UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO E IMPLEMENTACION DEL SISTEMA DE INSCRIPCIONES DE LA FACULTAD DE DERECHO BAJO UN AMBIENTE CLIENTE/SERVIDOR.

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

BARRETO DURAN JOSE

DIRECTOR DE TESIS: ING. ORLANDO ZALDIVAR ZAMORATEGUI

MEXICO, D. F. SEPTIEMBRE 1997.

Dedicatorias

A Dios:

Por permitirme existir y darme la oportunidad de enfrentar grandes retos que me hacen ser cada día mejor.

A mi Mamá, la Profesora Ma. Inés Durán Anguiano:

Por haberme dado la vida y ser una fuente interminable de comprensión, ternura y amor. Gracias Mamá por todo ese apoyo moral y económico, sin el cual hubiera sido imposible concluir mi carrera.

A mi Papá, el Lic. José Barreto Macías:

Por enseñarme que la educación es la mejor forma de afrontar la vida, e invitarme a resolver los problemas con serenidad e inteligencia. Sin duda Papá, tu apoyo moral y económico me será de enorme utilidad toda mi vida.

A mi hermana Mercedes:

Cuyos momentos de alegría me dan fuerza para seguir adelante. Deseando que este trabajo sea un estímulo para que tu vida profesional y personal esté llena de satisfacciones.

A mi novia, la Lic. Rocío Lechuga Romo:

Gracias mi amor por tu apoyo incondicional que llena de energía todas las facetas de mi existencia. Los valores que me has inculcado me convierten en un ser humano sensible y con voluntad de ayudar. Deseo fervientemente que podamos compartir nuestra vida envueltos en esa magia indescriptible llamada amor.

A mis amigos:

Gracias a todos por haber compartido conmigo grandes momentos de alegria y tristezas, y haberme impulsado siempre a no detener un paso firme y constante para alcanzar mis metas. A ustedes les debo buena parte de mi forma de ser.

A mi Alma Mater, la Universidad Nacional Autónoma de México:

En cuyas aulas recibi el conocimiento y la formación que me harán desempeñar mi profesión con alto sentido de responsabilidad, sabiendo que llevaré muy en alto y con enorme orgullo la satisfacción de pertenecer a la máxima casa de estudios del pais, mi querida Universidad Nacional.

A la Facultad de Ingeniería:

Cuyos conocimientos impartidos por brillantes profesores, me han sido y serán de enorme utilidad profesionalmente, y constituyen una sólida base para mantener una actualización constante en un entorno cambiante y cada vez más exigente. Sin duda, haber pasado por sus aulas ha cambiado mi personalidad.

Al Centro de Cómputo de la Facultad de Derecho:

Cuyo personal me abrió las puertas y me apoyó decididamente para poder realizar con éxito este trabajo.

Al Ing. Orlando Zaldivar Zamorategui:

Cuyo ejemplo de excelencia y atinada dirección, me permitieron ser más exigente conmigo mismo y tener la convicción de alcanzar siempre mejores resultados para concluir exitosamente mi tesis de licenciatura.

Contenido

Contenido

Introducción

Introducción

La historia de la humanidad ha presenciado que la ingeniería es una de las actividades centrales que más ha contribuido al desarrollo del ser humano. Mediante el uso de la ciencia y la tecnología, se crean soluciones que al quedar inmersas en la realidad de las personas comunes, elevan su nivel de vida. Tal es el caso de las carreteras, de los procesos de producción industrial, de las redes de telecomunicación, de las computadoras y de un sin número de obras que los ingenieros generan.

La ingeniería en computación tiene un pasado relativamente corto e interesante y seguramente su futuro será de vital importancia para el quehacer científico, económico, cultural y social de cualquier nación del orbe. Es por ello que, en la medida en que seamos capaces de responder a las demandas de la sociedad en materia de cómputo, estaremos cumpliendo con la misión de nuestra Facultad y por supuesto con la de nuestra Universidad Nacional.

En el presente trabajo, el objetivo es analizar, diseñar, construir e implantar el sistema de inscripciones de la Facultad de Derecho de la UNAM bajo un ambiente cliente/servidor. Actualmente, este proceso se apoya en un sistema hecho en Paradox que ha quedado desactualizado operativa y tecnológicamente, representa demasiados problemas de diversa índole, como quedar fuera de operación constantemente durante horas críticas (caídas del sistema), alto tráfico en la red de cómputo, pérdida de información, complejidad en el mantenimiento y

adecuación de cambios en los reglamentos de la UNAM, etc., implicando todo esto un alto esfuerzo y costo de mantenimiento.

En el primer capítulo se presentan los fundamentos teóricos más importantes sobre los cuales se sustenta la tecnología cliente/servidor. Se detallan los esquemas de administración de los sistemas operativos y se particulariza para el caso de UNIX, la plataforma con la que cuenta la Facultad. Se plantean los conceptos principales de redes de computadoras, como las arquitecturas de comunicación, el modelo OSI y el protocolo TCP/IP. Posteriormente, se plasma la teoría principal acerca de las bases de datos: la arquitectura de un sistema de base de datos, los modelos de datos jerárquico, de red y relacional, el proceso de diseño y los sistemas manejadores de bases de datos. Respecto a la ingeniería de software, se presenta el ciclo de vida del software, el proceso de análisis y diseño estructurado y el enfoque orientado a objetos. De lleno en cliente/servidor, se consideran las arquitecturas de cómputo, los componentes del modelo cliente/servidor: el cliente, el servidor y la red de comunicación. Se profundiza además respecto a las categorías de aplicaciones, los tipos de procesamiento y los mitos alrededor del modelo. Para concluir el capítulo uno, se estudia la metodología de desarrollo de aplicaciones bajo cliente/servidor.

En el segundo capítulo se hace el planteamiento del problema a resolver, considerando la manera en que está estructurada la institución que demanda la solución y la recopilación de expectativas de los diferentes clientes del proceso.

En el capítulo tres, referente a alternativas de solución, se aprecia la parte de la ingeniería encargada de imaginar todas las posibles soluciones que satisfacen de un modo u otro las necesidades de los clientes. Para cada una de ellas se tiene un análisis de ventajas y desventajas, y finalmente el esquema de elección que nos permite llegar a la mejor alternativa.

De lleno en el proceso de solución, en el capítulo cuatro se detalla el análisis y diseño de la alternativa elegida con anterioridad. Como el enfoque que se sigue es estructurado, en la parte de análisis se realiza el modelo ambiental y el modelo de comportamiento.

Respecto al diseño, se construye dividiéndolo en su parte externa e interna. En esta fase se alberga el diseño de la base de datos, misma que constituye la columna vertebral de toda la herramienta.

Una vez concluidas todas las especificaciones necesarias, en el capítulo cinco denominado implantación, se muestra el proceso de desarrollo del sistema a partir de los diseños elaborados. Se tiene la creación de la base de datos en el manejador Sybase y la elaboración de la parte cliente del sistema, mediante el uso del lenguaje PowerBuilder.

Otro importante aspecto es vivir a la ingeniería apoyando a la misma ingeniería, tal es el caso del CASE ErWin (Computer Aided Software Engineering - Ingeniería de software asistida por computadora), el cual desempeña un papel fundamental desde el diseño y hasta la implementación de la base de datos.

A pesar del enfoque de calidad que considera controles a lo largo de todo el proceso de elaboración del sistema, existe una etapa específica de pruebas durante el desarrollo y otra al final. Los resultados se encuentran en este mismo capítulo. Finalmente se describen los requisitos operacionales del sistema creado y la puesta a punto del mismo.

Capítulo 1

Fundamentos Teóricos

1.1

Sistemas operativos

1.1.1.-Generalidades.

Los programas de computadora pueden clasificarse en dos tipos :

a) Programas de sistema: Controlan la operación de la computadora, desde el CPU, la memoria principal, la memoria secundaria hasta los dispositivos de entrada/salida.

b) Programas de aplicación: Son los que resuelven problemas directos de los usuarios, v.gr. una hoja de cálculo, un procesador de textos o un programa de graficación.

El programa de sistema más importante lo constituye el Sistema Operativo (S.O.). Este controla y administra todos los recursos de la máquina y provee una plataforma para el desarrollo de los programas de aplicación. De esta manera, el S.O. se asegura que los usuarios utilicen eficazmente los recursos y servicios que proporciona una computadora. Un sistema operativo es una interfaz entre los usuarios y el hardware de la máquina, y desde esa perspectiva, realiza tareas para ambas partes.

La cantidad de componentes del hardware y la enorme diversidad de estatus del mismo, llevó a la tecnología a desarrollar una capa por arriba de éste que permitiera una interacción más amigable con las computadoras: El sistema operativo.

Dentro de las capas de un sistema de cómputo tenemos:

I.- El hardware, cuyos componentes principales son :

a) Los dispositivos físicos, conformados por procesadores, la unidad lógico-aritmética, registros, fuentes de poder, etcétera.

b) La microprogramación: Software primitivo que controla de forma directa los dispositivos y que reside en la memoria de sólo lectura (ROM: read only memory), en esta parte se llevan a cabo pequeños pasos para ejecutar las instrucciones del lenguaje máquina.

c) El lenguaje máquina: Consta de una serie de instrucciones que hacen operaciones aritméticas, desplazan datos a través de la máquina y comparan valores.

II.- Los programas de sistema.

