*************************************************}

Begin
  {Código que se emplea para que se puedan hacer los accesos}
  long_reg_ser:=sizeof(Reg_Ser);
  assign(Serv,'F:\USER\APARTA\serv.bt');
  rewrite(Serv);
  AbreArchivos;


  {Código que se emplea para el acceso propiamente dicho}
  Val(ParamStr(1),Operacion,codigo);
  Case Operacion of
    1: Begin
       cadena_a_arreglo(ParamStr(2),reg_ser.clave_ser,2);
       cadena_a_arreglo(ParamStr(3),reg_ser.nombre_ser,20);
       cadena_a_arreglo(ParamStr(4),reg_ser.dirIP1,3);
       cadena_a_arreglo(ParamStr(5),reg_ser.dirIP2,3);
       cadena_a_arreglo(ParamStr(6),reg_ser.dirIP3,3);
       cadena_a_arreglo(ParamStr(7),reg_ser.dirIP4,3);
       cadena_a_arreglo(ParamStr(8),reg_ser.dirEth,15);
       Estado_bt:= btrv(INSERTAR_REGISTRO,bp_ser,reg_ser,long_reg_ser,
                        LlaveSer0,0);
       case estado_bt of
         0: writeln(Serv,'0,ALTA EXITOSA');
         5: writeln(Serv,'5,EL SERVIDOR YA ESTA DADO DE ALTA');
       else
         writeln(Serv,'1,IMPOSIBLE HACER ALTA');
       end; { Case }
     End;

    2: Begin
       cadena_a_arreglo(ParamStr(2),LlaveSer0,2);
       Estado_bt:= btrv(REGISTRO_IGUAL,bp_ser,reg_ser,long_reg_ser,
                        LlaveSer0,0);
       If estado_bt=0 then
         begin
           Estado_bt:= btrv(BORRAR_REGISTRO,bp_ser,reg_ser,long_reg_ser,
                            LlaveSer0,0);
           If estado_bt=0 then
             Writeln(Serv,'0,SERVIDOR BORRADO')
           else
             Writeln(Serv,'1,IMPOSIBLE HACER BAJA');
           { End If }
         End
       Else
         Writeln(Serv,'4,NO EXISTE ESTE SERVIDOR');
       { End If }
     End;
```

```
  3: Begin
      cadena_a_arreglo(ParamStr(2),LlaveSer0,2);
      Estado_bt:= btrv(REGISTRO_IGUAL,bp_ser,reg_ser,long_reg_ser,
                  LlaveSer0,0);
      If estado_bt=0 then
       begin
        cadena_a_arreglo(ParamStr(2),reg_ser.clave_ser,2);
        cadena_a_arreglo(ParamStr(3),reg_ser.nombre_ser,20);
        cadena_a_arreglo(ParamStr(4),reg_ser.dirIP1,3);
        cadena_a_arreglo(ParamStr(5),reg_ser.dirIP2,3);
        cadena_a_arreglo(ParamStr(6),reg_ser.dirIP3,3);
        cadena_a_arreglo(ParamStr(7),reg_ser.dirIP4,3);
        cadena_a_arreglo(ParamStr(8),reg_ser.dirEth,15);
        Estado_bt:= btrv(ACTUALIZAR_REGISTRO,bp_ser,reg_ser,long_reg_ser,
                    LlaveSer0,0);
        if estado_bt=5 then
         Writeln(Serv,'5,YA ESTA DADO DE ALTA EL SERVIDOR')
        else
         if estado_bt=0 then
          Writeln(Serv,'0,ACTUALIZACION EXITOSA')
         else
          Writeln(Serv,'1,IMPOSIBLE HACER ACTUALIZACION');
         { end if }
        { end if }
       end
      Else
       Writeln(Serv,'4,NO EXISTE ESTE SERVIDOR');
      { End if }
     End;

  4: Begin
      cadena_a_arreglo(ParamStr(2),LlaveSer0,2);
      Estado_bt:= btrv(REGISTRO_IGUAL,bp_ser,reg_ser,long_reg_ser,
                  LlaveSer0,0);
      If estado_bt=0 then
       Writeln(Serv,'0,'+reg_ser.clave_ser+','+
            reg_ser.nombre_ser+','+
            reg_ser.dirIP1+','+
            reg_ser.dirIP2+','+
            reg_ser.dirIP3+','+
            reg_ser.dirIP4+','+
            reg_ser.dirEth)
      Else
       Writeln(Serv,'4,NO EXISTE ESTE SERVIDOR'+',0,0,0,0,0,0');
      { End if }
     End;
  End; { Case }
  CierraArchivos;
  close(Serv);
End.

Program Tipo_Maquina(Input,Output,TMa);
uses
  btr;

{*****************************************************}
Type
  arr29  = array [1..29] of char;   {para el nombre del archivo btrieve}
  arr128 = array [1..128] of char;  {para el databuffer}
  arr20  = array [1..20] of char;   {para desc. de m quina}
  RegistroTma = Record              {registro del archivo de tipoMaq}
             ClaveTipoMaq  : char;
             DescripTipoMaq : arr20;
             end;
var
  estado_bt,
  long_reg_TMa,
  codigo      : integer;
```

```
  aborto      : boolean;
  bp_TMa      : arr128;
  reg_Tma     : RegistroTma;
  llaveTma0   : char;
  arch_TMa    : arr29;
  TMa         : Text;
  Operacion   : byte;
{*****************************************************}
Procedure AbreArchivos;
 begin
  long_reg_Tma := sizeof(reg_Tma);
  arch_TMa := 'F:\USER\APARTA\TIPOMAQ.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_TMa,reg_Tma,long_reg_Tma,arch_TMa,0);
 end;
{ end procedure AbreArchivos}
{*****************************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_TMa,reg_Tma,long_reg_Tma,arch_TMa,0);
 end;
{ end procedure AbreArchivos}

{*****************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
 MAX = 74;
var
 long : byte absolute cadena_entrada;
 arreglo : array[1..MAX] of char absolute arreglo_salida;
 i : byte;
begin
 if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
 for i := 1 to long do arreglo[i] := cadena_entrada[i];
 for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*****************************************************}

Begin
 {Código que se emplea para que se puedan hacer los accesos}
 long_reg_Tma:=sizeof(reg_Tma);
 assign(TMa,'F:\USER\APARTA\TMa.txt');
 rewrite(TMa);
 AbreArchivos;


 {Código que se emplea para el acceso propiamente dicho}
 Val(ParamStr(1),Operacion,codigo);
 Case Operacion of
  1: Begin
      cadena_a_arreglo(ParamStr(2),reg_Tma.ClaveTipoMaq,1);
      cadena_a_arreglo(ParamStr(3),reg_Tma.DescripTipoMaq,20);
      Estado_bt:= btrv(INSERTAR_REGISTRO,bp_TMa,reg_Tma,long_reg_Tma,
                  llaveTMa0,0);
      case estado_bt of
       0: writeln(TMa,'0,ALTA EXITOSA');
       5: writeln(TMa,'5,EL TIPO DE MAQUINA YA ESTA DADO DE ALTA');
      else
       writeln(TMa,'1,IMPOSIBLE HACER ALTA');
      end; { Case }
     End;

  2: Begin
      cadena_a_arreglo(ParamStr(2),llaveTMa0,1);
```

```
    Estado_bt:= btrv(REGISTRO_IGUAL,bp_TMa,reg_Tma,long_reg_Tma,
                 LlaveTMa0,0);
    If estado_bt=0 then
     begin
      Estado_bt:= btrv(BORRAR_REGISTRO,bp_TMa,reg_Tma,long_reg_Tma,
                   LlaveTMa0,0);
      If estado_bt=0 then
       Writeln(TMa,'0,TIPO DE MAQUINA BORRADO')
      else
       Writeln(TMa,'1,IMPOSIBLE HACER BAJA');
      { End If }
     End
    Else
     Writeln(TMa,'4,NO EXISTE ESTE TIPO DE MAQUINA');
    { End If }
   End;

  3: Begin
    cadena_a_arreglo(ParamStr(2),LlaveTMa0,1);
    Estado_bt:= btrv(REGISTRO_IGUAL,bp_TMa,reg_Tma,long_reg_Tma,
                 LlaveTMa0,0);
    If estado_bt=0 then
     begin
    cadena_a_arreglo(ParamStr(2),reg_Tma.ClaveTipoMaq,1);
    cadena_a_arreglo(ParamStr(3),reg_Tma.DescripTipoMaq,20);
      Estado_bt:= btrv(ACTUALIZAR_REGISTRO,bp_TMa,reg_Tma,long_reg_Tma,
                   LlaveTMa0,0);
      If estado_bt=5 then
       Writeln(TMa,'5,YA ESTA DADO DE ALTA EL TIPO DE MAQUINA')
      else
       If estado_bt=0 then
        Writeln(TMa,'0,ACTUALIZACION EXITOSA')
       else
        Writeln(TMa,'1,IMPOSIBLE HACER ACTUALIZACION');
       { end If }
      { end If }
     end
    Else
     Writeln(TMa,'4,NO EXISTE ESTE TIPO DE MAQUINA');
    { End If }
   End;

  4: Begin
     cadena_a_arreglo(ParamStr(2),LlaveTMa0,1);
     Estado_bt:= btrv(REGISTRO_IGUAL,bp_TMa,reg_Tma,long_reg_Tma,
                 LlaveTMa0,0);
     If estado_bt=0 then
      Writeln(TMa,'0,'+reg_Tma.ClaveTipoMaq+','+
           reg_Tma.DescripTipoMaq)
     Else
      Writeln(TMa,'4,NO EXISTE ESTE TIPO DE MAQUINA'+',0');
     { End If }
    End;
 End; { Case }
 CierraArchivos;
 close(TMa);
End.

Program Tipo_Usuarios(Input,Output,TUs);
uses
 btr;

{*****************************************************}
Type
 arr29  = array [1..29] of char;   {para el nombre del archivo btrieve}
 arr128 = array [1..128] of char;  {para el databuffer}
 arr20  = array [1..20] of char;   {para desc. del usuario}
 RegistroTUs = Record              {registro del archivo de tipoUsu}
         ClaveTipoUsu   : char;
```

```
            DescripTipoUsu : arr20;
            end;
var
  estado_bt,
  long_reg_TUs,
  codigo      : integer;
  aborto      : boolean;
  bp_TUs      : arr128;
  reg_TUs     : RegistroTUs;
  llaveTUs0   : char;
  arch_TUs    : arr29;
  TUs         : Text;
  Operacion   : byte;
{*****************************************************}
Procedure AbreArchivos;
  begin
    long_reg_TUs := sizeof(reg_TUs);
    arch_TUs := 'F:\USER\APARTA\TIPOUSU.DAT  ';
    estado_bt:=btrv(ABRIR_ARCHIVO,bp_TUs,reg_TUs,long_reg_TUs,arch_TUs,0);
  end;
{ end procedure AbreArchivos}
{*****************************************************}

Procedure CierraArchivos;
  begin
    estado_bt:=btrv(CERRAR_ARCHIVO,bp_TUs,reg_TUs,long_reg_TUs,arch_TUs,0);
  end;
{ end procedure AbreArchivos}

{*****************************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
                            var arreglo_salida; { Arreglo de salida }
                            maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
  MAX = 74;
var
  long : byte absolute cadena_entrada;
  arreglo : array[1..MAX] of char absolute arreglo_salida;
  i : byte;
begin
  if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
  for i := 1 to long do arreglo[i] := cadena_entrada[i];
  for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*****************************************************}

Begin
  {Código que se emplea para que se puedan hacer los accesos}
  long_reg_TUs:=sizeof(reg_TUs);
  assign(TUs,'F:\USER\APARTA\TUs.txt');
  rewrite(TUs);
  AbreArchivos;


  {Código que se emplea para el acceso propiamente dicho}
  Val(ParamStr(1),Operacion,codigo);
  Case Operacion of
    1: Begin
         cadena_a_arreglo(ParamStr(2),reg_TUs.ClaveTipoUsu,1);
         cadena_a_arreglo(ParamStr(3),reg_TUs.DescripTipoUsu,20);
         Estado_bt:= btrv(INSERTAR_REGISTRO,bp_TUs,reg_TUs,long_reg_TUs,
                          llaveTUs0,0);
         case estado_bt of
           0: writeln(TUs,'0,ALTA EXITOSA',';0');
           5: writeln(TUs,'5,EL TIPO DE USUARIO YA ESTA DADO DE ALTA');
         else
```

```
      writeln(TUs,'1,IMPOSIBLE HACER ALTA');
    end; { Case }
  End;

  2: Begin
    cadena_a_arreglo(ParamStr(2),LlaveTUs0,1);
    Estado_bt:= btrv(REGISTRO_IGUAL,bp_TUs,reg_TUs,long_reg_TUs,
                LlaveTUs0,0);
    If estado_bt=0 then
     begin
      Estado_bt:= btrv(BORRAR_REGISTRO,bp_TUs,reg_TUs,long_reg_TUs,
                  LlaveTUs0,0);
      If estado_bt=0 then
       Writeln(TUs,'0,TIPO DE USUARIO BORRADO')
      else
       Writeln(TUs,'1,IMPOSIBLE HACER BAJA');
      { End If }
     End
    Else
     Writeln(TUs,'4,NO EXISTE ESTE TIPO DE USUARIO');
    { End If }
  End;

  3: Begin
    cadena_a_arreglo(ParamStr(2),LlaveTUs0,1);
    Estado_bt:= btrv(REGISTRO_IGUAL,bp_TUs,reg_TUs,long_reg_TUs,
                LlaveTUs0,0);
    If estado_bt=0 then
     begin
    cadena_a_arreglo(ParamStr(2),reg_TUs.ClaveTipoUsu,1);
    cadena_a_arreglo(ParamStr(3),reg_TUs.DescripTipoUsu,20);
     Estado_bt:= btrv(ACTUALIZAR_REGISTRO,bp_TUs,reg_TUs,long_reg_TUs,
                LlaveTUs0,0);
     If estado_bt=5 then
      Writeln(TUs,'5,YA ESTA DADO DE ALTA EL TIPO DE USUARIO')
     else
      if estado_bt=0 then
       Writeln(TUs,'0,ACTUALIZACION EXITOSA')
      else
       Writeln(TUs,'1,IMPOSIBLE HACER ACTUALIZACION');
       { end if }
     { end if }
    end
    Else
     Writeln(TUs,'4,NO EXISTE ESTE TIPO DE USUARIO');
    { End If }
  End;

  4: Begin
    cadena_a_arreglo(ParamStr(2),LlaveTUs0,1);
    Estado_bt:= btrv(REGISTRO_IGUAL,bp_TUs,reg_TUs,long_reg_TUs,
                LlaveTUs0,0);
    If estado_bt=0 then
     Writeln(TUs,'0,'+reg_TUs.ClaveTipoUsu+','+
          reg_TUs.DescripTipoUsu)
    Else
     Writeln(TUs,'4,NO EXISTE ESTE TIPO DE USUARIO'+',0');
    { End If }
  End;
 End; { Case }
 CierraArchivos;
 close(TUs);
End.