Esta parte comprende los editores, compiladores e intérpretes de comandos , y por otro lado, el sistema operativo. Una diferencia entre ambos, es que el software que constituye al sistema operativo no puede ser modificado o sustituido por uno construido por el usuario mismo, a diferencia de un compilador o un editor. La interfaz principal entre un usuario y el sistema operativo se denomina intérprete de comandos, y para el caso de UNIX se le conoce como shell.

III.- Programas de aplicación.

Son escritos por los usuarios para resolver tareas específicas, v.gr. cálculo del método de aproximaciones sucesivas, determinación de raíces complejas de un polinomio, procesadores de

textos, controladores de impresión de alta velocidad, etcétera.

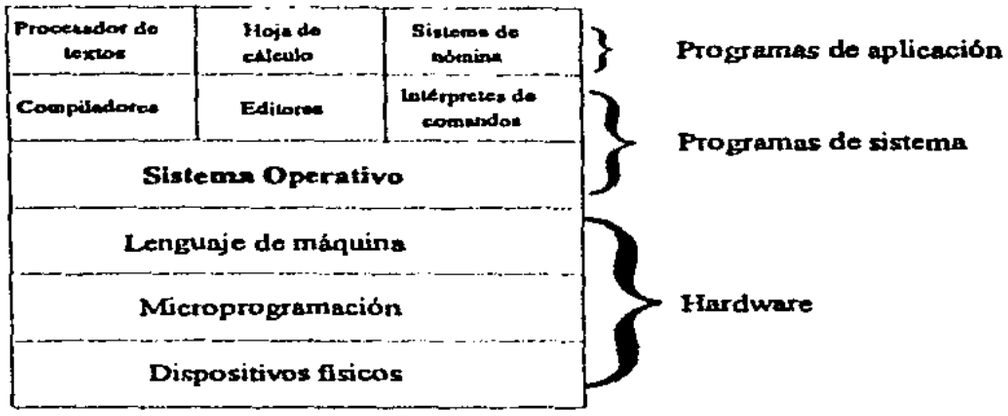


Fig. 1.1.1 Capas de un sistema de cómputo.

Los programas de usuario se comunican con el sistema operativo y le solicitan servicio mediante las "llamadas al sistema". Estas manipulan tanto a los procesos como a los archivos. El sistema operativo es el código que lleva a cabo las llamadas al sistema.

Los sistemas operativos realizan cuatro funciones fundamentales:

1.- Administración de procesos.
2.- Administración de la memoria.
3.- Administración de archivos.
4.- Administración de los dispositivos de entrada y salida.

1.1.2 Administración de procesos.

El concepto de proceso se introdujo en los sesentas por los diseñadores del sistema Multics, y constituye el concepto principal de cualquier sistema operativo. En las palabras de Deitel " Un proceso se puede definir como un programa que se está ejecutando." [DEI87]

En un medio ambiente de cómputo con varios usuarios, cada uno de ellos puede ejecutar procesos distintos simultáneamente, una computadora sólo ejecuta un programa a la vez, pero el tiempo en que posee el CPU dicho programa es del orden de los milisegundos, lo que conlleva a ejecutar varios procesos en un segundo. Lo anterior es lo que nos da una sensación de paralelismo. La actividad anterior, que alterna de proceso en proceso se le denomina multiprogramación.

Estados de un proceso.

Los tres estados en que se puede encontrar un proceso son:

a) Ejecución: El proceso posee el CPU.

b) Listo: Preparado para ejecutarse, en cuanto exista CPU disponible.

c) Bloqueado: El proceso está esperando que ocurra algo para continuar.

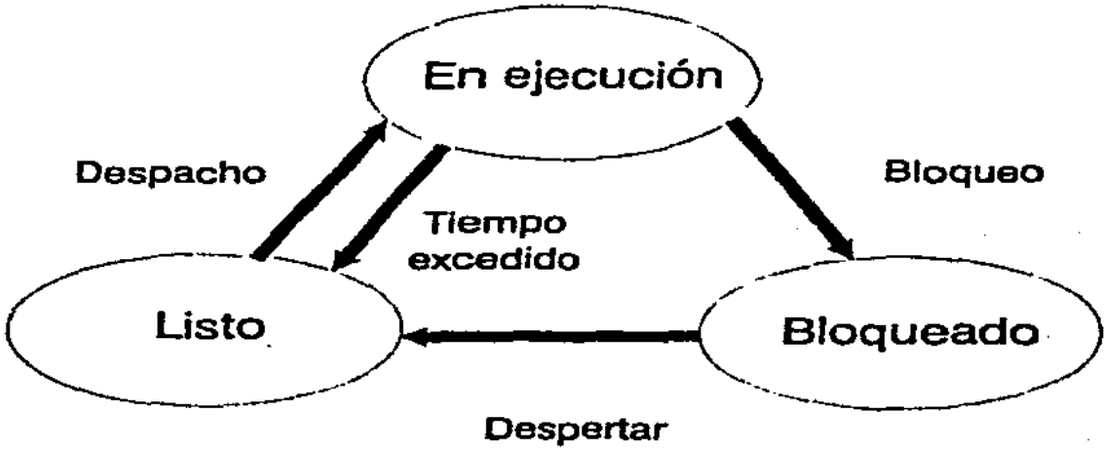


Fig. 1.1.2.1 Estados de un proceso

Se pueden dar varias transacciones entre los estados de los procesos:

Listo-Ejecución (Despacho).

Cuando se libera el CPU, entra el primer proceso de la fila de listos, es decir, cambia del estado listo al estado de ejecución.

Ejecución-Listo (Tiempo excedido).

El hardware de la computadora asigna a cada proceso un tiempo denominado "cuanto", que al ser rebasado por un proceso en ejecución, hace que el sistema operativo tome el control y el proceso cambie al estado de listo.

Ejecución-Bloqueado (Bloqueo) y Bloqueado-Listo (Despertar).

Cuando un proceso se encuentra en ejecución y se tiene que dar una operación de entrada/salida, se pasa del estado de ejecución al de bloqueo. Cuando la operación de entrada/salida concluye, se transita del estatus de bloqueo al de listo.

Los datos claves de un proceso son: la identificación, el estado actual, la prioridad, la memoria, los recursos y los valores de registro entre otros, se almacenan en una estructura de datos llamada bloque de control de proceso (process control block, PCB).

Para lograr un uso más eficiente de los recursos del sistema, el sistema operativo realiza varias operaciones con los procesos, como: crear, destruir, suspender, reanudar, cambiar la prioridad, bloquear, despertar y despachar a los procesos. Cuando un proceso sufre una interrupción, es necesario guardar el estado en el que se encuentra al momento de ser interrumpido, para que una vez concluida la interrupción, pueda reanudar con las mismas condiciones.

En algunos casos los procesos pueden compartir áreas de almacenamiento, lo que se conoce como condiciones de competencia. Para evitar lo anterior, existe una sección de código denominada crítica, que impide a otros procesos trabajar en esa zona. "Los procesos se pueden comunicar entre sí mediante las primitivas de comunicación entre procesos, las cuales se utilizan para garantizar que dos procesos no se encuentren jamás al mismo tiempo dentro de sus regiones críticas." [DEI87]

Entre las primitivas de comunicación entre procesos tenemos: los semáforos, los monitores, los contadores de eventos y la transferencia de mensajes.

Otra peculiaridad de los procesos es que requieren planificarse, es decir, se debe determinar mediante ciertos algoritmos qué proceso va a ejecutarse a continuación. Los algoritmos de planificación más populares son: las colas de varios niveles, el "round robin", la planificación por prioridades y primero el trabajo más corto.

En el almacenamiento principal reside una parte vital del sistema operativo, el núcleo. Entre sus funciones están: la manipulación de procesos, las interrupciones, el bloque de control de procesos, la comunicación interprocesos, la sincronización de procesos, el soporte de entrada/salida, la asignación y desasignación del almacenamiento, entre otras. La tendencia general es migrar las partes del núcleo al microcódigo, logrando mayor seguridad y rapidez.

1.1.3 Administración de la memoria.

Memoria real.

Las función del administrador de la memoria consiste en asignar espacio en memoria a los procesos que la requieran y liberarla cuando la desocupen, para ello se lleva un registro de las partes de la memoria que se estén utilizando y se debe coordinar el intercambio entre la memoria principal y el disco.

Entre las estrategias de administración de memoria tenemos:

a) **Búsqueda.** Traen a memoria el siguiente fragmento de programas o de datos. Existen dos clasificaciones:

1.- Por demanda. Los fragmentos se traen a petición explícita.
2.- Anticipada. El sistema se anticipa a los requerimientos del programa, y trae datos o programas antes de que sean necesarios.

b) De colocación. Determinan el lugar de la memoria donde será colocado un programa. Se dividen en tres tipos :

1.- Primer ajuste. Coloca el programa en el primer hueco encontrado.
2.- Mejor ajuste. Posiciona el programa en el hueco más pequeño con capacidad de aceptarlo.
3.- Peor ajuste. Coloca el programa en el hueco más grande disponible.

c) De reposición. Su objetivo es decidir cuál de los programas que ocupan el almacenamiento principal será retirado para hacer espacio al no haber huecos disponibles.