Program Usuario(input,Output,Usu);
uses
 btr;

{*************************************************}
```

```
Type
 arr29  = array [1..29] of char;   {para el nombre del archivo btrieve}
 arr128 = array [1..128] of char;  {para el databuffer}
 arr20  = array [1..20] of char;   {para nombres}
 arr6   = array [1..6] of char;    {para la matrícula y carrera}
 arr2   = array [1..2] of char;    {para acumulador de sanciones}
 arr40  = array [1..40] of char;   {para motivo}
 RegistroUsu = Record              {registro del archivo de usuarios}
          matricula : arr6;
          paterno   : arr20;
          materno   : arr20;
          nombre    : arr20;
          tipo      : char;
          habil     : char;
          status    : char;
          acumula_sanciones : arr2;
          carrera   : arr6;
          end;
 RegistroUsuDes = Record
                  matricula_des : arr6;
                  motivo : arr40;
                  fecha : arr6;
                  end;
 RegLlaveUsu1= Record
          paterno : arr20;
          materno : arr20;
          nombre : arr20;
          end;
 RegLlaveUsu2= Record
          paterno   : arr20;
          materno   : arr20;
          nombre    : arr20;
          matricula : arr6;
          end;
 RegLlaveUsu3= Record
          matricula : arr6;
          carrera : arr6;
          end;
var
 estado_bt,
 long_reg_usu,
 long_reg_usu_des,
 codigo      : integer;
 aborto      : boolean;
 bp_usu      : arr128;
 reg_usu     : RegistroUsu;
 llaveUsu0   : arr2;
 arch_usu    : arr29;
 Usu         : Text;
 Operacion   : byte;
{*********************************************}
Procedure AbreArchivos;
 begin
  long_reg_usu := sizeof(reg_usu);
  arch_usu := 'F:\USER\APARTAU\USUARIO.DAT  ';
  estado_bt:=btrv(ABRIR_ARCHIVO,bp_usu,reg_usu,long_reg_usu,arch_usu,0);
 end;
{ end procedure AbreArchivos}
{*********************************************}

Procedure CierraArchivos;
 begin
  estado_bt:=btrv(CERRAR_ARCHIVO,bp_usu,reg_usu,long_reg_usu,arch_usu,0);
 end;
{ end procedure AbreArchivos}

{*********************************************}

procedure cadena_a_arreglo( cadena_entrada : string; { Cadena a transformar a arreglo }
```

```
                var arreglo_salida; { Arreglo de salida }
                maxcar : byte); { Numero maximo de caracteres del arreglo objeto }
const
  MAX = 74;
var
  long : byte absolute cadena_entrada;
  arreglo : array[1..MAX] of char absolute arreglo_salida;
  i : byte;
begin
  if (maxcar < 0) or (maxcar > MAX) then maxcar := 2;
  for i := 1 to long do arreglo[i] := cadena_entrada[i];
  for i := succ(long) to maxcar do arreglo[i] := #32; {ESPACIO}
end;
{ cadena_a_arreglo }

{*****************************************************}

Begin
  {Código que se emplea para que se puedan hacer los accesos}
  long_reg_usu:=sizeof(Reg_Usu);
  assign(Usu,'F:\USER\APARTA\usu.txt');
  rewrite(Usu);
  AbreArchivos;

  {Código que se emplea para el acceso propiamente dicho}
  Val(ParamStr(1),Operacion,codigo);
  Case Operacion of
   1: Begin
      cadena_a_arreglo(ParamStr(2),reg_usu.matricula,6);
      cadena_a_arreglo(ParamStr(3),reg_usu.paterno,20);
      cadena_a_arreglo(ParamStr(4),reg_usu.materno,20);
      cadena_a_arreglo(ParamStr(5),reg_usu.nombre,20);
      cadena_a_arreglo(ParamStr(6),reg_usu.tipo,1);
      cadena_a_arreglo(ParamStr(7),reg_usu.habil,1);
      cadena_a_arreglo(ParamStr(8),reg_usu.status,1);
      cadena_a_arreglo(ParamStr(9),reg_usu.acumula_sanciones,2);
      cadena_a_arreglo(ParamStr(10),reg_usu.carrera,6);
      Estado_bt:= btrv(INSERTAR_REGISTRO,bp_usu,reg_usu,long_reg_usu,
                       LlaveUsu0,0);
      case estado_bt of
        0: writeln(Usu,'0,ALTA EXITOSA');
        5: writeln(Usu,'5,EL USUARIO YA ESTA DADO DE ALTA');
      else
        writeln(Usu,'1,IMPOSIBLE HACER ALTA');
      end; { Case }
     End;

   2: Begin
      cadena_a_arreglo(ParamStr(2),LlaveUsu0,6);
      Estado_bt:= btrv(REGISTRO_IGUAL,bp_usu,reg_usu,long_reg_usu,
                       LlaveUsu0,0);
      If estado_bt=0 then
        begin
          Estado_bt:= btrv(BORRAR_REGISTRO,bp_usu,reg_usu,long_reg_usu,
                           LlaveUsu0,0);
          If estado_bt=0 then
            Writeln(Usu,'0,USUARIO BORRADO')
          else
            Writeln(Usu,'1,IMPOSIBLE HACER BAJA');
          { End If }
        End
      Else
        Writeln(Usu,'4,NO EXISTE ESTE USUARIO');
      { End If }
     End;

   3: Begin
      cadena_a_arreglo(ParamStr(2),LlaveUsu0,6);
      Estado_bt:= btrv(REGISTRO_IGUAL,bp_usu,reg_usu,long_reg_usu,
```

```
                LlaveUsu0,0);
      If estado_bt=0 then
        begin
          cadena_a_arreglo(ParamStr(2),reg_usu.matricula,6);
          cadena_a_arreglo(ParamStr(3),reg_usu.paterno,20);
          cadena_a_arreglo(ParamStr(4),reg_usu.materno,20);
          cadena_a_arreglo(ParamStr(5),reg_usu.nombre,20);
          cadena_a_arreglo(ParamStr(6),reg_usu.tipo,1);
          cadena_a_arreglo(ParamStr(7),reg_usu.habil,1);
          cadena_a_arreglo(ParamStr(8),reg_usu.status,1);
          cadena_a_arreglo(ParamStr(9),reg_usu.acumula_sanciones,2);
          cadena_a_arreglo(ParamStr(10),reg_usu.carrera,6);
          Estado_bt:= btrv(ACTUALIZAR_REGISTRO,bp_usu,reg_usu,long_reg_usu,
                LlaveUsu0,0);
          If estado_bt=5 then
            Writeln(Usu,'5,YA ESTA DADO DE ALTA EL USUARIO')
          else
            If estado_bt=0 then
              Writeln(Usu,'0,ACTUALIZACION EXITOSA')
            else
              Writeln(Usu,'1,IMPOSIBLE HACER ACTUALIZACION');
            { end if }
          { end if }
        end
      Else
        Writeln(Usu,'4,NO EXISTE ESTE USUARIO');
      { End if }
    End;

  4: Begin
      cadena_a_arreglo(ParamStr(2),LlaveUsu0,6);
      Estado_bt:= btrv(REGISTRO_IGUAL,bp_usu,reg_usu,long_reg_usu,
                LlaveUsu0,0);
      If estado_bt=0 then
        Writeln(Usu,'0,'+reg_usu.matricula+','+
                reg_usu.paterno+','+
                reg_usu.materno+','+
                reg_usu.nombre+','+
                reg_usu.tipo+','+
                reg_usu.habil+','+
                reg_usu.status+','+
                reg_usu.acumula_sanciones+','+
                reg_usu.carrera)
      Else
        Writeln(Usu,'4,NO EXISTE ESTE USUARIO'+',0,0,0,0,0,0,0,0');
      { End If }
    End;
 End; { Case }
 CierraArchivos;
 close(Usu);
End.

const
 ESPACIO=#32;
 CheckBreak : Boolean = False;

Type
 arr14  = array [1..14] of char;
 arr128 = array [1..128] of char;
 arr20  = array [1..20] of char;
 arr6   = array [1..6] of char;
 arr15  = array [1..15] of char;
 arr3   = array [1..3] of char;
 arr2   = array [1..2] of char;
 arr4   = array [1..4] of char;
 arr5   = array [1..5] of char;
 arr10  = array [1..10] of char;
 arr13  = array [1..13] of char;
 arr29  = array [1..29] of char;
```

```
arr40  = array [1..40] of char;

RegistroUsu = Record
        matricula : arr6;
        paterno : arr20;
        materno : arr20;
        nombre : arr20;
        tipo : char;
        habil : char;
        status : char;
        sanciones : arr2;
        carrera : arr6;
        end;
RegistroUsuDes = Record
                 matricula_des : arr6;
                 motivo : arr40;
                 fecha : arr6;
                 end;
RegistroApa = Record
        matr_ap : arr6;
        cve_hora_ap : arr2;
        cve_maq_ap : arr3;
        cve_tipo_ap : char;
        end;
RegistroMaq = Record
        claveMaq : arr3;
        claveTipo : char;
        serieCPU : arr15;
        serieTeclado :arr15;
        serieMonitor : arr15;
        serieMouse : arr15;
        modelo : arr15;
        RAM : arr3;
        HD : arr4;
        micro : arr6;
        video : arr5;
        FD : arr5;
        servidor : arr2;
        isla : arr2;
        enServicio : char;
        end;
RegistroSer = Record
        clave_ser : arr2;
        nombre_Ser : arr20;
        dirIP1 : arr3;
        dirIP2 : arr3;
        dirIP3 : arr3;
        dirIP4 : arr3;
        dirEth : arr15;
        end;
RegistroTUs = Record
        claveTipoUsu : char;
        descripTipoUsu : arr20;
        end;
RegistroTMa = Record
        claveTipoMaq : char;
        descripTipoMaq : arr20;
        end;
RegistroHor = Record
        ClaveHora : arr2;
        horas : arr13;
        end;
RegistroOcu = Record
        matricula : arr6;
        hora : arr2;
        maquina : arr3;
        Tipo_maq : char;
        end;
RegLlaveUsu1= Record
```

```
        paterno : arr20;
        materno : arr20;
        nombre : arr20;
        end;
RegLlaveUsu2= Record
        paterno   : arr20;
        materno   : arr20;
        nombre    : arr20;
        matricula : arr6;
        end;
RegLlaveUsu3= Record
        matricula : arr6;
        carrera : arr6;
        end;
RegLlaveMaq1= Record
        ClaveTipo  : char;
        EnServicio : char;
        end;
RegLlaveApa1= Record
        clave_Hora_ap : arr2;
        clave_Maq_ap : arr3;
        clave_Tipo_ap : char;
        end;
RegLlaveSer3= Record
        dirIP1 : arr3;
        dirIP2 : arr3;
        dirIP3 : arr3;
        dirIP4 : arr3;
        end;
RegllaveCan0= Record
        matricula_can : arr6;
        end;
RegistroHr=  record
        hr,
        min,
        sec,
        sec100 : word;
        end;
RegistroFec=  record
        year,
        month,
        day,
        dow : word;
        end;
RegistroLun= record
        cve_hora_lun,
        cont_hr_lun0,
        cont_maq_lun0,
        cont_hr_lun1,
        cont_maq_lun1,
        cont_hr_lun2,
        cont_maq_lun2 : integer;
        end;
RegistroMar= record
        cve_hora_mar,
        cont_hr_mar0,
        cont_maq_mar0,
        cont_hr_mar1,
        cont_maq_mar1,
        cont_hr_mar2,
        cont_maq_mar2 : integer;
        end;
RegistroMie= record
        cve_hora_mie,
        cont_hr_mie0,
        cont_maq_mie0,
        cont_hr_mie1,
        cont_maq_mie1,
        cont_hr_mie2,
```

```
        cont_maq_mie2 : integer;
        end;
RegistroJue= record
        cve_hora_jue,
        cont_hr_jue0,
        cont_maq_jue0,
        cont_hr_jue1,
        cont_maq_jue1,
        cont_hr_jue2,
        cont_maq_jue2 : integer;
        end;
RegistroVie= record
        cve_hora_vie,
        cont_hr_vie0,
        cont_maq_vie0,
        cont_hr_vie1,
        cont_maq_vie1,
        cont_hr_vie2,
        cont_maq_vie2 : integer;
        end;
RegistroSab= record
        cve_hora_sab,
        cont_hr_sab0,
        cont_maq_sab0,
        cont_hr_sab1,
        cont_maq_sab1,
        cont_hr_sab2,
        cont_maq_sab2 : integer;
        end;
RegistroEstLun= record
        cve_hora_est_lun,
        cont_tot_apa_lun0,
        cont_apa_sinent_lun0,
        cont_ent_sinapa_lun0,
        cont_tot_apa_lun1,
        cont_apa_sinent_lun1,
        cont_ent_sinapa_lun1,
        cont_tot_apa_lun2,
        cont_apa_sinent_lun2,
        cont_ent_sinapa_lun2 : integer;
        end;
RegistroEstMar= record
        cve_hora_est_mar,
        cont_tot_apa_mar0,
        cont_apa_sinent_mar0,
        cont_ent_sinapa_mar0,
        cont_tot_apa_mar1,
        cont_apa_sinent_mar1,
        cont_ent_sinapa_mar1,
        cont_tot_apa_mar2,
        cont_apa_sinent_mar2,
        cont_ent_sinapa_mar2 : integer;
        end;
RegistroEstMie= record
        cve_hora_est_mie,
        cont_tot_apa_mie0,
        cont_apa_sinent_mie0,
        cont_ent_sinapa_mie0,
        cont_tot_apa_mie1,
        cont_apa_sinent_mie1,
        cont_ent_sinapa_mie1,
        cont_tot_apa_mie2,
        cont_apa_sinent_mie2,
        cont_ent_sinapa_mie2 : integer;
        end;
RegistroEstJue= record
        cve_hora_est_jue,
        cont_tot_apa_jue0,
        cont_apa_sinent_jue0,
```

```
        cont_ent_sinapa_jue0,
        cont_tot_apa_jue1,
        cont_apa_sinent_jue1,
        cont_ent_sinapa_jue1,
        cont_tot_apa_jue2,
        cont_apa_sinent_jue2,
        cont_ent_sinapa_jue2 : integer;
        end;
RegistroEstVie= record
        cve_hora_est_vie,
        cont_tot_apa_vie0,
        cont_apa_sinent_vie0,
        cont_ent_sinapa_vie0,
        cont_tot_apa_vie1,
        cont_apa_sinent_vie1,
        cont_ent_sinapa_vie1,
        cont_tot_apa_vie2,
        cont_apa_sinent_vie2,
        cont_ent_sinapa_vie2 : integer;
        end;
RegistroEstSab= record
        cve_hora_est_sab,
        cont_tot_apa_sab0,
        cont_apa_sinent_sab0,
        cont_ent_sinapa_sab0,
        cont_tot_apa_sab1,
        cont_apa_sinent_sab1,
        cont_ent_sinapa_sab1,
        cont_tot_apa_sab2,
        cont_apa_sinent_sab2,
        cont_ent_sinapa_sab2 : integer;
        end;
RegistroRed = Record
        matr_red : arr6;
        cve_hora_red : arr2;
        cve_maq_red : arr3;
        cve_tipo_red : char;
        end;
RegistroCan= Record
        matricula_can : arr6;
        end;
var
estado_bt1, estado_bt3, estado_bt4, estado_bt5, estado_bt6,
estado_bt7, estado_bt8, estado_bt9, estado_bt10, estado_bt11,
estado_bt12, estado_bt13 : integer;
passwd,anio,mes,dia : string;
ProgramName,ProgramName2,ProgramName3 : string[79];
aborto,funciona_red : boolean;
hr,min,sec,sec100 : word;
year,month,day,dow : word;
arr_mat,arr_matricula,arr_matri_apa,arr_mat_red : arr6;
long_reg_usu,
long_reg_usu_des,
long_reg_apa,
long_reg_maq,
long_reg_ser,
long_reg_tus,
long_reg_tma,
long_reg_hor,
long_reg_ocu,
long_reg_lun,
long_reg_mar,
long_reg_mie,
long_reg_jue,
long_reg_vie,
long_reg_sab,
long_reg_est_lun,
long_reg_est_mar,
long_reg_est_mie,
```

```
long_reg_est_jue,
long_reg_est_vie,
long_reg_est_sab,
long_reg_red,
long_reg_can : integer;
bp_usu,
bp_usu_des,
bp_usu_des_f,
bp_apa,
bp_maq,
bp_ser,
bp_tus,
bp_tma,
bp_hor,
bp_ocu,
bp_lun,
bp_mar,
bp_mie,
bp_jue,
bp_vie,
bp_sab,
bp_est_lun,
bp_est_mar,
bp_est_mie,
bp_est_jue,
bp_est_vie,
bp_est_sab,
bp_red,
bp_can : arr128;
bp_usu_f,
bp_apa_f,
bp_maq_f,
bp_ser_f,
bp_tus_f,
bp_tma_f,
bp_hor_f,
bp_ocu_f,
bp_lun_f,
bp_mar_f,
bp_mie_f,
bp_jue_f,
bp_vie_f,
bp_sab_f,
bp_est_lun_f,
bp_est_mar_f,
bp_est_mie_f,
bp_est_jue_f,
bp_est_vie_f,
bp_est_sab_f,
bp_can_f : arr128;
reg_usu : RegistroUsu;
reg_usu_des : RegistroUsuDes;
reg_apa : RegistroApa;
reg_maq : RegistroMaq;
reg_ser : RegistroSer;
reg_tus : RegistroTus;
reg_tma : RegistroTma;
reg_hor : RegistroHor;
reg_ocu : RegistroOcu;
reg_lun : RegistroLun;
reg_mar : RegistroMar;
reg_mie : RegistroMie;
reg_jue : RegistroJue;
reg_vie : RegistroVie;
reg_sab : RegistroSab;
reg_est_lun : RegistroEstLun;
reg_est_mar : RegistroEstMar;
reg_est_mie : RegistroEstMie;
reg_est_jue : RegistroEstJue;
```

```
reg_est_vie : RegistroEstVie;
reg_est_sab : RegistroEstSab;
reg_red : RegistroRed;
reg_can : RegistroCan;
FDA    : boolean;
llaveUsu0 : arr6;
llaveUsu1 : RegLlaveUsu1;
llaveUsu2 : RegLlaveUsu2;
llaveUsuDes : arr6;
llaveMaq1 : RegLlaveMaq1;
llaveApa0 : arr6;
llaveApa1 : RegLlaveApa1;
llaveApa2 : arr2;
llaveApa4 : arr3;
llaveMaq0 : arr3;
llaveSer0 : arr2;
llaveSer1 : arr20;
llaveSer2 : arr15;
llaveSer3 : RegLlaveSer3;
llaveTUs0 : char;
llaveTUs1 : arr20;
llaveTMa0 : char;
llaveTMa1 : arr20;
llaveHor0 : arr2;
llaveOcu0 : arr6;
llaveOcu1 : arr3;
llaveOcu2 : arr2;
llaveOcu3 : char;
llaveLun0 : arr2;
llaveMar0 : arr2;
llaveMie0 : arr2;
llaveJue0 : arr2;
llaveVie0 : arr2;
llaveSab0 : arr2;
llaveEstLun0 : arr2;
llaveEstMar0 : arr2;
llaveEstMie0 : arr2;
llaveEstJue0 : arr2;
llaveEstVie0 : arr2;
llaveEstSab0 : arr2;
llaveCan0    : RegllaveCan0;
arch_usu,
arch_usu_des,
arch_apa,
arch_maq,
arch_ser,
arch_tus,
arch_tma,
arch_hor,
arch_ocu,
arch_lun,
arch_mar,
arch_mie,
arch_jue,
arch_vie,
arch_sab,
arch_est_lun,
arch_est_mar,
arch_est_mie,
arch_est_jue,
arch_est_vie,
arch_est_sab,
arch_can : arr14;
arch_usu_f,
arch_usu_des_f,
arch_apa_f,
arch_maq_f,
arch_ser_f,
arch_tus_f,
```

```
arch_tma_f,
arch_hor_f,
arch_ocu_f,
arch_lun_f,
arch_mar_f,
arch_mie_f,
arch_jue_f,
arch_vie_f,
arch_sab_f,
arch_est_lun_f,
arch_est_mar_f,
arch_est_mie_f,
arch_est_jue_f,
arch_est_vie_f,
arch_est_sab_f,
arch_red,
arch_can_f : arr29;
RegHr : RegistroHr;
RegFec : RegistroFec;
con_apa_0,
con_apa_1,
con_apa_2,
sin_apa_0,
sin_apa_1,
sin_apa_2,
cont_hr_0,
cont_hr_1,
cont_hr_2,
cont_maq_0,
cont_maq_1,
cont_maq_2,
cont_apa_0,
cont_apa_1,
cont_apa_2,
reconstruir,
cuenta_0,
contador    : Integer;
{**********************************************}
```

APENDICE 2

A continuación se presentan los principales módulos desarrollados en Visual Basic.