Organizaciones para sistemas de almacenamiento real.

a) Sistemas monousuario. Sólo se permite un usuario a la vez.

b) Multiprogramación de partición fija con traducción y carga absolutas. Su uso inició al tener varios usuarios compartiendo una máquina. La organización consiste en que los programas se ejecutan en particiones específicas, y en caso de estar ocupadas, el programa debe esperar, aun cuando hubiese otras libres.

c) Multiprogramación de partición fija con traducción y carga relocalizable. Similar a la anterior, con la ventaja de que puede cargarse un programa en cualquier partición disponible y que lo pueda aceptar.

d) Multiprogramación de partición variable. Tiene una ventaja respecto a las organizaciones anteriores al asignar la cantidad de espacio exacta que un programa requiere. La desventaja es que al liberarse quedan muchos huecos por toda la memoria. Para lograr juntar todos los huecos se ejecuta un proceso de compactación, cuyo resultado es tener adyacentes a todos los programas y un agujero grande de memoria disponible.

e) Sistemas de intercambio de almacenamiento. Se ejecuta un solo programa hasta que no pueda continuar, y es intercambiado por el siguiente programa.

Memoria virtual.

En este esquema, se tiene un número mayor de direcciones que las que existen en la memoria primaria. Para hacer posible la memoria virtual existen dos métodos: paginación y segmentación.
" La clave del concepto de almacenamiento virtual está en la disociación de las direcciones virtuales referenciadas en un proceso en ejecución de las direcciones disponibles en el almacenamiento primario." [TAN92]

En este tipo de almacenamiento, se requiere de mapas de direcciones virtuales a las reales, y una premisa fundamental es minimizar por medio de la transformación de bloques, la cantidad de información transformada que debe mantenerse en el almacenamiento primario. La transformación es común que se realice en memoria cache.

A los bloques de tamaño fijo se les conoce como páginas y los variables se denominan segmentos.

Paginación.

Las páginas virtuales son del mismo tamaño que las páginas reales, y por lo tanto, se pueden colocar en cualquier página disponible. En la paginación es necesaria una transformación de un número de página en un marco de página, cuando la transformación es directa se requiere que la tabla completa del mapa de páginas resida en la memoria primaria.

Segmentación.

Los bloques son variables y se ajustan para almacenar unidades lógicas mínimas como estructuras de datos. Para ejecutar un proceso, se necesita que su segmento actual esté en memoria primaria, pero no se requiere que todos los segmentos del proceso residan en el almacenamiento primario.

1.1.4 Administración de archivos.

La información que procesa una computadora, que en ocasiones puede ser muy grande, debe conservarse aun cuando el proceso que la generó o transformó haya concluido, y en algunos casos es necesario que varios procesos accesen la misma información simultáneamente. Tanto la memoria real como la virtual son capaces de almacenar datos, pero se pierden tan pronto como el proceso que manipula esa información haya terminado. Todo esto conlleva a requerir un medio de conservar nuestra información en medios externos: los archivos.

Un archivo se puede definir como "Una colección de datos con nombre que suele residir en un dispositivo de almacenamiento secundario como un disco o una cinta." [TAN92]

Los archivos están formados por registros, los cuales pueden ser afectados por operaciones como lectura, escritura, actualización, inserción y borrado. A su vez, a todo el archivo se le puede manipular por medio de acciones como abrir, cerrar, crear, destruir, copiar, renombrar y desplegar.

Organización de archivos.

Existen cuatro esquemas de organización de archivos que son formas de acomodarlos dentro del almacenamiento secundario:

a) Secuencial. La colocación de registros se da en orden físico, esto es, el registro siguiente es el que está a continuación físicamente. Esta organización se usa regularmente en el almacenamiento en cintas magnéticas.

b) Secuencial indexado. Los registros se almacenan en cierta secuencia lógica que dependa de la clave de cada registro. Existe un índice que contiene la dirección física de los registros principales. Los accesos pueden ser secuenciales, por orden de clave, o directos buscando a lo largo del índice creado. Lo más frecuente es tener esta organización en disco.

c) Directo. Los archivos tienen acceso al azar, dependiendo de las direcciones físicas del dispositivo de almacenamiento de acceso directo. El orden es fijado por el propio usuario, según convenga a la aplicación.

d) De partición. En este caso se tiene archivos de subarchivos secuenciales denominados miembros, cuya dirección de inicio se almacena en el directorio del archivo.

Métodos de acceso.

El sistema operativo provee funciones de acceso a archivos que se denominan métodos de acceso, entre los cuales figuran el básico y por colas. El primero es usado cuando se desconoce la secuencia de proceso de los registros, y en el caso de las colas, se tiene una idea del orden en que se irán procesando los registros.

En los sistemas de administración de archivos se utiliza una matriz de control de acceso, que indica las propiedades de acceso de los usuarios o grupos de usuarios respecto a los archivos.

Asignación de archivos.

Para asignar los archivos al almacenamiento secundario existen varias maneras:

a) Contigua. Un archivo se asigna a zonas contiguas del almacenamiento secundario, donde el tamaño de la zona lo especifica el usuario. Lo anterior conlleva a un acceso ágil, pero se presentan problemas de fragmentación.

b) No contigua. Es posible tener un archivo en diversas áreas del disco, lo que ocasiona que el numero de búsquedas se incremente. Las dos variantes dentro de la categoría "no contigua" son: la asignación encadenada orientada hacia el sector y la asignación por bloques.

Para el primer caso, el disco y los archivos están divididos en sectores, estos últimos pueden almacenarse en diversas secciones dispersas del disco. La asignación por bloques es mas eficiente, consiste en asignar bloques de sectores contiguos; cuando se necesita un nuevo bloque se procura que esté lo más cercano al bloque de archivo existente.

Todo sistema de archivos debe tener funciones de respaldo y recuperación, que garanticen la permanencia de la información cuando ésta es afectada por diversos factores como: fallas de energía eléctrica, siniestros naturales o a consecuencia misma de errores en los procesos. Un método sencillo de respaldo consiste en hacer copias de los archivos y situarlas en lugares diferentes a la fuente.

1.1.5 Administración de entrada/salida.

Los dispositivos de entrada/salida se dividen en dos categorías:

Dispositivos de bloque. Se tienen bloques con una dirección y tamaño específico, v.gr. 1 Kbyte, en los cuales se puede leer o escribir en el momento necesario. Un ejemplo de estos dispositivos es un disco o una cinta.

Dispositivos de carácter: En éstos no existen los bloques, sólo la recepción y envío de flujos de caracteres. Las impresoras, las terminales o el ratón, son dispositivos de caracter.

Las unidades de E/S poseen un componente electrónico denominado controlador del dispositivo y una parte mecánica constituida por el dispositivo mismo. Para el caso de las computadoras personales o las minicomputadoras, el controlador del dispositivo es una tarjeta con circuitos montable en la computadora.

La gran ventaja de los controladores de dispositivos, es que ellos mismos manipulan a los dispositivos físicos y el sistema operativo se encarga de interactuar con los primeros, definiéndoles algunos parámetros de funcionamiento. Podemos entonces considerar a los controladores como intermediarios entre el sistema operativo y los dispositivos de E/S.

Acceso directo a memoria.

El acceso directo a memoria es un mecanismo diseñado para minimizar el tiempo de lectura de los dispositivos de E/S y por lo tanto liberar de trabajo al CPU. Esto se logra de la siguiente manera, el CPU proporciona la dirección del bloque en disco, la dirección en memoria donde debe colocarse el bloque y la cantidad de bytes a transferir. Una vez que el controlador tiene la información del disco en el buffer, procede a cargarla en memoria en la dirección que se especificó, hasta vaciar el contenido del buffer.

El software de E/S se divide en cuatro capas:

1.- Manejadores de interrupciones. La premisa esencial es que un proceso que ha iniciado una operación de E/S quede bloqueado hasta concluir la E/S y ocurra la interrupción.

Capa	Funciones
Procesos del usuario	Hace llamadas y da formato a la E/S.
Software independiente del dispositivo	Nombre, protección, bloqueo, uso de buffers y asignación.
Manejadores de dispositivos	Conforma los registros del dispositivo y verifica el estado.
Manejadores de interrupciones	Despierta al manejador al término de la E/S.
Hardware	Lleva a cabo la operación de E/S.

Fig. 1.1.5.1 Capas de entrada/salida y funciones de cada una.

2.- Manejadores de dispositivos. Los controladores de dispositivos tienen registros de dispositivos que se usan para ingresar los comandos. Los manejadores proporcionan los comandos y se aseguran de su correcta ejecución.

3.- Software de sistema operativo independiente de los dispositivos. Realiza las actividades de E/S comunes a todos los dispositivos, tales como: informe de errores, asignación y liberación de los dispositivos de uso exclusivo, uso de buffers y protección de dispositivos, entre otras.

4.- Software de E/S a nivel usuario. Está constituido por bibliotecas que se ligan con los programas del usuario para atender peticiones de E/S.

Como ejemplos de dispositivos de E/S tenemos a los discos, las terminales, las impresoras y los relojes.

1.1.6 El sistema operativo UNIX.

1.1.6.1 Historia.

En los años cuarentas y cincuentas, las computadoras eran de uso personal. El usuario utilizaba la máquina de forma exclusiva por el tiempo que la había solicitado. En los sesentas se incorpora el concepto de procesamiento por lotes y tarjetas perforadas, donde los resultados de los programas introducidos por medio de estas tarjetas se obtenían varias horas después. Posteriormente, se introdujo el concepto de tiempo compartido, por medio del sistema CTSS del M.I.T. Entre 1965 y 1970, la comunidad científica de este instituto unió fuerzas con personal de Bell Laboratories y General Electric, para crear el " Multiplexed Information and Computing Service" (Servicio de Información y Cómputo con Multiplexión) más conocido como MULTICS.

Desafortunadamente el proyecto MULTICS fracasó debido principalmente al atraso en la aparición del compilador del PL/I, el lenguaje seleccionado para la creación del MULTICS. Ken Thompson, uno de los colaboradores del proyecto, decidió crear un MULTICS nuevo, escrito en lenguaje ensamblador en una computadora PDP-7. Brian Kernighan denominó a este nuevo esfuerzo, con cierto sentido de burla, "Uniplexed Information and Computing Service - UNICS " (Sistema de Información y Cómputo con Uniplexión). Finalmente el nombre se cambió a UNIX.

Dennis Ritchie se unió al equipo de Thompson y realizaron una evolución importante respecto al UNIX, tanto a nivel hardware como software, lo trasladaron de la PDP-7 a la PDP-11/70 y se reescribió en el lenguaje de alto nivel B (forma simplificada del BCPL y del CPL). Este avance, permitió no tener que escribir el UNIX para cada tipo de máquina. La carencia de estructuras del B motivaron a Ritchie a diseñar el lenguaje de programación C, el cual usaron para volver a escribir UNIX en C.

Como en la década de los setentas las leyes no permitían a AT&T comercializar con el software, el UNIX fue otorgado a bajo costo a varias universidades que empezaban a enterarse de su existencia. Este hecho condicionó que se volviera popular entre los estudiantes, quienes al incorporarse a la industria lo siguieron difundiendo.

Hubo otro acontecimiento importante en la historia del UNIX, la creación del compilador portable de C, realizado por Steve Johnson de los Laboratorios Bell. Este producto superó al compilador anterior al lograr código objeto para una cantidad considerable de máquinas, no sólo para la PDP-11; este hecho lo hizo más portable.

Uno de los centros educativos que adquirió el UNIX fue la Universidad de Berkeley, quien aprovechó la distribución que incluía programas fuente. Esta universidad le añadió sustanciales mejoras como: manejo de memoria virtual y paginación, nuevo editor (vi), nuevo shell (csh) y compiladores de Pascal y Lisp, entre otras cosas. Lo anterior condujo a que muchas instituciones se basaran en el UNIX de Berkeley en vez del de AT&T.

La IEEE intentó formar un estándar que considerara la intersección de las carcaterísticas de la versión de AT&T y la de Berkeley, este esfuerzo se concentró en el estándar 1003.1. Desafortunadamente, IBM, Hewlett-Packard y DEC, crearon la OSF (Open Software Foundation) para restarle poder a AT&T con una nueva versión de UNIX que incorporara lo contenido en el estándar y características propias como la interfaz gráfica MOTIF y el manejo de ventanas X11. A final de cuentas, hoy no se tiene una versión estándar y existen varios fabricantes que agrupan sus propias características.

1.1.6.2 Características generales de UNIX.

Entre las características más importantes de UNIX podemos citar:

a) Se trata de un sistema multiusuario con capacidad de multiproceso, esto es, se permite que varios usuarios utilicen el sistema simultáneamente y además cada uno puede ejecutar varios procesos al mismo tiempo.

b) Su utilización es para usuarios sofisticados. Los comandos que se utilizan no tienen mensajes de seguridad, aunque es posible implementarlos, y sólo se usa cierta parte de los verbos para indicar una acción, v.gr. para borrar un archivo se usa rm y los nombres de archivos deseados, en lugar de

remove. En esencia, puedo afirmar que su uso es para programadores experimentados, no para principiantes, como sucede en otros sistemas operativos.

c) Su popularidad se debe entre otras cosas a su gran portabilidad, es decir, puede usarse en gran cantidad de máquinas, desde computadoras personales hasta las supercomputadoras como la CRAY.

d) Está escrito en lenguaje C, en lugar de ensamblador como la mayoría de los sistemas operativos. Esto condiciona una comprensión adecuada y una gran adaptación a las diferentes computadoras.

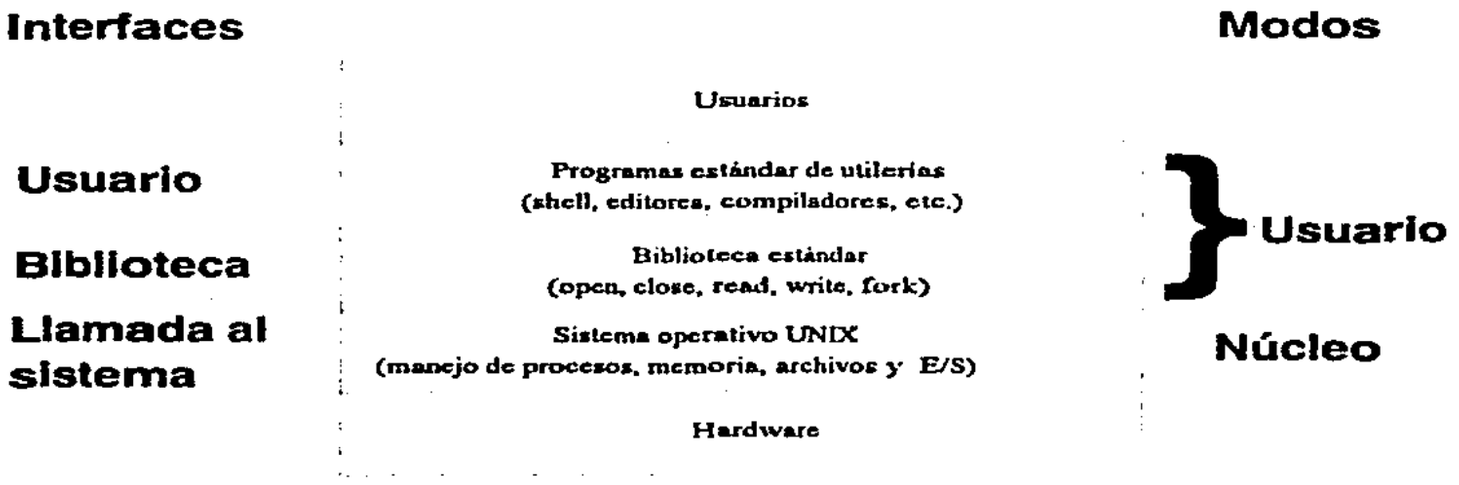


Fig. 1.1.6.2.1 Interfaces de UNIX

1.1.6.3 Administración del sistema operativo UNIX.

Procesos.

Una de las principales ventajas es que se permite la multiprogramación y por ende cada usuario puede tener varios procesos activos simultáneamente. Así mismo, cada proceso ejecuta un solo programa y tiene un contador que le indica la siguiente instrucción a ejecutarse.

En UNIX es posible crear procesos y se va generando una estructura jerárquica de los mismos del estilo padre-hijos.

Los procesos se pueden comunicar de dos maneras:

Por medio de la transferencia de mensajes. Para transferir bytes de un proceso a otro se crean medios denominados "tubos", en donde la salida de un proceso suele ser la entrada de otro.

A través de las interrupciones del software. Se envía una señal a otro proceso, el cual decide como manejarla, ya sea que la ignore, la atrape o permita que aniquile el proceso.

Memoria.

Para cada proceso en UNIX existe un espacio de direcciones dividido en tres segmentos: texto, datos y pila.

El segmento de texto se produce cuando el compilador y el ensamblador traducen el programa escrito en algún lenguaje cualquiera a código de máquina.

En el caso del segmento de datos, se almacenan variables, cadenas, arreglos y cualquier otro dato de un programa en un espacio específico. Puede haber dos tipos de datos: inicializados y no inicializados.

El segmento de pila contiene las variables del entorno y la línea de comando del shell para llamar al programa. Algo interesante dentro de UNIX es que para aprovechar al máximo la memoria, se permiten segmentos de texto compartidos.

Sistema de archivos.

El manejo de archivos en UNIX tiene una estructura jerárquica. Cuando un proceso requiere de un archivo, el primer paso es abrirlo mediante la llamada al sistema open y en caso de permitir el acceso, se regresa un entero denominado "descriptor de archivo". Los descriptores de archivo van desde el 0 para la entrada estándar, 1 para la salida estándar y 2 para el error estándar, a partir de este número por cada archivo abierto se asigna un número consecutivo.

Para especificar los nombres de archivos, se usa la ruta absoluta de acceso detallando la ubicación desde el subdirectorio raíz. Para contrarrestar esta larga ruta es posible crear un directorio de trabajo con rutas relativas de acceso.

Cuando es necesario compartir archivos que pertenecen a otros usuarios se maneja el concepto de enlace, que permite accesar datos de otros directorios de trabajo. En algunos casos, como en los mainframes, se tienen varios discos; UNIX facilita su manejo al permitir que un disco se monte en el árbol de directorios de otro disco. Para evitar que varios procesos usen el mismo archivo simultáneamente, UNIX posee la propiedad de cerradura (locking), con la cual se indica qué archivo se desea bloquear, con qué byte de inicio y el número de bytes.

Entrada/Salida en UNIX.