```
Sub B_APARTADO_Click ()
'Cambia el puntero de ratón a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Menú de Apartados
F_APARMENU.Show
'Descarga la pantalla de Menú Principal
Unload F_MNUPRIN
End Sub

Sub B_CONSULTA_Click ()
'Cambia al apuntador del raton a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Menu de Consultas
F_CONSULTAMENU.Show
'Descarga la pantalla de Menu Principal
Unload F_MNUPRIN
End Sub

Sub B_MANTENIMIENTO_Click ()
'Cambia el puntero de ratón a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Menú de Mantenimiento
F_MANTENIMENU.Show
'Descarga la pantalla de Menú Principal
Unload F_MNUPRIN
End Sub

Sub BC_SALIDA_Click ()
    I_SALIDA_Click
End Sub

Sub Form_Load ()
'Cambia el apuntador del raton a default
Screen.MousePointer = 0
 F_MNUPRIN.Top = 0
 F_MNUPRIN.Left = 0
 F_MNUPRIN.Width = 9705
 F_MNUPRIN.Height = 7320
 ' Realiza el procedimiento de seguridad
 proc_seg_mnuprin
End Sub

Sub I_APARTADO_Click ()
'Cambia el puntero de ratón a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Menu de Apartados
F_APARMENU.Show
'Descarga la pantalla de Menú Principal
Unload F_MNUPRIN
End Sub

Sub I_MANTENIMIENTO_Click ()
 B_MANTENIMIENTO_Click
End Sub
```

```
Sub I_SALIDA_Click ()
 'Carga la portada
 F_PORTADA.Show
 'Descarga Menu Principal
  Unload F_MNUPRIN
End Sub

Sub PA_SALIDA_Click ()
 I_SALIDA_Click
End Sub

Sub proc_seg_mnuprin ()
 Select Case password
 Case NIVEL_2:
   'Esconde Consultas y Respaldos
   I_APARTADO.Visible = False
   B_APARTADO.Visible = False
   I_CONSULTA.Visible = False
   B_CONSULTA.Visible = False
   'Modifica el tamaño al panel
   PA_MNUPRIN.Top = 3480
   PA_MNUPRIN.Width = 975
   PA_MNUPRIN.Left = 720
   PA_MNUPRIN.Height = 1335
   'Reubica la imagen de mantenimiento
   I_MANTENIMIENTO.Top = 360
   I_MANTENIMIENTO.Left = 240
Case NIVEL_3, NIVEL_4:
   'Esconde la opcion de mantenimiento
   I_MANTENIMIENTO.Visible = False
   B_MANTENIMIENTO.Visible = False
 End Select
End Sub

Sub BC_APARTADOS_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de USUARIOS
 PcHorario.Show
 'Descarga la pantalla de Menu de Apartados
 Unload F_APARMENU
End Sub

Sub BC_CANCELAC_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de cancelaciones
 CANCELACION.Show
 'Descarga la pantalla de Menu de Apartado
 Unload F_APARMENU
End Sub

Sub BC_ENTRADA_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de ENTRADAS
 F_ENTRADAS.Show
 'Descarga la pantalla de Menu de Apartado
 Unload F_APARMENU
End Sub

Sub BC_MENUPRIN_Click ()
  'Cambia al apuntador del raton a reloj de arena
   Screen.MousePointer = 11
   F_MNUPRIN.Show
   Unload Me
End Sub
```

```
Sub BC_SALIDA_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de SALIDAS
  F_SALIDAS.Show
 'Descarga la pantalla de Menu de Apartado
 Unload F_APARMENU
End Sub

Sub Form_Load ()
 'Cambia el apuntador del reloj a default
 Screen.MousePointer = 0
 'Inicializa la posicion de la pantalla
 F_APARMENU.Top = 0
 F_APARMENU.Left = 0
 'Inicializa el tamaño de la pantalla
 F_APARMENU.Width = 9705
 F_APARMENU.Height = 7320
 'Llama al procedimiento de seguridad
 PROC_SEG_APARTADO
End Sub

Sub I_MNPRIN_Click ()
   BC_MENUPRIN_Click
End Sub

Sub mnacanc_I2_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de cancelaciones
  CANCELACION.Show
 'Descarga la pantalla de Menu de Apartado
 Unload F_APARMENU
End Sub

Sub mnaentr_I3_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de SALIDAS
  F_SALIDAS.Show
 'Descarga la pantalla de Menu de Apartado
 Unload F_APARMENU
End Sub

Sub mnapart_I1_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de USUARIOS
  PcHorario.Show
 'Descarga la pantalla de Menu de Apartados
 Unload F_APARMENU
End Sub

Sub mnasali_I4_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de ENTRADAS
 F_ENTRADAS.Show
 'Descarga la pantalla de Menu de Apartado
 Unload F_APARMENU
End Sub

Sub PROC_SEG_APARTADO ()
  Select Case password
  Case NIVEL_4:
    'Esconde Cancelaciones
    BC_CANCELAC.Visible = False
    MNACANC_I2.Visible = False
    'Esconde entradas
```

```
    BC_ENTRADA.Visible = False
    MNAENTR_I3.Visible = False
    'Esconde salidas
    BC_SALIDA.Visible = False
    MNASALI_I4.Visible = False
    'Reubica boton de apartados
    BC_APARTADOS.Top = 3240
    BC_APARTADOS.Left = 6840
    'Reubica imagen de apartados
    MNAPART_I1.Top = 3000
    MNAPART_I1.Left = 5640
  End Select
End Sub
'
' Declaracion de variables
'
Dim puede_apartar As String 'Autorizacion para apartar equipo

Sub BC_CONTINUAR_Click ()
 Dim tipmaq As String    'tipo de máquina
 'Valida que la hora proporcionada se mayor a la hora actual
 If (Mid$(c_pchorari.Text, 1, 2)) > (Format$(Now, "hh")) Then
  If Len(TX_MATRICULA) = 0 Then    ' valida que la matricula tenga datos
   MsgBox "Teclear matricula ", 16, ""
  TX_MATRICULA.SetFocus
   Exit Sub
 End If
  'Identifica el TIPO de máquina
  If BO_MAC Then
    tipmaq = "1"
  End If
  If BO_WORKST Then
    tipmaq = "2"
  End If
  If BO_PCVALUE Then
    tipmaq = "3"
  End If
Else
  MsgBox "Proporcione una hora superior a las " + Format$(Now, "hh") + " Hrs.", 64, ""
  c_pchorari.SetFocus
  Exit Sub
End If
  'Llama al procedimiento PROC_ALTA_APARTADO
  PROC_ALTA_APARTADO (TX_MATRICULA.Text), (tipmaq), (Mid$(c_pchorari.Text, 1, 2))
End Sub

Sub BC_MNUPRIN_Click ()
   F_MNUPRIN.Show
   Unload PcHorario
   Unload F_APARMENU
End Sub

Sub BC_REGRESAR_Click ()
 'Cambia el apuntador del reloj a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de Menu de Apartados
 F_APARMENU.Show
 'Descarga la pantalla de Apartados
 Unload PcHorario
End Sub

Sub BO_MAC_Click ()
   L2_PCHORARI.Visible = True
   c_pchorari.Visible = True
   BO_PCVALUE.Value = False
   BO_MAC.Value = True
   BO_WORKST.Value = False
End Sub
```

```
Sub BO_PCVALUE_Click ()
   L2_PCHORARI.Visible = True
   c_pchorari.Visible = True
   BO_PCVALUE.Value = True
   BO_MAC.Value = False
   BO_WORKST.Value = False
End Sub

Sub BO_WORKST_Click ()
   L2_PCHORARI.Visible = True
   c_pchorari.Visible = True
   BO_PCVALUE.Value = False
   BO_MAC.Value = False
   BO_WORKST.Value = True
End Sub

Sub C_PCHORARI_GotFocus ()
'Despliega mensaje de ayuda
TX_AYUDA.Text = "Seleccione el horario "
 BC_CONTINUAR.Enabled = True
End Sub

Sub Form_Load ()
' se cargan datos en el combo de horario
   c_pchorari.AddItem "07:00 - 7:59"
   c_pchorari.AddItem "08:00 - 8:59"
   c_pchorari.AddItem "09:00 - 9:59"
   c_pchorari.AddItem "10:00 - 10:59"
   c_pchorari.AddItem "11:00 - 11:59"
   c_pchorari.AddItem "12:00 - 12:59"
   c_pchorari.AddItem "13:00 - 13:59"
   c_pchorari.AddItem "14:00 - 14:59"
   c_pchorari.AddItem "15:00 - 15:59"
   c_pchorari.AddItem "16:00 - 16:59"
   c_pchorari.AddItem "17:00 - 17:59"
   c_pchorari.AddItem "18:00 - 18:59"
   c_pchorari.AddItem "19:00 - 19:59"
   c_pchorari.AddItem "19:00 - 19:59"
   c_pchorari.AddItem "20:00 - 20:59"
   c_pchorari.AddItem "21:00 - 21:59"
   c_pchorari.AddItem "22:00 - 22:59"
   c_pchorari.AddItem "23:00 - 23:59"
   c_pchorari.ListIndex = 0
'Cambia el apuntador del reloj a default
Screen.MousePointer = 0
'inicializa la posicion de la pantalla
PcHorario.Top = 0
PcHorario.Left = 0
'Inicializa el tamaño de la pantalla
PcHorario.Width = 9705
PcHorario.Height = 7320
'Deshabilita el boton de continuar
BC_CONTINUAR.Enabled = False
End Sub

Sub I_MNPRIN_Click ()
   F_MNUPRIN.Show
   Unload PcHorario
   Unload F_APARMENU
End Sub
Sub Label2_Click ()
   L2_PCHORARI.Visible = True
   c_pchorari.Visible = True
End Sub

Sub TX_MATRICULA_GotFocus ()
'Despliega mensaje de ayuda
TX_AYUDA.Text = "Teclee su matricula y presione enter para continuar "
End Sub
```

```
Sub B0_MAC_Click ()
   L2_PCHORARI.Visible = True
   c_pchorarl.Visible = True
   BO_PCVALUE.Value = False
   BO_MAC.Value = True
   BO_WORKST.Value = False
End Sub

Sub B0_MAC_GotFocus ()
'Despliega mensaje de ayuda
TX_AYUDA.Text = "Selección de un equipo Macintosh"
End Sub

Sub L_MNPRIN_Click ()
   F_MNUPRIN.Show
   Unload PcHorario
   Unload F_APARMENU
End Sub

Sub PROC_ALTA_APARTADO (matricula As String, tipmaq As String, horario As String)
'Parametros: matricula: numero de matricula del usuario
'           tipmaq: tipo de máquina
'           horario: hora del apartado
 Dim instruc As String    'instruccion de acceso a la base de datos
 Dim msg_bt As String     'Mensaje de la base de datos
 Dim num_archi As Integer 'Identificador de archivo
 Dim res_bt As String     'Codigo de respuesta de la base de datos
 Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
 'Forma la instrucción de acceso a la base de datos
 instruc = "F:\USER\APARTA\APART " + matricula + " " + tipmaq + " " + horario
 'Cambia el apuntador del raton a reloj de arena
 Screen.MousePointer = 11
 'Ejecuta el acceso a la base de datos (shell)
 shellval = Shell(instruc)
 'Verifica si el shell ha terminado
 While GetModuleUsage(shellval) > 0
     z% = DoEvents()
 Wend
 'Cambia el apuntador del ratón a default
 Screen.MousePointer = 0
 'Lee el archivo de estatus del shell
 num_archi = FreeFile
 Open "F:\USER\APARTA\APART.TXT" For Input As #num_archi
 Input #num_archi, res_bt, msg_bt
 Close #num_archi
 'Modifica el mensage de BTRIEVE
 If msg_bt = "EL USUARIO TIENE OTRO APARTADOA O ESTA EN EL C.E.C." Then
    msg_bt = "EL USUARIO TIENE OTRO APARTADO o ESTA EN EL C.E.C."
 End If
 'Despliega el mensage de estatus del shell
 MsgBox msg_bt, 64, ""
 If res_bt <> BT_SUCCESS Then
    TX_MATRICULA.Text = ""
    F_PCHORARI.Visible = False
    c_pchorarl.Visible = False
    BC_CONTINUAR.Enabled = False
    TX_MATRICULA.SetFocus
 End If
End Sub

Function PROC_VAL_USU () As Integer
  Dim instruc As String    'instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'identificador de archivo
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  Dim est_bt As String     'status de usuario
  Dim matricula As String  'numero de matricula
  Dim paterno As String    'apellido paterno
  Dim materno As String    'apellido materno
```

```
Dim nombre As String    'nombre usuario
Dim TIPO_USU As String  'tipo de usuario
Dim APARTA_SINO As String 'puede apartar S/No
Dim status As String    'xxxxxx
Dim acumula As String   'xxxxxx
Dim CARRERA As String   'Carrera de usuario
Dim mat As String
'Despliega mensaje de ejecucion de busqueda
'TX_AYUDA.ForeColor = &H0&
'TX_AYUDA.Text = "Ejecutando busqueda de usuario " + tx_matricula.Text + " ..."
PROC_VAL_USU = False
'Agrega ceros a la izquierda del numero de matricula
mat = TX_MATRICULA.Text
TX_MATRICULA.Text = DATO_CON_CEROS(mat, 6)
'Forma la instrucción de acceso a la base de datos
instruc = "F:\USER\APARTA\USU " + "4" + " " + TX_MATRICULA.Text
'Cambia el apuntador del raton a reloj de arena
Screen.MousePointer = 11
'Ejecuta el acceso a la base de datos (shell)
shellval = Shell(instruc)
'Verifica si el shell a terminado
While GetModuleUsage(shellval)
    z% = DoEvents()
Wend
'Cambia el apuntador del raton a default
Screen.MousePointer = 0
' Status de la consulta
STARETOK = 0
'Lee el archivo de estatus del shell
num_archi = FreeFile
Open "F:\USER\APARTA\USU.TXT" For Input As #num_archi
Input #num_archi, est_bt, matricula, paterno, materno, nombre, TIPO_USU, APARTA_SINO,          status, acumula, CARRERA
'Input #num_archi, est_bt, APARTA_SINO
Close #num_archi
puede_apartar = APARTA_SINO
If est_bt = "0" And APARTA_SINO = "S" Then
  PROC_VAL_USU = True
End If
End Function

Sub BC_CONTINUAR_Click ()
  If Len(TX_MATRICULA) = 0 Then    ' valida que matricula tenga datos
    MsgBox "Teclear matricula ", 16, ""
    TX_MATRICULA.SetFocus
    Exit Sub
  End If
  ' Call PROC_APARTADO_CANCELACION(Val(TX_MATRICULA.Text))
End Sub

Sub Command1_Click ()
  MsgBox "Se realizó cancelación del apartado", 64, ""
  'CANCELACION.Hide
'  Apartados.Hide
'  Matricula.Hide
'  Unload Apartados
'  Unload Matricula
'  Unload Me
  'F_MNUPRIN.Show
End Sub

Sub Command2_Click ()
  MsgBox "Cancelación abortada", 64, ""
  'CANCELACION.Hide
  'F_APARMENU.Hide
'    Unload Apartados
'  Unload Matricula
  Unload Me
  'F_APARMENU.Show
End Sub
```

```
Sub L_RETMENUPRIN_Click ()
L_MNPRIN_Click
End SubSub L_MNPRIN_Click ()
    Unload Me
    Unload F_APARMENU
    F_MNUPRIN.Show
End Sub

Sub PROC_CANCELA_APARTADO (matricula As String)
'Parametros: matricula: numero de matricula del usuario
'           tipmaq: tipo de máquina
'           horario: hora del apartado
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim res_bt As String     'Codigo de respuesta de la base de datos
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)