UNIX maneja los dispositivos de entrada/salida como archivos especiales, es decir, son una extensión del sistema de archivos. De esta manera, a la impresora o a la terminal, se les asignan nombres de archivos como lp y tty1, dentro del subdirectorio *device*. Todas las normas referentes al sistema de archivos se aplican igual para los archivos especiales. Los archivos especiales se clasifican en archivos de bloque y de carácter. Los primeros tienen la característica de poder ser direccionados y accesados de forma individual, por ejemplo los discos. Los archivos de caracter se usan para dispositivos de entrada o salida de un flujo de datos, como podrían ser una terminal, una impresora o un plotter.

1.2

Comunicaciones

1.2.1 Redes de computadoras.

El concepto de la comunicación electrónica junto con la computación en general, están revolucionando las formas de vida de empresas, instituciones educativas y paulatinamente la del ciudadano común.

El tema de esta tesis no es la excepción. La comunicación de datos entre un servidor y los clientes juega un papel fundamental. El objetivo de este capítulo es ubicar teóricamente el concepto de las redes de computadoras y el protocolo TCP/IP.

Se entiende por red de cómputo un conjunto de computadoras autónomas interconectadas por alguna vía, ya sea microondas, satélite, láser, hilo de cobre o fibra óptica; que tienen la capacidad de intercambiar información, como imágenes, voz o datos.

Para que el concepto de red exista, es necesario que las computadoras interconectadas sean autónomas, esto es, que ninguna tenga cierta influencia en el control de la otra.

El punto de diferencia entre una red de computadoras y un sistema distribuido lo constituye el hecho de que en el segundo caso la existencia de múltiples computadoras es transparente al usuario. Es labor

del sistema operativo "el seleccionar el mejor procesador, encontrar y transportar todos los archivos de entrada al procesador y poner los resultados en un lugar apropiado." [TAN91]

Como grandes ventajas de las redes podemos citar las siguientes.

1.- Compartir recursos. Una de las principales ventajas de la tecnología de redes es la capacidad para compartir la información y el equipo con que cuente la red, sin importar la ubicación física del demandante. Esto hace posible que, por ejemplo, se pueda ejecutar un programa en la Ciudad de México con datos generados en Guadalajara e imprimir los resultados en Mérida.

2.- Alta fiabilidad. Esto se consigue al permitir que la operación global de la red continúe, aun cuando haya fallado algún procesador; puesto que las otras máquinas pueden absorber el trabajo de la que se encuentre fuera de servicio.

3.- Menor costo. Las redes de computadoras tienen un costo mucho menor que los mainframes, y a pesar de que éstos tienen una velocidad de procesamiento mayor, en muchos casos resulta en una mejor relación beneficio/costo el invertir en redes de computadoras.

4.- Tamaño a la medida. Desde mi perspectiva, la mayor ventaja de las redes es que a medida que una organización va variando de tamaño, ya sea creciendo o disminuyendo, la red se puede adaptar, simplemente con aumentar o quitar computadoras. Así, la red siempre tendrá el tamaño que la organización le demande.

1.2.2 Estructura de red.

Una red de computadoras se compone de dos elementos principales, las computadoras que ejecutan programas de usuario, denominadas terminales y las subredes que interconectan a las computadoras. Las subredes tienen dos elementos principales: las líneas de transmisión encargadas de mover la información a través de la red y los elementos de conmutación, denominados IMP, que significa "Interchange Message Procesor", encargados de conectar las líneas de transmisión.

1.2.3 Arquitecturas de comunicación.

1.2.3.1 Canales punto a punto.

La red está unida por varios cables o líneas telefónicas rentadas, conectadas a un par de IMP's. Se puede dar el caso de que se necesite transferir un mensaje o paquete a través de un IMP a otro y no se cuente con una conexión física directa, en tal caso se usan IMP's intermediarios. La condición es que la línea de salida esté libre para poder reexpedir el mensaje.

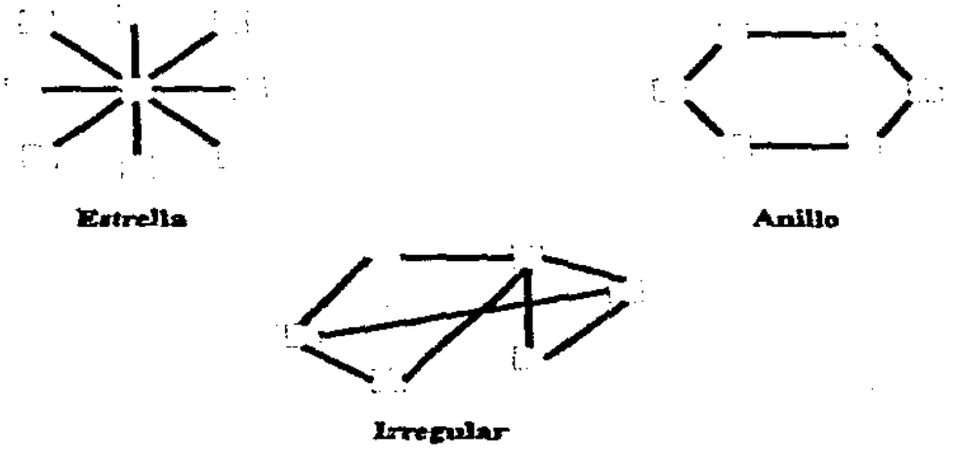


Figura 1.2.3.1.1 Topologías para redes punto a punto.

1.2.3.2 Canales multipunto.

Esta arquitectura es más común en las redes de área local, en las cuales tenemos en un circuito integrado dentro de cada terminal el IMP. En estos canales existe una sola vía de comunicación compartida por todas las máquinas de la red. Los mensajes enviados por una máquina son recibidos por todas las demás, pero se utiliza un campo de dirección dentro del paquete en el cual se especifica a quién va dirigido.

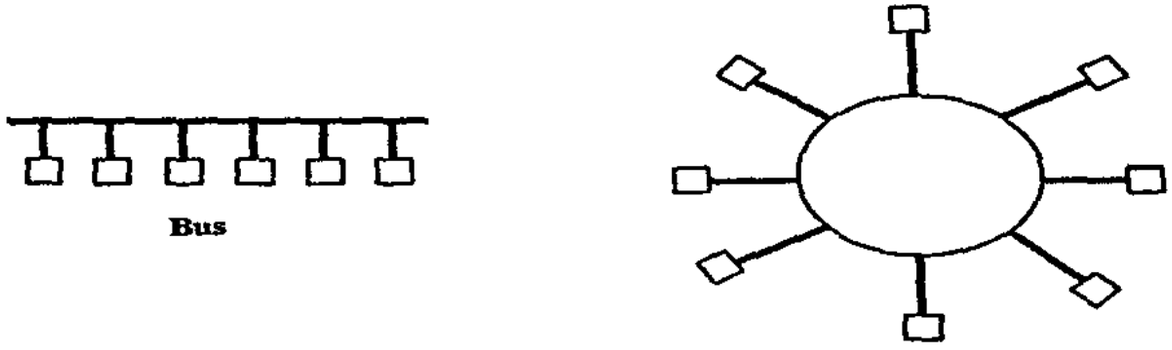


Figura 1.2.3.2.1 Topologías para redes multipunto.

De esta manera, cuando una computadora envía un mensaje, a su paso por la red, cada una de las otras verifica el campo de dirección y en caso de pertenecerle toma el paquete y de lo contrario lo ignora.

En los canales multipunto es posible enviar información a un subconjunto de máquinas o a toda la red usando un código especial, que consiste en poner en uno algunos bits del campo de dirección.

Las topologías de una red multipunto son de bus, satélite o radio y anillo. En el caso del bus, cualquier máquina puede transmitir en cualquier momento siempre y cuando el canal no esté ocupado. En ese caso, se implementa un mecanismo de arbitraje para resolver los conflictos de tráfico.

En los casos de comunicación multipunto por radiodifusión o satélite, los IMP están habilitados con una antena con la que pueden enviar o recibir información.

Un tercer tipo de comunicación es el de anillo, en donde cada bit se propaga sin esperar el resto del paquete al cual pertenece. De esta forma, algunos bits recorren toda la red antes de que todo el paquete se haya terminado de enviar.

Redes multipunto estáticas.

En este tipo de redes, a cada terminal se le asigna un intervalo de tiempo en el cual puede transmitir; con la desventaja del desperdicio de tiempo cuando no hay nada que transmitir.

Redes multipunto dinámicas.

El tiempo se asigna bajo demanda. Existen dos métodos de asignación: el centralizado y el distribuido. En el primer caso hay una unidad de arbitraje de bus que, con base en algún algoritmo, determina quién es el siguiente. Para la forma descentralizada, cada máquina decide si transmite o no; para evitar situaciones de caos se tienen algoritmos de control.

1.2.4 Medios de transmisión.

Los tres medios de transmisión más comunes son: cable coaxial, par trenzado y fibra óptica, aunque por otro lado se tienen las opciones inalámbricas.

Cable coaxial.

El cable coaxial consiste de un conductor de cobre envuelto por un material aislante. El cable coaxial contiene una segunda capa, que rodea al aislante, comúnmente un conductor en forma de malla entrelazada. Tiene además una capa protectora de material no conductor. El cable coaxial es más vulnerable a la interferencia que el par trenzado, pero es capaz de soportar 100 Mbps. El cable coaxial ha sido usado principalmente por la industria de la televisión, para transmisión de sus programas. El cable coaxial que es usado debe tener una resistencia de 50 ohms. [SAN93]

Par trenzado.

Un par trenzado consiste de dos alambres de cobre aislados y entrelazados. Una cantidad de pares trenzados pueden ser agrupados y cubiertos por una envoltura protectora, sin formar entre sí alguna interferencia. Este tipo de cableado es frecuentemente usado por las compañías de teléfonos su transmisión llega a 10 millones de bits por segundo (10Mbps). En este tipo de cableado se requiere una resistencia de 100 ohms. [SAN93]

Fibra óptica.

La fibra óptica puede transportar señales utilizando rayos de luz modulada, conocidos como pulsos. Una fibra óptica consiste de un cilindro de vidrio muy delgado llamado núcleo rodeado por una capa de vidrio y al igual que los demás con una capa protectora. La fibra óptica soporta un ancho de banda muy grande debido a que la luz tiene una frecuencia de 10^8Mhz, la transmisión alcanza 565Mbps comercialmente hablando, pero se ha comprobado la transmisión de 200,000Mbps. Las señales transmitidas por la fibra óptica no están sujetas a interferencias eléctricas, sin embargo todavía los costos son extremadamente más caros que los cables eléctricos anteriormente mencionados. [TAN91].

Medios inalámbricos.

Este medio de transmisión se puede dividir en 2 grandes ramas:

. LAN inalámbrica

. Móvil inalámbrica

La diferencia crítica radica en las facilidades de transmisión mientras que la comunicación LAN inalámbrica utiliza transmisores y receptores que se encuentran dentro de una área limitada, la móvil inalámbrica involucra a terceros, como compañías de telecomunicaciones; los usuarios que se encuentran fuera de su área necesitan utilizar radios, celulares o estaciones de satélite.

El transmisor y receptor (transceptor) inalámbrico de una LAN se localiza en algún lugar fijo, el cual recibe las señales de las estaciones de trabajo. Este medio de transmisión es recomendado cuando un equipo de trabajo requiere establecerse temporalmente en algún sitio y una vez terminado el proyecto trasladarse a otro lugar.

Una LAN inalámbrica requiere un equipo transceptor conectado a por lo menos un servidor mediante algún cableado ethernet. y el propio transceptor recibe y envía señales a las estaciones.

Se cuentan con 3 diferentes formas de transmisión inalámbrica:

. Luz infrarroja

. Radio de frecuencia simple

. Radio de espectro expandido

Microondas.

Este medio de transmisión también es inalámbrico. Como sabemos la comunicación por medio de microondas no requiere un medio físico, es posible transmitir a grandes distancias. Los enlaces por medio de microondas pueden ser los siguientes:

. Satélite.

. Enlaces de edificio a edificio en área metropolitana.

. Enlaces a través de suelos difíciles para tendido de cable.

La configuración de este medio de transmisión requiere de por lo menos dos antenas bidireccionales las cuales enfocan haces de energía electromagnética u ondas de radio de uno a otro punto, es decir la señal no se transmite de un lugar a muchos tan sólo de punto a punto. Las antenas deben ser altas para que la transmisión no encuentre obstáculo alguno. La distancia entre las antenas no deben rebasar las 30 millas. Hoy en día se cuenta con rangos de frecuencia de 2 a 5 Ghz con el ancho de banda más alto. [SHE95]

La comunicación por satélite transmite señales desde transceptores ubicados en tierra a los ubicados en el espacio. Los satélites se sincronizan con la órbita de la Tierra logrando un lugar fijo con respecto a la Tierra. Las antenas consecuentemente se orientan al satélite para poder transmitir la señal por medio de haces de microondas. En cuestión de transmisión satelital se cuentan con 3 rangos de frecuencia [SHE95]:

- Banda C. Las frecuencias de los enlaces ascendentes son del orden de los 6 Ghz y de los descendentes son del orden de los 4 Ghz.
- Banda Ku. Las frecuencias de los enlaces ascendentes son del orden de los 14 Ghz y de los descendentes son del orden de los 11 Ghz.
- Banda Ka . Las frecuencias de los enlaces ascendentes son del orden de los 30 Ghz y de los descendentes son del orden de los 20 Ghz.

1.2.5 Arquitecturas de redes.

Jerarquías de protocolos.

Las redes de computadoras se organizan en capas para reducir la complejidad de su diseño. Esas capas se construyen una sobre otra y el objetivo es ofrecer servicios a las capas superiores sin que éstas tengan conocimiento detallado de cómo se realizan las cosas en las capas predecesoras.

Las reglas y convenciones utilizadas para la comunicación entre capas se le denomina protocolo. De un modo más sencillo, un protocolo se puede entender como el conjunto de reglas que coordinan eficientemente el intercambio de mensajes entre computadoras. Para que la comunicación entre una capa de una máquina se pueda dar con la capa equivalente de otra, es preciso que la información de datos y control se vaya pasando a la capa inferior hasta que llega a la capa 1, debajo de la cuál se encuentra el medio físico por el que se realiza la comunicación real.

Entre las capas existe algo que se conoce como interface, que se encarga de definir los servicios y operaciones que la capa inferior ofrece a la superior. Así pues, una arquitectura de red se define como el conjunto de capas y protocolos.

1.2.6 Modelos de interconexión.

Netware.

Las capas del modelo Netware son:

a) Física. Por medio de la cual se lleva la transmisión de la información de un equipo físico de la red a otro.

b) Interfaz abierta de enlace de datos y especificaciones de la interfaz del controlador de red. Esta capa se divide en 3 partes la ODI y NDIS que corresponden a la capa de enlace del modelo OSI. Su función consiste en definir protocolos que envíen y reciban información entre unidades conectadas.

c) IPX (Internetwork Packet Exchange). Similar a la capa de red del modelo OSI, el cual se encarga de definir los protocolos sin conexión que encaminan los datos en forma dinámica.

d) SPX (Sequenced Packet Exchange). Comparando con el modelo OSI esta capa correspondería a la de transporte. Esta capa se encarga de que la información transmitida de un punto a otro llegue en el mismo orden en que fueron enviadas.

e) NetBIOS. Mientras los conductos nominados se encargan de ejecutar los servicios de nivel de sesión en redes netware, NetBIOS se encarga de la gestión de la sesión de comunicaciones, la denominación de nodos, la difusión de los nombres y emplazamientos de los nombres de los servidores y la transmisión de los datagramas sin conexión.

f) NCP (Netware Core Protocol). Este nivel se encarga de proporcionar acceso a los archivos, servicios de impresión, asignación de recursos, gestión, seguridad de la red, además de la comunicación entre servidores.

LAN Manager.

Fue creado por IBM en unión con Microsoft, LAN Manager es utilizado por los productos del propio Microsoft para conexión en red y es Windows NT su principal impulsor. A continuación se muestran las capas de este modelo y la relación que existe con el modelo OSI [SHL95]:

Systems Network Arquitecture (SNA).

Es un modelo creado por IBM caracterizado por ser una arquitectura jerárquica, cumple ciertas premisas fundamentales del modelo OSI como:

. Cada capa proporciona servicio a la superior y se vale de la inferior.

. Establece comunicación con el nivel correspondiente en otra red SNA.

Las capas del SNA se pueden dividir en dos grupos, el primero en el cual se consideran las 4 capas superiores: transmisión, flujo de datos, presentación y transacción proporcionando estas funciones. Por otro lado existe la parte inferior que se conforma por las capas: física, de enlace y ruteo, las cuales definen el control de ruteo y proporcionan las funciones y servicios del mismo.

Una SNA está constituida por nodos que se clasifican en controladores, que son los que supervisan el funcionamiento de los periféricos; procesadores frontales, que reducen la carga de trabajo a los CPUs principales; y los Hosts [TAN91]. Cada uno de los nodos contienen una o más unidades direccionables de red (NAU-Network Address Unit), éstas son módulos de software que permiten a los procesos utilizar la red conectándose directamente a una NAU.

Aunque OSI y SNA tienen el mismo número de capas no quiere decir que las capas correspondientes de ambos realicen la misma función, el propósito del modelo OSI es diferente al del modelo SNA. El objetivo del modelo OSI es establecer reglas y orden en el mundo de la comunicación entre redes generando un estándar. Ahora bien, el modelo SNA está diseñado para intercambiar información entre nodos de red que pertenecen a una única arquitectura. Esta única arquitectura es sobre la cual IBM construye varios de sus programas. La existencia de una única arquitectura permite a IBM conjuntar a la medida su software y hardware para lograr máxima eficiencia. De cualquier forma IBM reconoce la importancia del modelo OSI como un estándar común para la interconexión de sistemas heterogéneos.

Digital Network Arquitecture (DNA).

En 1974 Digital Equipment Corporation (DEC) empezó el desarrollo de una arquitectura de red llamada Digital Network Architecture (DNA), que es propietaria como SNA.

DNA fue creada como un producto de red punto a punto. Esto significa que ningún nodo es controlador o propietario de la red, esto mismo, hace que la red sea muy flexible, sin embargo la administración suele ser muy compleja.

La última versión de DNA es completamente compatible con las primeras 5 etapas del modelo OSI.