  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\CANCELA " + matricula + " " + tipmaq + " " + horario
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell ha terminado
  While GetModuleUsage(shellval) > 0
      z% = DoEvents()
  Wend
  'Cambia el apuntador del raton a default
  Screen.MousePointer = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTA\CANCELA.TXT" For Input As #num_archi
  Input #num_archi, res_bt, msg_bt
  Close #num_archi
  'Despliega el mensage de estatus del shell
  MsgBox msg_bt, 64, ""
  If res_bt <> BT_SUCCESS Then
     TX_MATRICULA.Text = ""
     P_CANCELAC.Visible = False
     TX_MATRICULA.SetFocus
  End If
End Sub

Function PROC_VAL_USU () As Integer
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  Dim est_bt As String     'status de usuario
  Dim matricula As String  'numero de matricula
  Dim paterno As String    'apellido paterno
  Dim materno As String    'apellido materno
  Dim nombre As String     'nombre usuario
  Dim TIPO_USU As String   'tipo de usuario
  Dim APARTA_SINO As String 'puede apartar SI/No
  Dim status As String     'xxxxxx
  Dim acumula As String    'xxxxxx
  Dim CARRERA As String    'Carrera de usuario
  'Despliega mensaje de ejecucion de busqueda
  'TX_AYUDA.ForeColor = &H0&
  'TX_AYUDA.Text = "Ejecutando busqueda de usuario " + tx_matricula.Text + " ..."
  PROC_VAL_USU = False
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\USU " + "4" + " " + TX_MATRICULA.Text
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
```

```
'Verifica si el shell a terminado
While GetModuleUsage(shellval)
    z% = DoEvents()
Wend
'Cambia el apuntador del raton a default
Screen.MousePointer = 0
' Status de la consulta
STARETOK = 0
'Lee el archivo de estatus del shell
num_archi = FreeFile
Open "F:\USER\APARTA\USU.TXT" For Input As #num_archi
Input #num_archi, est_bt, matricula, paterno, materno, nombre, TIPO_USU, APARTA_SINO, status, acumula, CARRERA
'Input #num_archi, est_bt, APARTA_SINO
Close #num_archi
If est_bt = "0" And APARTA_SINO = "S" Then
   PROC_VAL_USU = True
End If
End Function

Sub BC_MNUPRIN_Click ()
   Unload Me
   Unload F_APARMENU
   F_MNUPRIN.Show
End Sub

Sub BC_REGRESAR_Click ()
   Unload Me
   F_APARMENU.Show
End Sub

Sub C1_CANCELACSI_Click ()
   PROC_CANCELA_APARTADO (TX_MATRICULA.Text)
   P_CANCELAC.Visible = False
   TX_MATRICULA.SetFocus
End Sub

Sub C2_CANCELACNO_Click ()
   MsgBox "Cancelación abortada", 64, ""
   P_CANCELAC.Visible = False
   TX_MATRICULA.Text = ""
   TX_MATRICULA.SetFocus
End Sub

Sub Form_Load ()
'Cambia el apuntador del reloj a default
Screen.MousePointer = 0
'Inicializa la posicion de la pantalla
Cancelacion.Top = 0
Cancelacion.Left = 0
'Inicializa el tamaño de la pantalla
Cancelacion.Width = 9705
Cancelacion.Height = 7320
End Sub

Sub TX_MATRICULA_GotFocus ()
'Despliega mensaje de ayuda
TX_AYUDA.Text = "Teclee su matricula y presione enter para continuar "
End Sub

'
' Declaracion de variables
'
Dim puede_apartar As String 'Autorizacion para apartar equipo

Sub L_MNPRIN_Click ()
   F_MNUPRIN.Show
   Unload F_Entradas
   Unload F_APARMENU
End Sub
```

```
Function Proc_Entrada1 (mat As String, hora As String) As String
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  Dim est_bt As String     'status de usuario
  Dim num_maq As String    'Número de Máquina Asignada
  'Agrega ceros a la izquierda de la matricula
  mat = TX_MATRICULA.Text
  mat = DATO_CON_CEROS(mat, 6)
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\ENTRADA1 " + " " + mat + " " + hora
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell a terminado
  While GetModuleUsage(shellval)
      z% = DoEvents()
  Wend
  'Cambia el apuntador del raton a default
  Screen.MousePointer = 0
  ' Status de la consulta
  STARETOK = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTA\ENTRA1.TXT" For Input As #num_archi
  Input #num_archi, est_bt, num_maq
  Close #num_archi
  'Discriminación de Mensajes
  If (est_bt <> "3") And (est_bt <> "0") Then
    MsgBox num_maq, 16, ""
  End If
  'Actualización del contador de entradas para el caso de entradas con apartado previo
  If est_bt = "0" Then
    L_MAQASIG.Text = num_maq
    Num_entradas = Num_entradas + 1
  End If
 'Asignamos el valor  de regreso de la llamada a la rutina de Entradas
  Proc_Entrada1 = est_bt
End Function

Function PROC_VAL_USU () As Integer
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  Dim est_bt As String     'status de usuario
  Dim Matricula As String  'numero de matricula
  Dim paterno As String    'apellido paterno
  Dim materno As String    'apellido materno
  Dim nombre As String     'nombre usuario
  Dim TIPO_USU As String   'tipo de usuario
  Dim APARTA_SINO As String 'puede apartar SI/No
  Dim status As String     'xxxxxx
  Dim acumula As String    'xxxxxx
  Dim CARRERA As String    'Carrera de usuario
  Dim mat As String        'Matricula del usuario
  'Despliega mensaje de ejecucion de busqueda
  'TX_AYUDA.ForeColor = &H0&
  'TX_AYUDA.Text = "Ejecutando busqueda de usuario " + tx_matricula.Text + " ..."
  PROC_VAL_USU = False
  'Agrega ceros a la izquierda del numero de matricula
  mat = TX_MATRICULA.Text
  TX_MATRICULA.Text = DATO_CON_CEROS(mat, 6)
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\USU " + "4" + " " + TX_MATRICULA.Text
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
```

```
'Ejecuta el acceso a la base de datos (shell)
shellval = Shell(insiruc)
'Verifica si el shell a terminado
While GetModuleUsage(shellval)
    z% = DoEvents()
Wend
'Cambia el apuntador del raton a default
Screen.MousePointer = 0
' Status de la consulta
STARETOK = 0
'Lee el archivo de estatus del shell
num_archi = FreeFile
Open "F:\USER\APARTA\USU.TXT" For Input As #num_archi
Input #num_archi, est_bt, Matricula, paterno, materno, nombre, TIPO_USU, APARTA_SINO, status, acumula, CARRERA
'Input #num_archi, est_bt, APARTA_SINO
Close #num_archi
puede_apartar = APARTA_SINO
If est_bt = "0" And APARTA_SINO = "S" Then
  PROC_VAL_USU = True
End If
End Function

Sub BC_CONTINUAR_Click ()
TX_MATRICULA = ""    'Inicializa campos
L_RETRASO = ""
L_MAQASIG = ""
  'Deshabilita el boton de continuar
  BC_CONTINUAR.Enabled = False
  P_CONTROL.Visible = False
  L_RETRASO.Visible = False
  L_MAQASIG.Visible = False
  TX_MATRICULA.SetFocus
End Sub

Sub BC_MNUPRIN_Click ()
  F_MNUPRIN.Show
  Unload F_Entradas
  Unload F_APARMENU
End Sub

Sub BC_REGRESAR_Click ()
  F_APARMENU.Show
  Unload F_Entradas
End Sub

Sub Form_Load ()
'Cambia el apuntador del reloj a default
Screen.MousePointer = 0
'Inicializa la posicion de la pantalla
F_Entradas.Top = 0
F_Entradas.Left = 0
'Inicializa el tamaño de la pantalla
F_Entradas.Width = 9705
F_Entradas.Height = 7320
'Deshabilita el boton de continuar
BC_CONTINUAR.Enabled = False
'Deshabilita el panel y etiquetas
P_CONTROL.Visible = False
L_MAQASIG.Visible = False
L_RETRASO.Visible = False
End Sub

Sub TX_MATRICULA_GotFocus ()
 'Despliega mensaje de ayuda
 TX_AYUDA.Text = "Teclee su matricula y presione enter para continuar "
'habilita el panel y etiquetas     P_CONTROL.Visible = False
 L_MAQASIG.Visible = False
 L_RETRASO.Visible = False
End Sub
```

```
Sub B0_MAC_Click ()
   L2_PCHORARI.Visible = True
   BO_PCVALUE.Value = False
   BO_MAC.Value = True
   BO_WORKST.Value = False
End Sub

Sub B0_MAC_GotFocus ()
 'Despliega mensaje de ayuda
 TX_AYUDA.Text = "Selección de un equipo Macintosh"
End Sub

Sub Proc_Entrada2 (tx_matricula As String, hora As String, tipomaq As String)
'Parametros: matricula: numero de matricula del usuario
'           tipmaq: tipo de máquina
'           horario: hora del apartado
  Dim instruc As String     'Instruccion de acceso a la base de datos
  Dim msg_bt As String      'Mensaje de la base de datos
  Dim num_archi As Integer  'Identificador de archivo
  Dim res_bt As String      'Codigo de respuesta de la base de datos
  Dim shellval As Integer   'Valor de retorno del shell (instance handle/identificador)
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\ENTRADA2 " + tx_matricula + " " + hora + " " + tipomaq
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell ha terminado
  While GetModuleUsage(shellval) > 0
      z% = DoEvents()
  Wend
  'Cambia el apuntador del raton a default
  Screen.MousePointer = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTA\ENTRA2.TXT" For Input As #num_archi
  Input #num_archi, res_bt, msg_bt
  Close #num_archi
  'Discriminación de Mensajes
  If res_bt <> "0" Then
    MsgBox msg_bt, 64, ""
  End If
  If res_bt = "0" Then
    MsgBox "MAQUINA ASIGNADA " + msg_bt, 16, ""
  End If
 'Actualización del contador de entradas para el caso de entradas con apartado previo
  If res_bt = "0" Then
    Num_entradas = Num_entradas + 1
  End If
  F_ENTRADAS.Show
  Unload F_Entrada2
End Sub

Sub BC_CONTINUAR_Click ()
Dim tipmaq As String    'tipo de máquina
  If Len(tx_matricula) = 0 Then    ' valida que la matricula tenga datos
    MsgBox "Teclear matricula ", 16, ""
    tx_matricula.SetFocus
    Exit Sub
  End If
  'Identifica el TIPO de máquina
  If BO_MAC Then
    tipmaq = "1"
  End If
  If BO_WORKST Then
    tipmaq = "2"
  End If
  If BO_PCVALUE Then
    tipmaq = "3"
```

```
  End If
  'Llama al procedimiento de entradas 2
  Proc_Entrada2 (tx_matricula.Text), (Format$(Now, "hh")), (tipmaq)
End Sub

Sub BC_REGRESAR_Click ()
 'Cambia el apuntador del reloj a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de Menu de Apartados
 F_APARMENU.Show
 'Descarga la pantalla de Apartados
 Unload F_Entrada2
End Sub

Sub BO_MAC_Click ()
   L2_PCHORARI.Visible = True
   BO_PCVALUE.Value = False
   BO_MAC.Value = True
   BO_WORKST.Value = False
End Sub

Sub BO_PCVALUE_Click ()
   L2_PCHORARI.Visible = True
   BO_PCVALUE.Value = True
   BO_MAC.Value = False
   BO_WORKST.Value = False
End Sub

Sub BO_WORKST_Click ()
   L2_PCHORARI.Visible = True
   BO_PCVALUE.Value = False
   BO_MAC.Value = False
   BO_WORKST.Value = True
End Sub

Sub Form_Load ()
   'Cambia el apuntador del reloj a default
   Screen.MousePointer = 0
   'Inicializa la posicion de la pantalla
   F_Entrada2.Top = 0
   F_Entrada2.Left = 0
   'Inicializa el tamaño de la pantalla
   F_Entrada2.Width = 9705
   F_Entrada2.Height = 7320
   'Inicializa los Elementos de La pantalla
   BC_CONTINUAR.Enabled = True
   tx_matricula.Text = matricula
   F_PCHORARI.Visible = True
   L2_PCHORARI.Caption = "Para la Hora Actual: " + Format$(Now, "hh") + " Hrs."
End Sub

Sub L_MNPRIN_Click ()
   F_MNUPRIN.Show
   Unload F_Entrada2
   Unload F_APARMENU
End Sub

Sub TX_MATRICULA_GotFocus ()
 'Despliega mensaje de ayuda
 TX_AYUDA.Text = "Teclee su matricula y presione enter para continuar "
End Sub

Sub L_MNPRIN_Click ()
   F_MNUPRIN.Show
   Unload F_SALIDAS
   Unload F_APARMENU
End Sub
```

```
Sub PROC_SALIDA (matricula As String)
'Parametros: matricula: numero de matricula del usuario
'            tipmaq: tipo de máquina
'            horario: hora del apartado
 Dim instruc As String    'instruccion de acceso a la base de datos
 Dim msg_bt As String     'Mensaje de la base de datos
 Dim num_archi As Integer 'identificador de archivo
 Dim res_bt As String     'Codigo de respuesta de la base de datos
 Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
 'Forma la instrucción de acceso a la base de datos
 instruc = "F:\USER\APARTA\SALIDA " + matricula
 'Cambia el apuntador del raton a reloj de arena
 Screen.MousePointer = 11
'Ejecuta el acceso a la base de datos (shell)
 shellval = Shell(instruc)
'Verifica si el shell ha terminado
 While GetModuleUsage(shellval) > 0
    z% = DoEvents()
 Wend
 'Cambia el apuntador del raton a default
 Screen.MousePointer = 0
 'Lee el archivo de estatus del shell
 num_archi = FreeFile
 Open "F:\USER\APARTA\SALIDA.TXT" For Input As #num_archi
 If Not EOF(num_archi) Then
   Input #num_archi, res_bt, msg_bt
   Close #num_archi
 Else
   msg_bt = "SALIDA EFECTUADA"
 End If
 'Despliega el mensage de estatus del shell
 MsgBox msg_bt, 64, ""
End Sub

Function PROC_VAL_USU () As Integer
 Dim instruc As String    'instruccion de acceso a la base de datos
 Dim msg_bt As String     'Mensaje de la base de datos
 Dim num_archi As Integer 'identificador de archivo
 Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
 Dim est_bt As String     'status de usuario
 Dim matricula As String  'numero de matricula
 Dim paterno As String    'apellido paterno
 Dim materno As String    'apellido materno
 Dim nombre As String     'nombre usuario
 Dim TIPO_USU As String   'tipo de usuario
 Dim APARTA_SINO As String 'puede apartar SI/No
 Dim status As String     'xxxxxx
 Dim acumula As String    'xxxxxx
 Dim CARRERA As String    'Carrera de usuario
 Dim mat As String
 'Despliega mensaje de ejecucion de busqueda
 'TX_AYUDA.ForeColor = &H0&
 'TX_AYUDA.Text = "Ejecutando busqueda de usuario " + tx_matricula.Text +"..."
 PROC_VAL_USU = False
 'Agrega ceros a la izquierda del numero de matricula
 mat = TX_MATRICULA.Text
 TX_MATRICULA.Text = DATO_CON_CEROS(mat, 6)
 'Forma la instrucción de acceso a la base de datos
 instruc = "F:\USER\APARTA\USU " + "4" + " " + TX_MATRICULA.Text
 'Cambia el apuntador del raton a reloj de arena
 Screen.MousePointer = 11
 'Ejecuta el acceso a la base de datos (shell)
 shellval = Shell(instruc)
 'Verifica si el shell a terminado
 While GetModuleUsage(shellval)
    z% = DoEvents()
 Wend
 'Cambia el apuntador del raton a default
 Screen.MousePointer = 0
```

```
'Inicializa el tamaño de la pantalla
F_MANTENIMENU.Width = 9705
F_MANTENIMENU.Height = 7320
End Sub

Sub I_MNPRIN_Click ()
'Cambia al apuntador del raton a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Menu Principal
F_MNUPRIN.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub I_SERVIDORES_Click ()
'Llama al procedimiento BC_SERVIDORES_Click
BC_SERVIDORES_Click
End Sub

Sub I_TIPMAQUINA_Click ()
'Llama al procedimiento BC_TIPMAQUINA_Click
BC_TIPMAQUINA_Click
End Sub

Sub I_USUARIOS_Click ()
'Llama al procedimiento BC_USUARIOS_Click
BC_USUARIOS_Click
End Sub

Sub BC_TERMINAR_Click ()
  F_MNUPRIN.Show
  Unload F_MAQUINA
End Sub

Sub BO_ALTAS_Click ()
  PA_CAMPOS_MAQUINA.Visible = True
  TX_NUM_MAQ.SetFocus
End Sub

Sub BO_BAJAS_Click ()
  PA_CAMPOS_MAQUINA.Visible = False
End Sub

Sub BO_CONSULTAS_Click ()
  PA_CAMPOS_MAQUINA.Visible = False
End Sub

Sub I_MODELO_VALUE_Click ()
 OB_MODELO_MAC = True
End Sub

Sub I_RETMENUPRIN_Click ()
  F_MNUPRIN.Show
  Unload F_MAQUINA
End Sub

Sub I_VALUE_Click ()
  OB_MODELO_PC = True
End Sub

Sub I_WORK_Click ()
  OB_MODELO_WORK = True
End Sub

Sub OB_MODELO_MAC_GotFocus ()
If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Seleccione una opción"
Else
  TX_AYUDA.Text = "Seleccione modificación"
```

```
Sub L_SALMANTE_Click ()
'Cambia al apuntador del raton a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Menu Principal
F_MNUPRIN.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub BC_MAQUINAS_Click ()
'Cambia el puntero de ratón a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de MAQUINAS
F_MAQUINA.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub BC_MENUPRIN_Click ()
'Cambia al apuntador del raton a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Menu Principal
F_MNUPRIN.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub BC_SERVIDORES_Click ()
'Cambia el puntero de ratón a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de SERVIDORES
F_SERVIDORES.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub BC_TIPMAQUINA_Click ()
'Cambia el puntero de ratón a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla TIPOS DE MAQUINA
F_TIPMAQUINAS.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub BC_TIPUSUARIO_Click ()
'Cambia el puntero de ratón a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla TIPOS DE USUARIO
F_TIPUSUARIOS.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub BC_USUARIOS_Click ()
'Cambia el puntero de ratón a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de USUARIOS
F_USUARIOS.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub Form_Load ()
'Cambia el apuntador del reloj a default
Screen.MousePointer = 0
'Inicializa la posicion de la pantalla
F_MANTENIMENU.Top = 0
F_MANTENIMENU.Left = 0
```

```
'Inicializa el tamaño de la pantalla
F_MANTENIMENU.Width = 9705
F_MANTENIMENU.Height = 7320
End Sub

Sub I_MNPRIN_Click ()
'Cambia al apuntador del raton a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Menu Principal
F_MNUPRIN.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_MANTENIMENU
End Sub

Sub I_SERVIDORES_Click ()
'Llama al procedimiento BC_SERVIDORES_Click
BC_SERVIDORES_Click
End Sub

Sub I_TIPMAQUINA_Click ()
'Llama al procedimiento BC_TIPMAQUINA_Click
BC_TIPMAQUINA_Click
End Sub

Sub I_USUARIOS_Click ()
'Llama al procedimiento BC_USUARIOS_Click
BC_USUARIOS_Click
End Sub

Sub BC_TERMINAR_Click ()
  F_MNUPRIN.Show
  Unload F_MAQUINA
End Sub

Sub BO_ALTAS_Click ()
  PA_CAMPOS_MAQUINA.Visible = True
  TX_NUM_MAQ.SetFocus
End Sub

Sub BO_BAJAS_Click ()
  PA_CAMPOS_MAQUINA.Visible = False
End Sub

Sub BO_CONSULTAS_Click ()
    PA_CAMPOS_MAQUINA.Visible = False
End Sub

Sub I_MODELO_VALUE_Click ()
  OB_MODELO_MAC = True
End Sub

Sub I_RETMENUPRIN_Click ()
  F_MNUPRIN.Show
  Unload F_MAQUINA
End Sub

Sub I_VALUE_Click ()
  OB_MODELO_PC = True
End Sub

Sub I_WORK_Click ()
  OB_MODELO_WORK = True
End Sub

Sub OB_MODELO_MAC_GotFocus ()
If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Seleccione una opción"
Else
  TX_AYUDA.Text = "Seleccione modificación"
```

```
End If
End Sub

Sub OB_MODELO_PC_GotFocus ()
If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Seleccione una opción"
Else
  TX_AYUDA.Text = "Seleccione modificación"
End If
End SubSub OB_MODELO_WORK_GotFocus ()
If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Seleccione una opción"
Else
  TX_AYUDA.Text = "Seleccione modificación"
End If
End Sub

Sub PROC_CATALOGO_MAQUINAS ()
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim res_bt As Integer    'Codigo de respuesta de la base de datos
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  Dim CLAVE As String      'Clave del catalogo de usuarios
  Dim descri As String     'Descripcion de la clave de catalogo de usuarios
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\CATAMAQ"
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell a terminado
  While GetModuleUsage(shellval)
      z% = DoEvents()
  Wend
  'Cambia el apuntador del ratón a default
  Screen.MousePointer = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTA\CATAMAQ.TXT" For Input As #num_archi
  While Not EOF(num_archi)
    Input #num_archi, CLAVE, DES
    CB_TIPO.AddItem CLAVE + " " + DES
  Wend
  Close #num_archi
End Sub

Sub PROC_LEE_CATALOGO_MAQUINAS (TIPO_MAQ As String)
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim res_bt As Integer    'Codigo de respuesta de la base de datos
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  Dim CLAVE As String      'Clave del catalogo de usuarios
  Dim descri As String     'Descripcion de la clave de catalogo de usuarios
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\TMAQ " + "4" + " " + TIPO_MAQ
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell a terminado
  While GetModuleUsage(shellval)
      z% = DoEvents()
  Wend
  'Cambia el apuntador del raton a default
  Screen.MousePointer = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
```

```
  Open "F:\USER\APARTA\TMA.TXT" For Input As #num_archi
  Input #num_archi, res_bt, CLAVE, DES
    CB_TIPO = CLAVE + " " + DES
  Close #num_archi
End Sub

Sub PROC_LIMPIA_CAMPOS_MAQUINA ()
   'Inicializa los campos de la máquina
   TX_NUM_MAQ = ""
   TX_CPU.Text = ""
   TX_RAM.Text = ""
   TX_TECLADO.Text = ""
   TX_HD.Text = ""
   TX_MONITOR.Text = ""
   TX_CAPACIDAD.Text = ""
   TX_MOUSE.Text = ""
   TX_MICRO.Text = ""
   TX_VIDEO.Text = ""
   TX_SERVIDOR.Text = ""
   TX_ISLA.Text = ""
   TX_TIPO = ""
   TX_MODELO = ""
   B_SER_S! = True
End Sub

Sub PROC_MAQ_ALTA (tip_oper As String, NUM_MAQ As String, tipo As String, cpu As String, teclado As String, monitor As String,
MOUSE As String, modelo As String, ram As String, hd As String, micro As String, video As String, fd As String, servidor As String, isla
As String, servi As String)
  Dim instruc As String     'Instruccion de acceso a la base de datos
  Dim msg_bt As String      'Mensaje de la base de datos
  Dim num_archi As Integer  'Identificador de archivo
  Dim res_bt As Integer     'Codigo de respuesta de la base de datos
  Dim shellval As Integer   'Valor de retorno del shell (instance handle/identificador)
  Dim habil  As String      ' ? para que lo usa JC
  Dim sone   As String      ' ? para que lo usa JC
 'Agrega ceros al numero de máquina
  NUM_MAQ = DATO_CON_CEROS(NUM_MAQ, 3)
  TX_NUM_MAQ = NUM_MAQ
  ' Status de la consulta
  STARETOK = 0
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\MAQ " + tip_oper + " " + NUM_MAQ + " " + tipo + " " + cpu + " " + teclado + " " + monitor + " " + MOUSE
+ " " + modelo + " " + ram + " " + hd + " " + micro + " " + video + " " + fd + " " + servidor + " " + isla + " " + servi
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell a terminado
  While GetModuleUsage(shellval)
      z% = DoEvents()
  Wend
  'Cambia el apuntador del raton a default
  Screen.MousePointer = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTA\MAQ.TXT" For Input As #num_archi
  Input #num_archi, res_bt, msg_bt
  Close #num_archi
 'Despliega el resultado de la alta
  If res_bt = BTR_SUCCESS Then
     ' La alta fue exitosa
     STARETOK = 1
  End If
  'Despliega el mensage de estatus del shell
  MsgBox msg_bt, 64, ""
End Sub
```

```
Sub PROC_MAQ_BAJA (tip_oper As String, NUM_MAQ As String)
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim res_bt As Integer    'Codigo de respuesta de la base de datos
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  'Agrega ceros al numero de máquina
   NUM_MAQ = DATO_CON_CEROS(NUM_MAQ, 3)
   TX_NUM_MAQ = NUM_MAQ
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\MAQ " + tip_oper + " " + NUM_MAQ
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell a terminado
  While GetModuleUsage(shellval)
      z% = DoEvents()
  Wend
  'Cambia el apuntador del raton a default
  Screen.MousePointer = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTA\MAQ.TXT" For Input As #num_archi
  Input #num_archi, res_bt, msg_bt
  Close #num_archi
  'Despliega el mensage de estatus del shell
  MsgBox msg_bt, 64, ""
End Sub

Sub PROC_MAQ_CAMBIO (tip_oper As String, NUM_MAQ As String, tipo As String, cpu As String, teclado As String, monitor As
String, MOUSE As String, modelo As String, ram As String, hd As String, micro As String, video As String, fd As String, servidor As
String, isla As String, servi As String)
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim res_bt As Integer    'Codigo de respuesta de la base de datos
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  'Agrega ceros al numero de máquina
   NUM_MAQ = DATO_CON_CEROS(NUM_MAQ, 3)
   TX_NUM_MAQ = NUM_MAQ
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\MAQ " + tip_oper + " " + NUM_MAQ + " " + tipo + " " + cpu + " " + teclado + " " + monitor + " " + MOUSE
+ " " + modelo + " " + ram + " " + hd + " " + micro + " " + video + " " + fd + " " + servidor + " " + isla + " " + servi
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell a terminado
  While GetModuleUsage(shellval)
      z% = DoEvents()
  Wend
  'Cambia el apuntador del raton a default
  Screen.MousePointer = 0
End Sub

Sub PROC_MAQ_CONSULTA (tip_oper As String, NUM_MAQ As String)
  Dim instruc As String    'Instruccion de acceso a la base de datos
  Dim msg_bt As String     'Mensaje de la base de datos
  Dim num_archi As Integer 'Identificador de archivo
  Dim shellval As Integer  'Valor de retorno del shell (instance handle/identificador)
  'Agrega ceros al numero de máquina
   NUM_MAQ = DATO_CON_CEROS(NUM_MAQ, 3)
   TX_NUM_MAQ = NUM_MAQ
   ' Status de la consulta
  STARETOK = 0
  'Despliega mensaje de ejecucion de consulta
  TX_AYUDA.ForeColor = &H0&
  TX_AYUDA.Text = "Ejecutando consulta de la máquina " + TX_NUM_MAQ.Text + " ..."
```

```
'Forma la instrucción de acceso a la base de datos
instruc = "F:\USER\APARTA\MAQ " + tip_oper + " " + NUM_MAQ
'Cambia el apuntador del raton a reloj de arena
Screen.MousePointer = 11
'Ejecuta el acceso a la base de datos (shell)
shellval = Shell(instruc)
'Verifica si el shell a terminado
While GetModuleUsage(shellval)
    z% = DoEvents()
Wend
'Cambia el apuntador del raton a default
Screen.MousePointer = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTA\MAQ.TXT" For Input As #num_archi
  Input #num_archi, est_bt, NUM_MAQ, tipo, cpu, teclado, monitor, MOUSE, modelo, ram, hd, micro, video, fd, servidor, isla, servi
  Close #num_archi
  'Despliega el resultado de la la consulta
  If est_bt = BTR_SUCCESS Then
    ' La consulta fue exitosa
    STARETOK = 1
  'Limpia los valores de los campos del servidor
    PROC_LIMPIA_CAMPOS_MAQUINA
    'Haz visible los campos de la máquina
    PA_CAMPOS_MAQUINA.Visible = True
    'Asigna el valor de los campos recuperados a los campos de la máquina
    TX_NUM_MAQ.Text = NUM_MAQ
    TX_CPU.Text = cpu
    TX_TECLADO.Text = teclado
    TX_MONITOR.Text = monitor
    TX_MOUSE.Text = MOUSE
    TX_RAM.Text = ram
    TX_HD.Text = hd
    TX_MICRO.Text = micro
    TX_VIDEO.Text = video
    TX_CAPACIDAD.Text = fd
    TX_SERVIDOR.Text = servidor
    TX_ISLA.Text = isla
    TX_MODELO = modelo
    ' Lee el catalogo de tipos de usuario
    PROC_LEE_CATALOGO_MAQUINAS (tipo)
    If servi = "S" Then
      B_SER_SI = True
    Else
      B_SER_NO = True
    End If
  Else
    'Despliega el mesage de estatus del shell
    MsgBox NUM_MAQ, 64, ""
  End If
End Sub

Sub TX_TIPO_GotFocus ()
If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Tecle el tipo de máquina"
Else
  TX_AYUDA.Text = "Tecle modificación al tipo de máquina"
End If
End Sub

Sub TX_TIPO_KeyPress (KeyAscii As Integer)
  Dim CHAR As String
  CHAR = UCase(Chr(KeyAscii))
  KeyAscii = Asc(CHAR)
  Select Case KeyAscii
  Case 13:                    ' enter
    If TX_TIPO = "" Then
      MsgBox "Dato requerido ", 16, ""
      CB_TIPO.SetFocus
```

```
      Exit Sub
    End If
  Case 9:                    'tab
    If TX_TIPO = "" Then
      MsgBox "Dato requerido ", 16, ""
      CB_TIPO.SetFocus
      Exit Sub
    End If
  Case 8:                    'backspace
  Case Else
    CHAR = Chr$(KeyAscii)
    If Len(TX_TIPO) = 1 Then
      B_SER_SI.SetFocus
    End If
  End Select
End Sub

Sub VALIDA_CAMPOS ()
  If TX_CPU.Text = "" Then
      MsgBox "Teclear número de serie de CPU", 16, ""
      TX_CPU.SetFocus
      Exit Sub
    End If
    If TX_TECLADO.Text = "" Then
      MsgBox "Teclear número de serie del teclado", 16, ""
      TX_TECLADO.SetFocus
      Exit Sub
    End If
    If TX_MONITOR.Text = "" Then
      MsgBox "Teclear número de serie del monitor", 16, ""
      TX_MONITOR.SetFocus
      Exit Sub
    End If
    If TX_MOUSE.Text = "" Then
      MsgBox "Teclear número de serie del ratón", 16, ""
      TX_MOUSE.SetFocus
      Exit Sub
    End If
    If TX_RAM.Text = "" Then
      MsgBox "Teclear la capacidad RAM", 16, ""
      TX_RAM.SetFocus
      Exit Sub
    End If
    If TX_HD.Text = "" Then
      MsgBox "Teclear la capacidad del disco duro ", 16, ""
      TX_HD.SetFocus
      Exit Sub
    End If
    If TX_CAPACIDAD.Text = "" Then
      MsgBox "Teclear la densidad del floppy disk", 16, ""
      TX_CAPACIDAD.SetFocus
      Exit Sub
    End If
    If TX_MICRO.Text = "" Then
      MsgBox "Teclear tipo de microprocesador", 16, ""
      TX_MICRO.SetFocus
      Exit Sub
    End If
    If TX_VIDEO.Text = "" Then
      MsgBox "Teclear número de tarjeta de video ", 16, ""
      TX_VIDEO.SetFocus
      Exit Sub
    End If
    If Len(TX_SERVIDOR) = 0 Then
      MsgBox "Teclear número de servidor en donde se encuentra la máquina", 16, ""
      TX_SERVIDOR.SetFocus
      Exit Sub
    End If
    If TX_ISLA.Text = "" Then
```

```
    MsgBox "Teclear número de isla a la que pertenece la máquina", 16, ""
    TX_ISLA.SetFocus
    Exit Sub
  End If
  If TX_TIPO = "" Then
    MsgBox "Teclear tipo de máquina", 16, ""
    CB_TIPO.SetFocus
    Exit Sub
  End If
End Sub

Sub B_SER_NO_GotFocus ()
 If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Seleccione una opción"
 Else
  TX_AYUDA.Text = "Seleccione modificación"
 End If
End Sub

Sub B_SER_SI_GotFocus ()
If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Seleccione una opción"
Else
  TX_AYUDA.Text = "Seleccione modificación"
End If
End Sub

Sub BC_CONTINUAR_Click ()
 Const BTN_ACEP = 1      'Seleccion de boton ACEPTAR
 Const BTR_SUCCESS = "0"  'Aceso a BTRIEVE con exito
' Static FLGVEZ1 As String 'Para los cambios. Primero consulta y desp. cambio
 Dim est_bt As String    'Estatus de acceso a BTRIEVE
 Dim NUM_MAQ As String   'Número de máquina devuelta por BTRIEVE
 Dim hay_baja As Integer 'Confirmacion de baja de registro
 Dim cpu As String       'CPU devuelto por BTRIEVE
 Dim ram As String       'RAM devuelta por BTRIEVE
 Dim teclado As String   'Teclado devuelto por BTRIEVE
 Dim hd As String        'Capacidad de HD devuelto por BTRIEVE
 Dim monitor As String   'Monitor devuelto por BTRIEVE
 Dim capacidad As String 'Capacidad devuelta por BTRIEVE
 Dim MOUSE As String     'Mouse devuelto por BTRIEVE
 Dim micro As String     'Microprocesador devuelto por BTRIEVE
 Dim video As String     'Video devuelto por BTRIEVE
 Dim servidor As String  'Servidor devuelto por BTRIEVE
 Dim isla As String      'Isla devuelta por BTRIEVE
 Dim tipo As String      'Tipo devuelto por BTRIEVE
 Dim servi As String     'En servicio o no devuelta por BTRIEVE
 Dim modelo As String    'Modelo de máquina devuelta por BTRIEVE
 Dim TX_TIPO As String   'Conversion del tipo de maquina
 Dim num_archi As Integer 'Identificador de archivo
 Dim msg_bt              'Mensaje de la base de datos
 Dim tip_oper As String  'Tipo de operación del usuario
 'Conversiones para btrieve del tipo de máquina
 TX_TIPO = Mid$(CB_TIPO.Text, 1, 1)
 If Len(TX_NUM_MAQ) = 0 Then     ' valida que número de máquina tenga datos
   MsgBox "Teclear número de maquina", 16, ""
   TX_NUM_MAQ.SetFocus
   Exit Sub
 End If
 If B_SER_SI Then
   servi = "S"
 Else
   servi = "N"
 End If
'Identifica el tipo de acceso a la base de datos (alta, baja, consulta o cambio)
  If BO_ALTA Then
    If TX_CPU.Text = "" Then
      MsgBox "Teclear número de serie de CPU", 16, ""
      TX_CPU.SetFocus
```