1.2.7 Modelo OSI.

En 1983 la Organización Internacional de Estándares (ISO) propuso un modelo para la normalización internacional de varios protocolos, al cual le denominó OSI (Open Systems Interconection, interconexión de sistemas abiertos). Al modelo lo conforman siete capas. Es importante precisar que en el modelo OSI no se indica los servicios y protocolos que se utilizan en cada capa, por lo tanto no es una arquitectura de red; sólo menciona lo que cada capa debe hacer.

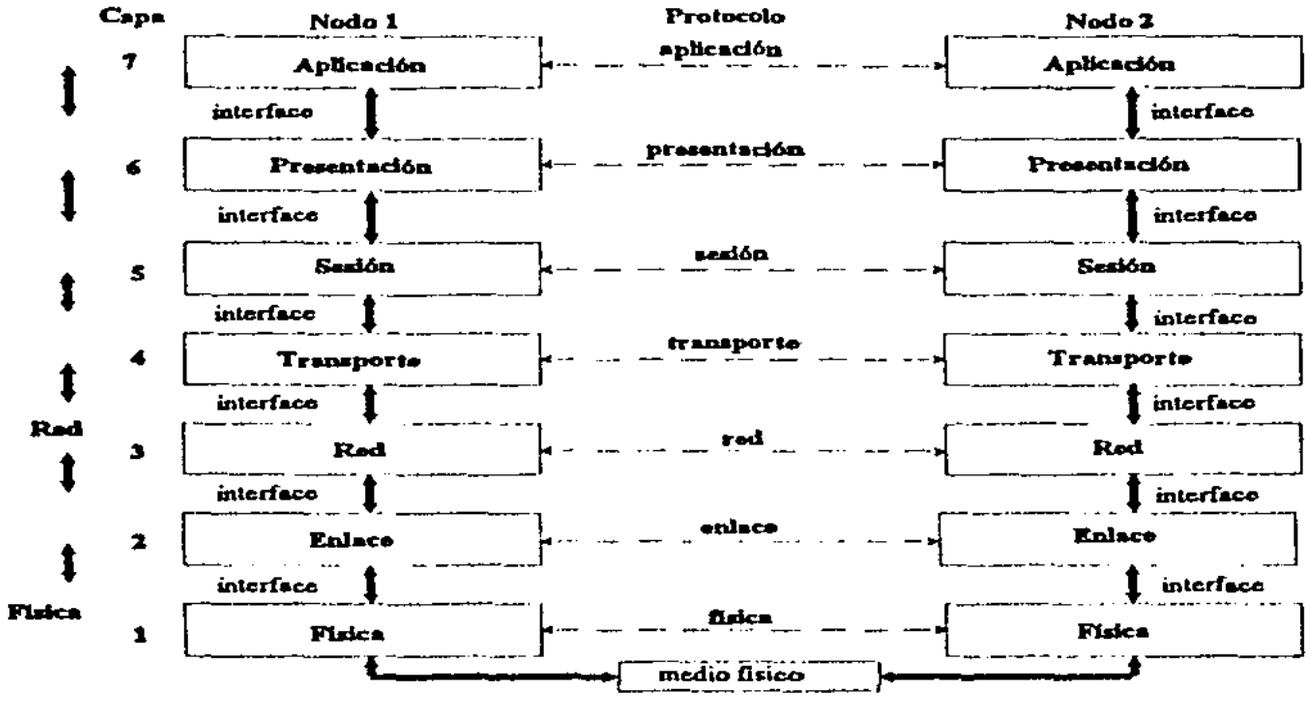


Figura 1.2.3. Arquitectura de red del modelo OSI.

Capa física.

La tarea primordial es la transmisión física de bits a través de un canal de comunicación, pero entre sus otras funciones se encuentran el establecimiento de conexión inicial y desconexión al final, la posibilidad de transmisiones bidireccionales simultáneas y las características de la interface y del medio de transmisión, como la velocidad de transmisión y la codificación de bits.

Capa de enlace.

La función base de esta capa es lograr una línea libre de errores de transmisión. La técnica utilizada consiste en partir los paquetes en tramas de datos que son transmitidos secuencialmente y analizar las tramas de confirmación enviadas por el receptor. En caso de error el mensaje debe retransmitirse.

Como ejemplos de protocolos en esta capa tenemos al CSMA/CD y al HDLC.

Capa de red.

Se encarga de operar la subred. Tiene como una de sus misiones fundamentales el direccionar los paquetes de transmisor a receptor. El protocolo IP y el X.25 residen en la capa de red.

Capa de transporte.

En la capa de transporte se toman los datos de la capa se sesión y si su tamaño lo amerita, se dividen para que sean tomados por la capa de red. Otra de las labores interesantes de esta capa es el hecho de multiplexar y demultiplexar conexiones de red cuyo tamaño depende del tráfico de información demandado. La capa de transporte puede ofrecer distintos tipos de conexiones, como el canal punto a punto sin error, en donde los mensajes se entregan en el mismo orden en que se enviaron, o como una alternativa, transportar mensajes aislados sin orden de distribución para destinos múltiples.

En esta capa entablan una conversación programas análogos situados en las máquinas origen y destino por medio de las cabeceras y los mensajes de control. TCP y UDP son ejemplos de protocolos de transporte.

Capa de sesión.

Esta capa permite que se establezcan sesiones entre los usuarios de las computadoras. Otra labor de relevancia es el control de la dirección del tráfico. En la red se maneja una bandera denominada testigo, proporcionada por la capa de sesión, la cual determina qué lado de la red puede realizar una cierta operación en un momento dado. Durante la transferencia de información entre computadoras son frecuentes las caídas del servicio. Mediante un mecanismo de inserción de puntos de verificación que provee la capa de sesión, sólo es necesaria la repetición de los datos desde el último punto de verificación.Un ejemplo de protocolo de la capa de sesión es el LU6.2.

Capa de presentación.

Su función se centra en el aseguramiento de la sintaxis y la semántica de la información transmitida. Una computadora maneja diversos tipos de datos que tienen que ser representados en algún código que finalmente se almacena en una estructura de datos. La capa de presentación ofrece el manejo de estructuras de datos abstractas y la conversión de la codificación interna de la máquina a la requerida por la red. ASN.1 y XDR son ejemplos de protocolos de la capa de presentación.

Capa de aplicación.

En esta capa se alojan servicios de uso general, como transferencia de archivos, el correo electrónico y las emulaciones de terminal. Es decir, el software que permite la interacción entre máquinas incompatibles reside en la capa de aplicación.

Cuando se requiere transferir información de una computadora a otra mediante una red, en cada capa del transmisor se le van añadiendo encabezados cuyo objetivo es garantizar algunos aspectos particulares de la transferencia descritos anteriormente. En el momento en que llega a su receptor se inicia la labor inversa y se van quitando los encabezados adicionados hasta tener nuevamente la información como en un inicio.

Durante la transmisión de datos entre computadoras es común que se presenten problemas como la interrupción de la comunicación o la presencia de datos corruptos. Existen cuatro tareas principales que debe cumplir un protocolo encaminadas al logro de una transmisión sin problemas. La primera es la detección de errores, implementada por medio de secuencias de números y mensajes de agradecimiento. La segunda es la eliminación de errores, que se logra retransmitiendo los mensajes o usando métodos de corrección como los códigos de Hamming. El direccionamiento constituye la tercer gran tarea y de lo que se trata es de que los mensajes lleguen en el mismo orden en que fueron enviados, apoyándose en campos de direccionamiento. Una situación clásica en la transferencia de información es la disparidad entre las velocidades de transmisor y receptor; para que esto no se convierta en un problema de pérdida de datos existen mecanismos de flujo de control, por medio de mensajes de agradecimiento, que ajustan la velocidad del transmisor.

Protocolos orientados a conexión y sin conexión.

En el primer caso se genera una comunicación antes de que los datos sean propiamente enviados, similar a una llamada telefónica, se tienen tres fases de operación: establecimiento de conexión, transferencia de datos y terminación de la comunicación. En la segunda categoría los datos se transmiten sin comunicación previa, análogo a un telegrama. A las unidades de información se les conoce como datagramas.

1.2.8 Arquitectura TCP/IP.

1.2.8.1 Historia.

Hacia finales de los años sesentas en Estados Unidos había una inquietud por la creación de una red que permitiera compartir los recursos de cómputo de las universidades y el gobierno. De esta manera surge la ARPANET (Advanced Research Project Agency Network), que usaba el protocolo 1822, número que correspondía a la nota de ingeniería en que fue especificada. Actualmente la ARPANET es una subred de la red mundial de computadoras INTERNET.

Inicialmente se pretendía que la ARPANET usara cuatro IMP's situados en universidades de California y Utah. EL contrato para este proyecto fue ganado por la compañía Bolt, Beranek y Newman (BBN) en 1968, que tuvo una influencia importante en el desarrollo de TCP/IP. Existía un requerimiento importante por servicios de logins remotos y transferencia de archivos que se incluyeron en el NCP (Network Control Program), el antecesor de TCP/IP. ARPANET entró en funcionamiento en 1971 con servicios de logins remotos, transferencia de archivos y correo electrónico.

En 1973 se descubrió que los protocolos usados en las capas inferiores eran inadecuados y debían sustituirse. De esta manera se inició un proyecto para la creación de un nuevo protocolo, el TCP/IP, cuyas primeras definiciones corrieron a cargo de V. Cerf y R. Kahn en 1974.