```
    Exit Sub
  End If
  If TX_TECLADO.Text = "" Then
    MsgBox "Teclear número de serie del teclado", 16, ""
    TX_TECLADO.SetFocus
    Exit Sub
  End If
  If TX_MONITOR.Text = "" Then
    MsgBox "Teclear número de serie del monitor", 16, ""
    TX_MONITOR.SetFocus
    Exit Sub
  End If
  If TX_MOUSE.Text = "" Then
    MsgBox "Teclear número de serie del ratón", 16, ""
    TX_MOUSE.SetFocus
    Exit Sub
  End If
  If TX_RAM.Text = "" Then
    MsgBox "Teclear la capacidad RAM", 16, ""
    TX_RAM.SetFocus
    Exit Sub
  End If
  If TX_HD.Text = "" Then
    MsgBox "Teclear la capacidad del disco duro ", 16, ""
    TX_HD.SetFocus
    Exit Sub
  End If
  If TX_CAPACIDAD.Text = "" Then
    MsgBox "Teclear la densidad del floppy disk", 16, ""
    TX_CAPACIDAD.SetFocus
    Exit Sub
  End If
  If TX_MICRO.Text = "" Then
    MsgBox "Teclear tipo de microprocesador", 16, ""
    TX_MICRO.SetFocus
    Exit Sub
  End If
  If TX_VIDEO.Text = "" Then
    MsgBox "Teclear número de tarjeta de video ", 16, ""
    TX_VIDEO.SetFocus
    Exit Sub
  End If
  If Len(TX_SERVIDOR) = 0 Then
    MsgBox "Teclear número de servidor en donde se encuentra la máquina", 16, ""
    TX_SERVIDOR.SetFocus
    Exit Sub
  End If
  If TX_ISLA.Text = "" Then
    MsgBox "Teclear número de isla a la que pertenece la máquina", 16, ""
    TX_ISLA.SetFocus
    Exit Sub
  End If
  If TX_TIPO = "" Then
    MsgBox "Teclear tipo de máquina", 16, ""
    CB_TIPO.SetFocus
    Exit Sub
  End If
  tip_oper = "1"          'Identifica una Alta
  'Despliega mensaje de ejecucion de alta
  TX_AYUDA.ForeColor = &H0&
  TX_AYUDA.Text = "Ejecutando alta de máquina" + TX_NUM_MAQ.Text + "..."
  'Llama el acceso a la base de datos con la operacion ALTA
  PROC_MAQ_ALTA (tip_oper), (TX_NUM_MAQ.Text), (TX_TIPO), (TX_CPU.Text), (TX_TECLADO.Text), (TX_MONITOR.Text),
(TX_MOUSE.Text), (TX_MODELO.Text), (TX_RAM.Text), (TX_HD.Text), (TX_MICRO.Text), (TX_VIDEO.Text),
(TX_CAPACIDAD.Text), (TX_SERVIDOR.Text), (TX_ISLA.Text), (servi)
  If STARETOK = 1 Then
    'Cambia el color de mensaje de ayuda
    TX_AYUDA.ForeColor = &H80&
    TX_NUM_MAQ.Text = ""
```

```
      'Listo para una alta
      BO_ALTA_Click
    End If
    'Coloca el foco en el número de máquina
    TX_NUM_MAQ.SetFocus
    Exit Sub
  End If
If BO_BAJA Then
  'primero hace la consulta
  tip_oper = "4"
  PROC_MAQ_CONSULTA (tip_oper), (TX_NUM_MAQ.Text)
  If STARETOK = 1 Then      ' La consulta es exitosa
    tip_oper = "2"          'Identifica una baja
    'Confirma la baja del usuario
    TX_AYUDA.Text = ""
    msgbox_text1 = "¿ Procede la baja ?"
    F_MSGBOX.Show MODO
    If FLG_MSGBOX = 1 Then   'SI procede
      'Despliega mensaje de ejecucion de baja
      TX_AYUDA.ForeColor = &H0&
      TX_AYUDA.Text = "Ejecutando baja de máquina " + TX_NUM_MAQ.Text + " ..."
      'Llama el acceso a la base de datos con la operacion BAJA
      PROC_MAQ_BAJA (tip_oper), (TX_NUM_MAQ.Text)
      'Cambia el color de mensaje de ayuda
      TX_AYUDA.ForeColor = &H80&
    End If
  End If
  'Listo para otra baja
  BO_BAJA_Click
  Exit Sub
End If
If BO_CAMBIO Then
  If FLGVEZ1 <> "1" Then
    'primero hace la consulta
    tip_oper = "4"
    PROC_MAQ_CONSULTA (tip_oper), (TX_NUM_MAQ.Text)
    If STARETOK = 1 Then
      ' Si la consulta fue exitosa
      FLGVEZ1 = "1"
      TX_NUM_MAQ.Enabled = False
      PA_CAMPOS_MAQUINA.Enabled = True
      TX_CPU.SetFocus
    Else
      TX_NUM_MAQ.Text = ""
      TX_NUM_MAQ.SetFocus
    End If
    Exit Sub
  End If
  'Valida datos requeridos
  If TX_CPU.Text = "" Then
    MsgBox "Teclear número de serie de CPU", 16, ""
    TX_CPU.SetFocus
    Exit Sub
  End If
  If TX_TECLADO.Text = "" Then
    MsgBox "Teclear número de serie del teclado", 16, ""
    TX_TECLADO.SetFocus
    Exit Sub
  End If
  If TX_MONITOR.Text = "" Then
    MsgBox "Teclear número de serie del monitor", 16, ""
    TX_MONITOR.SetFocus
    Exit Sub
  End If
  If TX_MOUSE.Text = "" Then
    MsgBox "Teclear número de serie del ratón", 16, ""
    TX_MOUSE.SetFocus
    Exit Sub
  End If
```

```
    If TX_RAM.Text = "" Then
      MsgBox "Teclear la capacidad RAM", 16, ""
      TX_RAM.SetFocus
      Exit Sub
    End If
    If TX_HD.Text = "" Then
      MsgBox "Teclear la capacidad del disco duro ", 16, ""
      TX_HD.SetFocus
      Exit Sub
    End If
    If TX_CAPACIDAD.Text = "" Then
      MsgBox "Teclear la densidad del floppy disk", 16, ""
      TX_CAPACIDAD.SetFocus
      Exit Sub
    End If
    If TX_MICRO.Text = "" Then
      MsgBox "Teclear tipo de microprocesador", 16, ""
      TX_MICRO.SetFocus
      Exit Sub
    End If
    If TX_VIDEO.Text = "" Then
      MsgBox "Teclear número de tarjeta de video ", 16, ""
      TX_VIDEO.SetFocus
      Exit Sub
    End If
    If Len(TX_SERVIDOR) = 0 Then
      MsgBox "Teclear número de servidor en donde se encuentra la máquina", 16, ""
      TX_SERVIDOR.SetFocus
      Exit Sub
    End If
    If TX_ISLA.Text = "" Then
      MsgBox "Teclear número de isla a la que pertenece la máquina", 16, ""
      TX_ISLA.SetFocus
      Exit Sub
    End If
    If TX_TIPO = "" Then
      MsgBox "Teclear tipo de máquina", 16, ""
      CB_TIPO.SetFocus
      Exit Sub
    End If
    'Conversiones para btrieve del tipo de máquina
    TX_TIPO = Mid$(CB_TIPO.Text, 1, 1)
  'Identifica un cambio
  tip_oper = "3"
  'Despliega mensaje de ejecucion de cambio
  TX_AYUDA.ForeColor = &H0&
  TX_AYUDA.Text = "Ejecutando cambio de máquina " + TX_NUM_MAQ.Text + "..."
  'Llama el acceso a la base de datos con la operacion CAMBIO
  PROC_MAQ_CAMBIO (tip_oper), (TX_NUM_MAQ.Text), (TX_TIPO), (TX_CPU.Text), (TX_TECLADO.Text),
(TX_MONITOR.Text), (TX_MOUSE.Text), (TX_MODELO.Text), (TX_RAM.Text), (TX_HD.Text), (TX_MICRO.Text), (TX_VIDEO.Text),
(TX_CAPACIDAD.Text), (TX_SERVIDOR.Text), (TX_ISLA.Text), (servi)
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTAMAQ.TXT" For Input As #num_archi
  Input #num_archi, est_bt, msg_bt
  Close #num_archi
  'Despliega el resultado del cambio
  MsgBox msg_bt, 64, ""
  'Limpia y cambia el color de mensaje de ayuda
  TX_AYUDA.Text = ""
  TX_AYUDA.ForeColor = &H80&
  TX_NUM_MAQ.Enabled = False
  TX_NUM_MAQ.Text = ""
  FLGVEZ1 = "0"
  'listo para un cambio
  BO_CAMBIO_Click
  Exit Sub
End If
If BO_CONSULTA Then
```

```
    tip_oper = "4"          'Identifica una consulta
    'Llama el acceso a la base de datos con la operacion CONSULTA
    PROC_MAQ_CONSULTA (tip_oper), (TX_NUM_MAQ.Text)
    If STARETOK = 1 Then     ' Si la consulta fue exitosa
      'Limpia y cambia el color de mensaje de ayuda
      TX_AYUDA.Text = ""
      TX_AYUDA.ForeColor = &H80&
      'Inhabilita los datos
      PA_CAMPOS_MAQUINA.Enabled = False
      'Coloca el foco en el numero de máquina
      TX_NUM_MAQ.SetFocus
    Else
      'Inicializa campos para una nueva consulta
      BO_CONSULTA_Click
    End If
  End If
End Sub

Sub BC_MNUPRIN_Click ()
  F_MNUPRIN.Show
  Unload F_MAQUINA
End Sub

Sub BC_REGRESAR_Click ()
  F_MANTENIMENU.Show
  Unload F_MAQUINA
End Sub

Sub BO_ALTA_Click ()
  PROC_LIMPIA_CAMPOS_MAQUINA
  If PA_CAMPOS_MAQUINA.Visible = False Then
    PA_CAMPOS_MAQUINA.Visible = True
  End If
  PA_CAMPOS_MAQUINA.Enabled = True
  TX_NUM_MAQ.Enabled = True
  FLGVEZ1 = "0"
  TX_NUM_MAQ.SetFocus
End Sub

Sub BO_BAJA_Click ()
  PROC_LIMPIA_CAMPOS_MAQUINA
  If PA_CAMPOS_MAQUINA.Visible = True Then
    PA_CAMPOS_MAQUINA.Visible = False
  End If
  TX_NUM_MAQ.Enabled = True
  FLGVEZ1 = "0"
  TX_NUM_MAQ.SetFocus
End Sub

Sub BO_CAMBIO_Click ()
  If FLGVEZ1 <> "1" Then  ' no se ha hecho la consulta
    PA_CAMPOS_MAQUINA.Visible = False
    PROC_LIMPIA_CAMPOS_MAQUINA
  End If
  TX_NUM_MAQ.Enabled = True
  TX_NUM_MAQ.SetFocus
End Sub

Sub BO_CONSULTA_Click ()
  PROC_LIMPIA_CAMPOS_MAQUINA
  PA_CAMPOS_MAQUINA.Visible = False
  TX_NUM_MAQ.Enabled = True
  FLGVEZ1 = "0"
  TX_NUM_MAQ.SetFocus
End SubSub CB_TIPO_GotFocus ()
  If FLGVEZ1 = "0" Then
    TX_AYUDA.Text = "Seleccione tipo de máquina"
  Else
    TX_AYUDA.Text = "Seleccione modificación"
```

```
  End If
End Sub

Sub Form_Load ()
 'Cambia el apuntador del reloj a default
 Screen.MousePointer = 0
 'Inicializa la posicion de la pantalla
 F_MAQUINA.Top = 0
 F_MAQUINA.Left = 0
 'Inicializa el tamaño de la pantalla
 F_MAQUINA.Width = 9705
 F_MAQUINA.Height = 7320
 FLGVEZ1 = "0"
 'Carga catalogo de tipos de máquinas
 PROC_CATALOGO_MAQUINAS
End Sub

Sub TX_ALTA_Click ()
   BO_ALTA = True
   BO_ALTA_Click
End Sub

Sub TX_BAJA_click (Index As Integer)
   BO_BAJA = True
   BO_BAJA_Click
End Sub

Sub TX_CAMBIO_Click ()
   BO_CAMBIO = True
   BO_CAMBIO_Click
End Sub

Sub TX_CAPACIDAD_Change ()
   Dim CHAR As String
   CHAR = UCase(Chr(KeyAscii))
   KeyAscii = Asc(CHAR)
End Sub

Sub TX_CONSULTA_Click ()
   BO_CONSULTA = True
   BO_CONSULTA_Click
End Sub

Sub TX_CPU_GotFocus ()
If FLGVEZ1 = "0" Then
   TX_AYUDA.Text = "Teclee número de CPU"
Else
   TX_AYUDA.Text = "Teclee modificación al número de CPU"
End If
End Sub

Sub TX_HD_Change ()
   Dim CHAR As String
   CHAR = UCase(Chr(KeyAscii))
   KeyAscii = Asc(CHAR)
End Sub

Sub TX_ISLA_Change ()
   Dim CHAR As String
   CHAR = UCase(Chr(KeyAscii))
   KeyAscii = Asc(CHAR)
End Sub

Sub TX_MICRO_Change ()
   Dim CHAR As String
   CHAR = UCase(Chr(KeyAscii))
   KeyAscii = Asc(CHAR)
   Select Case KeyAscii
   Case 13:                    ' enter
```

```
    If TX_MICRO = "" Then
     MsgBox "Dato requerido ", 16, ""
     TX_MICRO.SetFocus
     Exit Sub
    End If
  Case 9:                    ' tab
    If TX_MICRO = "" Then
     MsgBox "Dato requerido ", 16, ""
     TX_MICRO.SetFocus
     Exit Sub
    End If
  Case 8:                    ' backspace
 ' Case Else
 '   CHAR = Chr$(KeyAscii)
 '   If Len(TX_MICRO) = 6 Then
 '     TX_VIDEO.SetFocus
 '   End If
  End Select
End Sub

Sub TX_MODELO_Change ()
  Dim CHAR As String
  CHAR = UCase(Chr(KeyAscii))
  KeyAscii = Asc(CHAR)
End Sub

Sub TX_MONITOR_Change ()
  Dim CHAR As String
  CHAR = UCase(Chr(KeyAscii))
  KeyAscii = Asc(CHAR)
End Sub

Sub TX_MOUSE_Change ()
  Dim CHAR As String
  CHAR = UCase(Chr(KeyAscii))
  KeyAscii = Asc(CHAR)
End Sub

Sub TX_NUM_MAQ_GotFocus ()
If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Teclee el número de máquina"
Else
  TX_AYUDA.Text = "Teclee modificación al número de máquina"
End If
End SubSub TX_RAM_Change ()
  Dim CHAR As String
  CHAR = UCase(Chr(KeyAscii))
  KeyAscii = Asc(CHAR)
End Sub

Sub TX_SERVIDOR_Change ()
  Dim CHAR As String
  CHAR = UCase(Chr(KeyAscii))
  KeyAscii = Asc(CHAR)
End Sub

Sub TX_TECLADO_Change ()
  Dim CHAR As String
  CHAR = UCase(Chr(KeyAscii))
  KeyAscii = Asc(CHAR)
End Sub

Sub TX_VIDEO_GotFocus ()
If FLGVEZ1 = "0" Then
  TX_AYUDA.Text = "Teclee el tipo de tarjeta de video"
Else
  TX_AYUDA.Text = "Teclee modificación al tipo de tarjeta de video"
End If
End Sub
```

```
Sub BC_MENUPRIN_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de Menu Principal
 F_MNUPRIN.Show
 'Descaga la pantalla de Menu de Consultas
 Unload F_CONSULTAMENU
End Sub

Sub GRAF1_Click ()
BC_ESTADISTICAS_Click
End Sub

Sub I_DISCO_Click ()
 'Cambia el puntero de raton a eloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de Respaldos
 F_RESPALDOS.Show
 'Descarga la pantalla de Menu de Mantenimiento
 Unload F_CONSULTAMENU
End Sub

Sub I_RETMENUPRIN_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de Menu Principal
 F_MNUPRIN.Show
 'Descarga la pantalla del Menu de Consultas
 Unload F_CONSULTAMENU
End Sub

Sub Image1_Click ()
BC_ESTADISTICAS_Click
End Sub

Sub L_SALCONSULTAMENU_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
 'Carga la pantalla de Menu Principal
 F_MNUPRIN.Show
 'Descaga la pantalla de Menu de Consultas
 Unload F_CONSULTAMENU
End Sub

Sub P_SALCONMENU_Click ()
Unload F_CONSULTAMENU
End Sub

Sub PROC_SEG_CONSULTA ()
  Select Case password
  Case NIVEL_4:
    'Esconde Graficas
    BC_ESTADISTICAS.Visible = False
    GRAPH1.Visible = False
    'Esconde Respaldos
    BC_RESPALDOS.Visible = False
    I_RESPALDO.Visible = False
    'Reubica boton de Consultas
    BC_CONSULTAS.Top = 3000
    BC_CONSULTAS.Left = 6762.331
    'Reubica imagen de Consultas
    I_USUARIOS.Top = 2880
    I_USUARIOS.Left = 5434.16
  End Select
End Sub

Sub BC_CONSULTAS_Click ()
 'Cambia el puntero de ratón a reloj de arena
 Screen.MousePointer = 11
```

```
'Carga la pantalla de Consultas
F_CONSULTAS.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_CONSULTAMENU
End Sub

Sub BC_ESTADISTICAS_Click ()
'Cambia el puntero de raton a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Estadisticas
F_ESTADISTICAS.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_CONSULTAMENU
End Sub

Sub BC_MNUPRIN_Click ()
   'Cambia al apuntador del raton a reloj de arena
   Screen.MousePointer = 11
   F_MNUPRIN.Show
   Unload Me
End Sub

Sub BC_RESPALDOS_Click ()
'Cambia el puntero de raton a reloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Respaldos
F_RESPALDOS.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_CONSULTAMENU
End Sub

Sub Form_Load ()
'Cambia el apuntador del raton a default
Screen.MousePointer = 0
 F_CONSULTAMENU.Top = 0
 F_CONSULTAMENU.Left = 0
 F_CONSULTAMENU.Width = 9705
 F_CONSULTAMENU.Height = 7320
'Llama al procedimiento de seguridad
PROC_SEG_CONSULTA
End Sub

Sub Graph1_Click ()
   BC_ESTADISTICAS_Click
End Sub

Sub I_respaldo_Click ()
'Cambia el puntero de raton a eloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Respaldos
F_RESPALDOS.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_CONSULTAMENU
End Sub

Sub I_respaldo_Click ()
'Cambia el puntero de raton a eloj de arena
Screen.MousePointer = 11
'Carga la pantalla de Respaldos
F_RESPALDOS.Show
'Descarga la pantalla de Menu de Mantenimiento
Unload F_CONSULTAMENU
End Sub

Sub Image1_Click ()
BC_ESTADISTICAS_Click
End Sub

Sub I_RETMENUPRIN_Click ()
```

```
    F_MNUPRIN.Show
    Unload F_CONSULTAS
End Sub

Sub Image2_Click ()
L_SALCONSULTA_Click
End Sub

Sub L_SALCONSULTA_Click ()
    F_MNUPRIN.Show
    Unload F_CONSULTAS
End Sub

Sub PROC_LIMPIA_CAMPOS_CONSULTA ()
    ' Limpia campos de consulta
    TX_N_MAQUINA.Text = ""
    TX_HORA.Text = ""
End Sub

Sub BC_CONTINUAR_Click ()
  Const BTN_ACEPT = 1          'Seleccion de boton Aceptar
  Const BTR_SUCCESS = "0"      'Acceso a BTRIEVE con éxito
  Dim est_bt    As String      'Estatus de acceso a BTRIVE
  Dim matricula As String      'Matricula devuelta por Btrieve
  Dim n_maquina As Integer     'Numero de maquina devuelto por BTRIEVE
  Dim hora      As String      'Hora devuelta por BTRIEVE
  Dim tip_oper  As String      'tipo de operación del usuario
  Dim instruc   As String      'instruccion de acceso a la base de datos
  Dim msg_bt    As String      'Mensaje de la base de datos
  Dim num_archi As Integer     'Identificador de archivo
  Dim shellval  As Integer     'Valor de retorno del shell (instance handle/identificador)
  Dim paterno     As String    'Apellido paterno devuelto por BTRIEVE
  Dim materno     As String    'Apellido materno devuelto por BTRIEVE
  Dim nombre      As String    'Nombre devuelto por BTRIEVE
  Dim TIPO_USU    As String    'Tipo de usuario devuelto por BTRIEVE
  Dim CARRERA     As String    'Carrera devuelta por BTRIEVE
  Dim APARTA_SINO As String    'Pude apartar o no el usuario BTRIEVE
  Dim status      As String
  Dim acumula     As String
  tip_oper = "4"
  TX_N_MAQUINA = ""
  TX_HORA = ""
  hora = ""
    If Len(tx_matricula) = 0 Then     ' valida que matricula tenga datos
      MsgBox "Teclear matricula ", 16, ""
      tx_matricula.SetFocus
      Exit Sub
    End If
  'Despliega mensaje de ejecucion de consulta
  TX_AYUDA.ForeColor = &H0&
  '-----------------------------------
  'Forma la instrucción de acceso a la base de datos
  instruc = "F:\USER\APARTA\USU " + "4" + " " + tx_matricula
  'Cambia el apuntador del raton a reloj de arena
  screen.MousePointer = 11
  'Ejecuta el acceso a la base de datos (shell)
  shellval = Shell(instruc)
  'Verifica si el shell a terminado
  While GetModuleUsage(shellval)
      z% = DoEvents()
  Wend
  'Cambia el apuntador del raton a default
  screen.MousePointer = 0
  ' Status de la consulta
  STARETOK = 0
  'Lee el archivo de estatus del shell
  num_archi = FreeFile
  Open "F:\USER\APARTA\USU.TXT" For Input As #num_archi
  Input #num_archi, est_bt, matricula, paterno, materno, nombre, TIPO_USU, APARTA_SINO, status, acumula, CARRERA
```

```
Close #num_archi
'Despliega el resultado de la la consulta
If est_bt = BTR_SUCCESS Then
   ' La consulta fue exitosa
'-------------------------
   TX_AYUDA.Text = "Ejecutando consulta del usuario " + tx_matricula.Text + "..."
'Forma la instrucción de acceso a la base de datos
'instruc = "F:\USER\APARTA\USU " + tip_oper + " " + tx_matricula
instruc = "F:\USER\APARTA\consulta " + tx_matricula
'Cambia el apuntador del raton a reloj de arena
screen.MousePointer = 11
'==========================================
'Ejecuta el acceso a la base de datos (shell)
shellval = Shell(instruc)
'Verifica si el shell a terminado
While GetModuleUsage(shellval)
    z% = DoEvents()
Wend
'Cambia el apuntador del raton a default
screen.MousePointer = 0
'==========================================
'Lee el archivo de estatus del shell
num_archi = FreeFile
Open "F:\USER\APARTA\consulta.TXT" For Input As #num_archi
Input #num_archi, est_bt, n_maquina, hora
Close #num_archi
'Despliega el resultado de la consulta
    If est_bt = BTR_SUCCESS Then
    'Limpia los valores de los campos del servidor
       PROC_LIMPIA_CAMPOS_CONSULTA
       'Asigna el valor de los campos recuperados a los campos del servidor
       TX_N_MAQUINA.Text = n_maquina
       TX_HORA.Text = hora
       'Haz visible los campos del servidor
       PA_CAMPOS_USUARIO.Visible = True
    Else
       'Haz invisible los campos del servidor
       PA_CAMPOS_USUARIO.Visible = False
       'Despliega el mensage de estatus del shell
       MsgBox "NO TIENE MAQUINA ASIGNADA " + tx_matricula, 64, ""
       tx_matricula.Text = ""
    End If
Else
       MsgBox "NO EXISTE LA MATRICULA " + tx_matricula, 64, ""
       tx_matricula.Text = ""
End If
     'Limpia y cambia el color de mensaje de ayuda
     TX_AYUDA.Text = ""
     TX_AYUDA.ForeColor = &H80&
     ' Inhabilita el panel
     PA_CAMPOS_USUARIO.Enabled = False
     'Coloca el foco en el número de matricula
     tx_matricula.SetFocus
End Sub

Sub BC_MNUPRIN_Click ()
   F_MNUPRIN.Show
   Unload F_CONSULTAS
End Sub

Sub BC_REGRESAR_Click ()
   F_CONSULTAMENU.Show
   Unload F_CONSULTAS
End Sub

Sub Form_Load ()
 'Cambia el apuntador de reloj de arena a apuntador
 screen.MousePointer = 0
 F_CONSULTAS.Top = 0
```

```
 F_CONSULTAS.Left = 0
 F_CONSULTAS.Width = 9705
 F_CONSULTAS.Height = 7320
End Sub

Sub TX_MATRICULA_GotFocus ()
'Despliega mensaje de ayuda
 TX_AYUDA.ForeColor = &H80&
 TX_AYUDA.Text = "Teclee el Número matrícula"
End Sub

Sub bc_regresar ()
 'Cambia el puntero de ratón a reloj de arena
 screen.MousePointer = 11
 'Carga la pantalla de Menu de Mantenimiento
 F_CONSULTAMENU.Show
 'Descarga la pantalla de Respaldos
 Unload F_RESPALDOS
End Sub

Sub BC_REGRESAR_Click ()
 'Cambia el puntero de ratón a reloj de arena
 screen.MousePointer = 11
 'Carga la pantalla de Menu de Mantenimiento
 F_CONSULTAMENU.Show
 'Descarga la pantalla de Respaldos
 Unload F_RESPALDOS
End Sub

Sub BC_TERMINAR_Click ()
 'F_RESPALDOS.MousePointer = 11  'Cambia el puntero de ratón a reloj de arena
 F_MNUPRIN.Show                 'Carga la pantalla de Menú Principal
 Unload F_RESPALDOS             'Descarga la pantalla de TIPMAQUINAS
End Sub

Sub L_RETMENUPRIN_Click ()
 'Cambia el puntero de ratón a reloj de arena
 F_RESPALDOS.MousePointer = 11
 'Carga la pantalla de Menú Principal
 F_MNUPRIN.Show
 'Descarga la pantalla de TIPMAQUINAS
 Unload F_RESPALDOS
End Sub

Sub Image2_Click ()
L_SALRESPALDO_Click
End Sub

Sub L_SALRESPALDO_Click ()
 'F_RESPALDOS.MousePointer = 11  'Cambia el puntero de ratón a reloj de arena
 F_MNUPRIN.Show                 'Carga la pantalla de Menú Principal
 Unload F_RESPALDOS             'Descarga la pantalla de TIPMAQUINAS
End Sub

Sub Option1_Click ()
   bc_aceptar.SetFocus
End Sub

Sub Option1_GotFocus ()
'Despliega mensaje de ayuda
 TX_AYUDA.ForeColor = &H80&
 TX_AYUDA.Text = "Elija una opción"
End Sub

Sub BC_ACEPTAR_Click ()
   Dim instruc As String      'Instruccion de acceso a la base de dato
   Dim shellval As Integer    'Valor de retorno del shell (instance handle/identificador)
   Dim BOTON As Integer       'Para determinar el valor que regresa el MSG
   Dim parametro_1 As String  'Para enviar el nombre de los archivos a copiar como parámetro
```

```
  Dim parametro_2 As String   'Para enviar el nombre de los archivos a copiar como parámetro
  On Error GoTo Verificar_driver
  BOTON% = MsgBox("SE REALIZA EL RESPALDO", 4 + 32 + 0, "")
  If BOTON = 6 Then
     TX_AYUDA.ForeColor = &H0&
     TX_AYUDA.Text = "REALIZANDO RESPALDO "
     'Cambia el apuntador del raton a reloj de arena
     screen.MousePointer = 11
     parametro_1 = "F:\USER\APARTA\*.TXT"
     parametro_2 = Dir1.Path
     instruc = "F:\USER\APARTA\RESPALDO.BAT " + parametro_1 + " " + parametro_2
     TX_AYUDA.Text = "REALIZANDO RESPALDO    " + instruc
     shellval = Shell(instruc)
     'Verifica si el shell a terminado
     While GetModuleUsage(shellval)
        z% = DoEvents()
     Wend
     'Cambia el apuntador del raton a reloj de arena
     screen.MousePointer = 11
     parametro_1 = "F:\USER\APARTA\*.DAT"
     parametro_2 = Dir1.Path
     instruc = "F:\USER\APARTA\RESPALDO.BAT " + parametro_1 + " " + parametro_2
     TX_AYUDA.Text = "REALIZANDO RESPALDO    " + instruc
     shellval = Shell(instruc)
     'Verifica si el shell a terminado
     While GetModuleUsage(shellval)
        z% = DoEvents()
     Wend
  End If
  TX_AYUDA = "ESCOGE OPCION"
  ' Cambia el apuntador del raton a default
  screen.MousePointer = 0
  TX_AYUDA.ForeColor = &H80&
Exit Sub

Verificar_driver:
   Numeroerror = Err
   Select Case Numerror
   Case 71:
     MsgBox "Error: UUUUUUNIDAD NO PREPARADA ", 48, " "
     Drive1.Drive = "c:"
     Dir1.Path = Drive1.Drive
End Select
End Sub

Sub BC_MNUPRIN_Click ()
  'F_RESPALDOS.MousePointer = 11  'Cambia el puntero de ratón a reloj de arena
  F_MNUPRIN.Show              'Carga la pantalla de Menú Principal
  Unload F_RESPALDOS          'Descarga la pantalla de TIPMAQUINAS
End Sub

Sub Command2_Click ()
  'Cambia el puntero de ratón a reloj de arena
  screen.MousePointer = 11
  'Carga la pantalla de Menu de Mantenimiento
  F_CONSULTAMENU.Show
  'Descarga la pantalla de Respaldos
  Unload F_RESPALDOS
End Sub

Sub Drive1_Change ()
 On Error GoTo Driver
   Dir1.Path = Drive1.Drive
 Exit Sub

Driver:
   MsgBox "Error: UNIDAD NO PREPARADA ", 48, ""
     Drive1.Drive = "c:"
     Dir1.Path = Drive1.Drive
```

```
  Exit Sub
End Sub

Sub Form_Load ()
 screen.MousePointer = 0
 TX_AYUDA.ForeColor = &H80&
' TX_AYUDA.Text = "Eliga una opción"
 F_RESPALDOS.Top = 0
 F_RESPALDOS.Left = 0
 F_RESPALDOS.Width = 9705
 F_RESPALDOS.Height = 7320
End Sub
'
'Declaracion de Constantes
'
 Const HORA_INICIAL = 7        'Hora inicial de servicio del CEC
 Const HORA_FINAL = 23         'Hora final de servicio del CEC
 Const NUM_REN = 9             'Numero de Columnas para tabla de estadisticas (0 a 5)
 Const NUM_COL = 21            'Numero de Renglones para tabla de estadisticas (0 a 15
 Const NSerie_diario = 2       'Numero de series de la grafica de comportamiento diario
 Const NPoints_diario = 6      'Numero de puntos de la grafica de comportamiento diario
 Const NPoints_diario_hora = 17 'Numero de puntos de la grafica de comportamiento por hora
 Const NSerie_semanal = 1      'Numero de series de la grafica de comportamiento semanal
 Const NPoints_semanal = 2     'Numero de puntos de la grafica de comportamiento semanal
'
'Declaracion de Constantes de F_ESTAD.FRM
 Dim array_demanda(HORA_INICIAL To HORA_FINAL, 1 To 2) As Integer 'Matriz de demanda de maquinas
 Dim array_oferta(HORA_INICIAL To HORA_FINAL, 1 To 2) As Integer  'Matriz de oferta de maquinas
 Dim demanda_tot(HORA_INICIAL To HORA_FINAL, 1 To 2) As Integer 'Matriz de totales de demanda de maquinas por columna-hora
 Dim oferta_tot(HORA_INICIAL To HORA_FINAL, 1 To 2) As Integer 'Matriz de totales de oferta de maquinas por columna-hora
 Dim tipo_grafica_cd As Integer 'Tipo de grafica de comportamiento diario
 Dim tipo_grafica_ch As Integer 'Tipo de grafica de comportamiento por hora

Sub BC_AREAS_G1_Click (Value As Integer)
 Graph1.GraphType = 8
 'Despliega la grafica de comportamiento diario
 Graph1.DrawMode = 2
End Sub

Function FileExists (path$) As Integer
  'Verifica si existe el archivo
  X = FreeFile
  On Error Resume Next
  Open path$ For Input As X
  If Err = 0 Then
    FileExists = True
  Else
    FileExists = False
  End If
  Close X
End Function

Function NOM_MES (num_mes As Integer) As String
  'Devuelve el nombre del mes
  Select Case num_mes
      Case 1: NOM_MES = "Ene"
      Case 2: NOM_MES = "Feb"
      Case 3: NOM_MES = "Mar"
      Case 4: NOM_MES = "Abr"
      Case 5: NOM_MES = "May"
      Case 6: NOM_MES = "Jun"
      Case 7: NOM_MES = "Jul"
      Case 8: NOM_MES = "Ago"
      Case 9: NOM_MES = "Sep"
      Case 10: NOM_MES = "Oct"
      Case 11: NOM_MES = "Nov"
      Case 12: NOM_MES = "Dec"
 End Select
End Function
```

```
Sub PROC_DIBUJA_GRAF_COMPORTAMIENTO_DIARIO ()
 Dim c As Integer 'Contador
 Dim r As Integer 'Contador
'Inicializa las caracteristicas de la grafica de compotamiento diario
Graph1.LegendText = "FECHAS"
Graph1.BottomTitle = "FECHAS"
Graph1.LeftTitle = "CANTIDAD"
Graph1.GraphTitle = "Comportamiento Diario de Entradas al C.E.C"
Graph1.NumSets = NSerie_diario
Graph1.NumPoints = NPoints_diario
'Lee las columnas de totales por dia de usuarios y maquinas de la tabla
'de estadisticas
For c = 1 To 2
  For r = 2 To 7
    GR_ESTADISTICAS.Row = r
    GR_ESTADISTICAS.Col = c
    Graph1.GraphData = Val(GR_ESTADISTICAS.Text)
  Next r
Next c
'Tamaño de la fuente para los titulos
 Graph1.FontSize = 200
 'Detalles para los gráficos
 Graph1.LegendText = "Usuarios"
 Graph1.LegendText = "Maquinas"
 'Despliega la Grafica de Comportamiento Diario
 Graph1.DrawMode = 2
End Sub

Sub PROC_DIBUJA_GRAF_COMPORTAMIENTO_HORA ()
 Dim i As Integer 'Contador
'Inicializa las caracteristicas de la grafica de compotamiento diario
Graph2.LegendText = "HORAS"
Graph2.BottomTitle = "HORAS"
Graph2.LeftTitle = ""
Graph2.GraphTitle = "Comportamiento por Hora de Entradas al CEC"
'Asigna el numero de puntos y el numero de serie de la grafica
Graph2.NumSets = NSerie_diario
Graph2.NumPoints = NPoints_diario_hora
'Lee las matrices de totales de columna-hora
For i = HORA_INICIAL To HORA_FINAL
   Graph2.GraphData = demanda_tot(i, 2)
Next i
For i = HORA_INICIAL To HORA_FINAL
   Graph2.GraphData = oferta_tot(i, 2)
Next i
'Tamaño de la fuente para los titulos
 Graph2.FontSize = 200
 'Detalles para los gráficos
 Graph2.LegendText = "Usuarios"
 Graph2.LegendText = "Maquinas"
 'Despliega la Grafica de Comportamiento Diario
 Graph2.DrawMode = 2
End Sub

Sub PROC_ESCRIBE_ESTADISTICAS (renglon As Integer)
'Parametros: renglon: Renglon a escribir de la tabla de estadisticas
 Dim i As Integer        'Contador
 Dim j As Integer        'Contador
 Dim maqdia As Integer   'Numero de maquinas por dia
 Dim usdia As Integer    'Numero de usuarios por dia
 Dim util As Single      'Porcentaje de utilizacion de maquinas en oferta
 'Asigna el renglon a escribir
 GR_ESTADISTICAS.Row = renglon
 'Escribe los datos de oferta y de demanda
 For i = HORA_INICIAL To HORA_FINAL
    'Escribe el numero de usuarios y maquinas
    GR_ESTADISTICAS.Col = i - 3
    GR_ESTADISTICAS.Text = Str$(array_demanda(i, 2)) + " " + Str$(array_oferta(i, 2))
    'Incrementa la columna de los usuarios al dia
```

```
    GR_ESTADISTICAS.Col = 1
    GR_ESTADISTICAS.Text = Str$(array_demanda(i, 2) + Val(GR_ESTADISTICAS.Text))
    'Incrementa la columna de las maquinas al dia
    GR_ESTADISTICAS.Col = 2
    GR_ESTADISTICAS.Text = Str$(array_oferta(i, 2) + Val(GR_ESTADISTICAS.Text))
  Next i
  'Inicializa las variables de usuarios al dia y de maquinas al dia
  usdia = 0
  maqdia = 0
  'Lee de la tabla de estadisticas los usuarios al dia
  GR_ESTADISTICAS.Col = 1
  usdia = Val(GR_ESTADISTICAS.Text)
  'Lee de la tabla de estadisticas las maquinas al dia
  GR_ESTADISTICAS.Col = 2
  maqdia = Val(GR_ESTADISTICAS.Text)
  'Calcula el porcentaje de utilizacion
  GR_ESTADISTICAS.Col = 3
  If maqdia > 0# Then
    GR_ESTADISTICAS.Text = Str$(usdia / maqdia * 100)
  Else
    GR_ESTADISTICAS.Text = "0.0"
  End If
End Sub

Sub PROC_LEE_ARCHIVOS_DEMANDA_OFERTA (archi_dem As String, archi_ofe As String)
'Parametros: archi_dem: Nombre del archivo de demanda de maquinas
'           archi_ofe: Nombre del archivo de oferta de maquinas
Dim existe_archi As Integer  'Bandera que indica si existe un archivo
Dim hora As Integer          'Hora de oferta y de demanda
Dim i As Integer             'Contador
Dim num_archi_dem As Integer 'Identificador de archivo de demanda de maquinas
Dim num_archi_ofe As Integer 'Identificador de archivo de oferta de maquinas
Dim num_maquinas As Integer  'Oferta de maquinas
Dim num_usuarios As Integer  'Demanda de usuarios
'Inicializa las matrices de oferta y demanda de maquinas
For i = HORA_INICIAL To HORA_FINAL
    array_demanda(i, 1) = i
    array_demanda(i, 2) = 0
    array_oferta(i, 1) = i
    array_oferta(i, 2) = 0
Next i
'Verifica si existe el archivo de demanda de usuarios
If FileExists(archi_dem) <> 0 Then
  'Asigna identificador al archivo de demanda de usuarios
  num_archi_dem = FreeFile
  'Lee el archivo de demanda de maquinas
  Open archi_dem For Input As #num_archi_dem
  While Not EOF(num_archi_dem)
      Input #num_archi_dem, hora, usuarios
      array_demanda(hora, 2) = usuarios
      'Incrementa el total por columna-hora para la tabla de estadisticas
      demanda_tot(hora, 2) = demanda_tot(hora, 2) + usuarios
  Wend
  Close #num_archi_dem
End If
'Verifica si existe el archivo de oferta de maquinas
If FileExists(archi_ofe) <> 0 Then
  'Asigna identificador al archivo de oferta de maquinas
  num_archi_ofe = FreeFile
  'Lee el archivo de oferta de maquinas
  Open archi_ofe For Input As #num_archi_ofe
  While Not EOF(num_archi_ofe)
      Input #num_archi_ofe, hora, maquinas
      array_oferta(hora, 2) = maquinas
      'Incrementa el total por columna-hora para la tabla de estadisticas
      oferta_tot(hora, 2) = oferta_tot(hora, 2) + maquinas
  Wend
  Close #num_archi_ofe
End If
```

```
End Sub

Sub PROC_SEMANAS_DE_MES ()
 Dim anio As Integer   'Año del reporte
 Dim dia As Integer    'Dia del reporte
 Dim mes As Integer    'Mes del reporte
 Dim semana As String  'Semana del reporte
 Dim mes_sig As Integer 'Siguiente Mes del reporte
 'Verifica que no se omitan el mes y el año del reporte
 If TX_MES.Text = "" Or TX_ANIO.Text = "" Then
   MsgBox "No omita el mes y el año de reporte", 64, ""
   Exit Sub
 Else
   'Convierte a numero el mes y el año del reporte
   mes = Val(TX_MES.Text)
   anio = Val(TX_ANIO.Text) + 1900
 End If
 'Limpia la lista de las semanas del mes
 CBX_SEMANA.Clear
 'Inicializa el siguiente mes del reporte
 mes_sig = mes
 'Colocate en el primer dia del mes
 dia = 1
 'Calcula las semanas del mes
 While mes = mes_sig
     'Verifica si el dia es lunes
     If Weekday(DateSerial(anio, mes, dia)) = 2 Then
       'Escribe el inicio del periodo
       semana = Format$(DateSerial(anio, mes, dia), "dd") + " " + NOM_MES(mes) + " a "
       'Calcula el mes final del periodo
       mes_sig = Val(Format$(DateSerial(anio, mes, dia + 5), "mm"))
       'Escribe el final del periodo
       semana = semana + Format$(DateSerial(anio, mes, dia + 5), "dd") + " " + NOM_MES(mes_sig)
       'Escribe la semana en la lista de semanas
       CBX_SEMANA.AddItem semana
     End If
     'Calcula el mes del siguiente dia
     mes_sig = Val(Format$(DateSerial(anio, mes, dia + 1), "mm"))
     'Pasa al siguiente dia
     dia = dia + 1
 Wend
 'Despliega la primera semana del mes
 CBX_SEMANA.ListIndex = 0
End Sub

Sub PROC_TOTALES_COLUMNAS_HORA ()
 Dim i 'Contador
 Dim maquinas As Integer  'Numero de usuarios
 Dim usuarios As Integer  'Numero de maquinas
 Dim uti As Single        'Porcentaje de utilizacion
 'Indica el renglon en donde se escriben los totales de columna-hora
 GR_ESTADISTICAS.Row = 8
 For i = HORA_INICIAL To HORA_FINAL
    GR_ESTADISTICAS.Col = i - 3
    GR_ESTADISTICAS.Text = Str$(demanda_tot(i, 2)) + " " + Str$(oferta_tot(i, 2))
 Next i
 'Inicializa el numero de maquinas, de usuarios y porcentaje
 'de utilizacion semanales
 maquinas = 0
 usuarios = 0
 util = 0
 'Calcula el total semanal de maquinas, de usuarios y porcentaje
 'de utilizacion semanales
 For i = 2 To 7
    GR_ESTADISTICAS.Row = i
    GR_ESTADISTICAS.Col = 1
    usuarios = Val(GR_ESTADISTICAS.Text) + usuarios
    GR_ESTADISTICAS.Col = 2
    maquinas = Val(GR_ESTADISTICAS.Text) + maquinas
```

```
    GR_ESTADISTICAS.Col = 3
    util = Val(GR_ESTADISTICAS.Text) + util
  Next i
  'Escribe los totales de maquinas, de usuarios y porcentaje
  'de utilizacion semanales
  GR_ESTADISTICAS.Row = 8
  GR_ESTADISTICAS.Col = 1
  GR_ESTADISTICAS.Text = usuarios
  GR_ESTADISTICAS.Col = 2
  GR_ESTADISTICAS.Text = maquinas
  GR_ESTADISTICAS.Col = 3
  If util > 0# Then
    GR_ESTADISTICAS.Text = util / 6
  Else
    GR_ESTADISTICAS.Text = 0
  End If
End Sub

Sub BC_BARRAS_2D_G1_Click (Value As Integer)
  If Graph1.GraphType = 3 Or Graph1.GraphType = 4 Then
    'Asigna al tipo de grafica Barras 2D
    Graph1.GraphType = 3
    'Despliega la Grafica de Comportamiento Diario
    Graph1.DrawMode = 2
  Else
    'Asigna al tipo de grafica Barras 2D
    Graph1.GraphType = 3
    'Llama al procedimiento PROC_DIBUJA_GRAF_COMPORTAMIENTO_DIARIO
    PROC_DIBUJA_GRAF_COMPORTAMIENTO_DIARIO
  End If
End Sub

Sub BC_EJECUTAR_Click ()
  Dim dem_archi As String     'Archivo de demanda de usuarios
  Dim anio_inicial As Integer  'Dia inicial del reporte
  Dim dia_inicial As Integer   'Dia inicial del reporte
  Dim indice_ren As Integer    'Indice del renglon de la tabla de estadisticas
  Dim mes_inicial As Integer   'Dia inicial del reporte
  Dim ofe_archi As String     'Archivo de demanda de maquinas
  'Cambia el apuntador del raton a reloj de arena
  Screen.MousePointer = 11
  'Identifica la fecha inicial del periodo del reporte
  anio_inicial = Val(TX_ANIO.Text)
  mes_inicial = Val(TX_MES.Text)
  dia_inicial = Val(Mid$(CBX_SEMANA.Text, 1, 2))
  'Escribe los titulos de las columnas de la tabla de estadisticas:
  '(PROC_ESCRIBE_TITULOS)
  PROC_ESCRIBE_TITULOS
  'Inicializa las matrices de totales de columna-hora
  For i = HORA_INICIAL To HORA_FINAL
      demanda_tot(i, 1) = i
      demanda_tot(i, 2) = 0
      oferta_tot(i, 1) = i
      oferta_tot(i, 2) = 0
  Next i
  'Inicializa el numero de renglon inicial a escribir de la tabla
  'de estadisticas
  indice_ren = 2
  For i = 0 To 5
    'Forma el path del archivo de demanda de maquinas
    dem_archi = "F:\USER\APARTA\" + Format$(DateSerial(anio_inicial, mes_inicial, dia_inicial + i), "ddmmyy") + "_D.TXT"
    'Forma el path del archivo de oferta de maquinas
    ofe_archi = "F:\USER\APARTA\" + Format$(DateSerial(anio_inicial, mes_inicial, dia_inicial + i), "ddmmyy") + "_O.TXT"
    'Llama al procedimiento PROC_LEE_ARCHIVOS_DEMANDA_OFERTA
    PROC_LEE_ARCHIVOS_DEMANDA_OFERTA (dem_archi), (ofe_archi)
    'Llama al procedimiento PROC_ESCRIBE_ESTADISTICAS
    PROC_ESCRIBE_ESTADISTICAS (indice_ren)
    'Incrementa el indice del renglon de la tabla de estadisticas
    indice_ren = indice_ren + 1
```

```
Next I
'Escribe los totales de las columnas-hora (PROC_TOTALES_COLUMNAS_HORA)
PROC_TOTALES_COLUMNAS_HORA
'Asigna al tipo de grafica Barras 3D como tipo inicial de la grafica
'de comportamiento diario
Graph1.GraphType = 4
'Llama al procedimiento PROC_DIBUJA_GRAF_COMPORTAMIENTO_DIARIO
PROC_DIBUJA_GRAF_COMPORTAMIENTO_DIARIO
'Asigna al tipo de grafica Barras 3D como tipo inicial de la grafica
'de comportamiento por hora
Graph2.GraphType = 3
'Llama al procedimiento PROC_DIBUJA_GRAF_COMPORTAMIENTO_DIARIO
PROC_DIBUJA_GRAF_COMPORTAMIENTO_HORA
'Cambia el apuntador del raton a default
Screen.MousePointer = 0
'Inhabilita el boton de Ejecutar
BC_EJECUTAR.Enabled = False
'Haz visible la malla, las graficas y el panel del control de graficas
GR_ESTADISTICAS.Visible = True
Graph1.Visible = True
Graph2.Visible = True
FRM_COMPORTAMIENTO(1).Visible = True
End Sub

Sub BC_INICIAR_Click ()
Dim I As Integer 'Contador
Dim J As Integer 'Contador
'Habilita el boton de Ejecutar
BC_EJECUTAR.Enabled = True
'Desaparece la malla, las graficas y el panel del control de graficas
GR_ESTADISTICAS.Visible = False
Graph1.Visible = False
Graph2.Visible = False
FRM_COMPORTAMIENTO(1).Visible = False
'Inicializa las matrices de totales de columna-hora
For I = HORA_INICIAL To HORA_FINAL
    demanda_tot(I, 1) = I
    demanda_tot(I, 2) = 0
    oferta_tot(I, 1) = I
    oferta_tot(I, 2) = 0
Next I
'Limpia la tabla de estadisticas
For I = 0 To NUM_REN - 1
  GR_ESTADISTICAS.Row = I
  For J = 0 To NUM_COL - 1
    GR_ESTADISTICAS.Col = J
    GR_ESTADISTICAS.Text = ""
  Next J
Next I
End Sub

Sub BC_LINEAS_G1_Click (Value As Integer)
If Graph1.GraphType = 3 Or Graph1.GraphType = 4 Then
  'Asigna al tipo de grafica lineas
  Graph1.GraphType = 6
  'Despliega la Grafica de Comportamiento Diario
  Graph1.DrawMode = 2
Else
  'Asigna al tipo de grafica lineas
  Graph1.GraphType = 6
  'Llama al procedimiento PROC_DIBUJA_GRAF_COMPORTAMIENTO_DIARIO
  PROC_DIBUJA_GRAF_COMPORTAMIENTO_DIARIO
End If
End Sub

Sub BC_INICIAR_Click ()
Dim I As Integer 'Contador
Dim J As Integer 'Contador
'Habilita el boton de Ejecutar
```

```
BC_EJECUTAR.Enabled = True
'Desaparece la malla, las graficas y el panel del control de graficas
GR_ESTADISTICAS.Visible = False
Graph1.Visible = False
Graph2.Visible = False
FRM_COMPORTAMIENTO(1).Visible = False
'Inicializa las matrices de totales de columna-hora
For i = HORA_INICIAL To HORA_FINAL
    demanda_tot(i, 1) = i
    demanda_tot(i, 2) = 0
    oferta_tot(i, 1) = i
    oferta_tot(i, 2) = 0
Next i
'Limpia la tabla de estadisticas
For i = 0 To NUM_REN - 1
  GR_ESTADISTICAS.Row = i
  For j = 0 To NUM_COL - 1
    GR_ESTADISTICAS.Col = j
    GR_ESTADISTICAS.Text = ""
  Next j
Next i
End Sub

Sub BC_SECTORES_3D_G1_Click (Value As Integer)
'Asigna al tipo de grafica Sectores 2D
Graph1.GraphType = 2
'Despliega la Grafica de Comportamiento Semanal
PROC_DIBUJA_GRAF_COMPORTAMIENTO_SEMANAL
End Sub
```