1.2.8.2 Características generales de TCP/IP.

a) Protocolos sin conexión en la capa de red.

b) Nodos como computadoras de intercambio de paquetes.

c) Ruteo dinámico.

d) Protocolos de transporte con funciones de seguridad.

e) Disponibilidad de un conjunto de programas de aplicación.

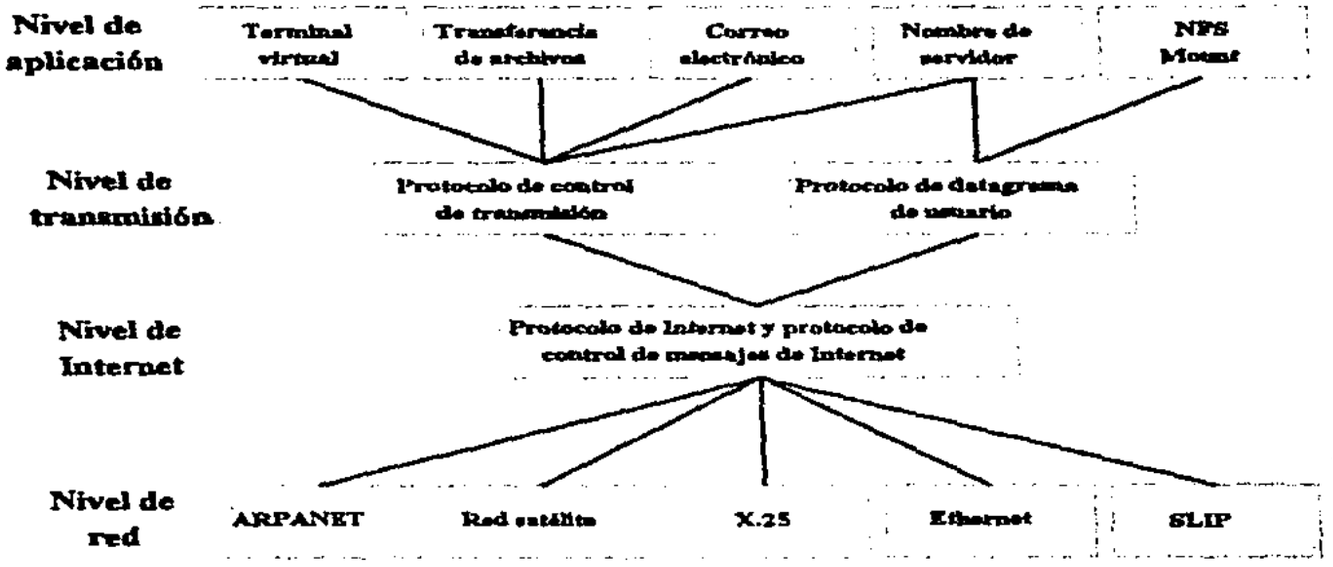


Figura 1.2.8.2.1. Modelo de la arquitectura TCP/IP.

1.2.8.3 Protocolos.

Protocolo Internet (IP).

El protocolo internet es la piedra angular de la arquitectura TCP/IP. Las principales funciones del protocolo son el direccionamiento de las computadoras y la fragmentación de paquetes. Las características esenciales de IP son:

a) Protocolo sin conexión.

b) Posibilidad de fragmentar mensajes.

c) Direccionamiento por direcciones de Internet de 32 bits.

d) Direcciones del protocolo de transporte de 8 bits.

e) Existencia de encabezados de checksum.

f) Algunos campos del protocolo son opcionales.

g) Tiempo de vida finito de los paquetes.

EI direccionamiento necesita cuatro direcciones al pasar a través de las cuatro capas del protocolo:

1.- Dirección de la subred.

2.- Dirección de Internet.

3.- Dirección del protocolo de transporte.

4.- Número de puerto.

La más importante es la de la Internet, puesto que dentro de los servicios de direccionamiento del IP se utilizan campos de dirección de esta red. Con el fin de transmitir paquetes a cualquier tipo de red, el IP debe ser capaz de adaptar el tamaño de sus datagramas a cualquier red. Para tal fin se usa la fragmentación, que consiste en que el IP de cada nodo de la red divida los paquetes que reciba para transmitirlos por subredes al siguiente nodo. De la misma manera, el IP del receptor debe ser capaz de armar los mensajes fragmentados.

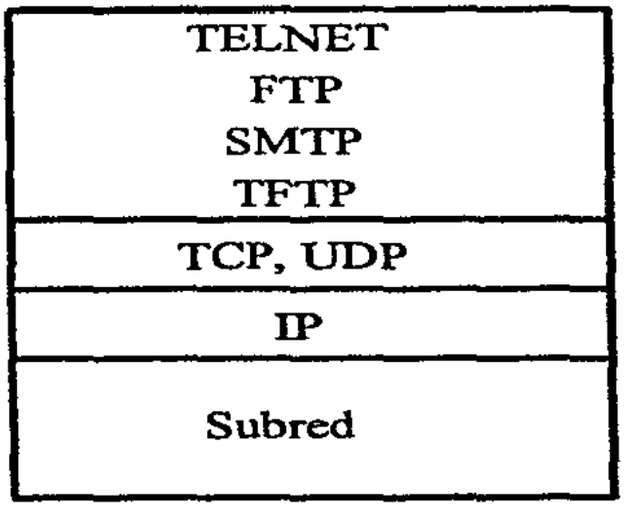


Figura 1.2.8.3.1. Capas de la arquitectura TCP/IP.

Protocolo de control de transmisión TCP (Transmission Control Protocol).

Se sitúa arriba del IP y se encuentra en la capa de transporte. Su principal función es la transportación segura de datos a través de la red. Las características que presenta TCP son:

a) Circuito virtual bidireccional duplex.

b) Los datos se transmiten divididos en grupos de datos.

c) Para lograr una transmisión segura se utilizan secuencia de números, agradecimientos, retransmisión de un segmento después de la expiración del tiempo de agradecimientos, checksums.

d) Datos urgentes y funciones de aceleramiento.

e) Direccionamiento de usuarios de transporte usando números de puerto de 16 bits.

f) Terminación suave de conexión.

El punto de diferencia entre TCP y otros protocolos es que no conserva los límites de bloques. Casi todos los encabezados en TCP son calculados en bytes, no en bloques. El tamaño de un segmento en una red se determina por la carga en la red, el tamaño de la ventana o los recursos de los receptores disponibles. Lo anterior implica que la eficiencia en la transferencia se afectaría al pretender conservar el tamaño de los bloques estáticamente, tal como fueron generados originalmente.

Protocolo de usuarios de datagramas (User Datagram Protocol).

Es un protocolo simple de transporte sin conexión, cuya dirección de transporte es 17. Utiliza direccionamiento por número de puerto y checksums de datos.

Protocolo de control de mensajes de Internet (Internet Control Message Protocol).

Es un protocolo de transporte cuya función fundamental es transportar los errores y el diagnóstico para el IP. Su dirección de transporte es 1.

Telnet.

El objetivo de este protocolo es permitir el acceso en forma de terminal virtual o login remoto a las computadoras de la red. En el caso del sistema operativo UNIX se utiliza el comando "telnet" en la parte del cliente, y se tiene un servidor denominado demonio identificado como "telnetd".

Protocolo de transferencia de archivos (File Transfer Protocol).

Los comandos del FTP se usan para envío, recepción, renombrado, borrado y unión de archivos; creación, cambio y borrado de directorios; y envío de correo electrónico. Para UNIX el comando para transferencia de archivos es ftp.

Existen dos canales en TCP para uso de ftp: el de comandos y el de datos. Se tienen dos modos de transmisión, texto y binario. Para el caso del modo texto los archivos se envían en líneas en código ASCII separadas por retorno de carro; esto tiene la ventaja de que pueden hacerse transferencias entre distintos tipos de sistemas. En modo binario un archivo se transmite como secuencia de bits, resultando en una mayor velocidad.

Protocolo simple de transferencia de correo (Simple Mail Transfer Protocol).

El SMTP es el protocolo de internet para correo electrónico. La manera de usarlo es simple, sólo es necesario especificar una introducción, la dirección de quien envía, destino del mensaje y los datos.

Protocolo trivial de transferencia de archivos (Trivial File Transfer Protocol).

Sus requisitos de uso son mínimos. Usa un protocolo de transporte sin conexión, UDP. Esto condiciona que la seguridad de la transmisión deba procurarse por medio de algoritmos diversos. Este protocolo soporta al igual que FTP dos modos de transmisión, binario y texto. Los códigos de TFTP son lectura de petición, escritura de petición, envío de datos, agradecimiento y error.

1.2.9 Protocolos de control de acceso.

Estándar IEEE 802.

En 1980 el Instituto de Ingenieros Electricistas y Eléctricos (IEEE) se dió a la tarea de normalizar un solo protocolo de acceso denominándole "802". Esta normalización trajo como consecuencia el gran desarrollo y auge de las redes locales.

Estándar IEEE 802.3, CSMA/CD.

Protocolo de Acceso: CSMA/CD [TAN91]. Los protocolos son usados en las topologías de bus y anillo debido a que los dispositivos están conectados a un mismo medio de transmisión, por lo cual si no existieran dichos protocolos, en el caso de que dos o más computadoras transmitan al mismo tiempo la interferencia en el medio puede causar que una o varias transmisiones resulten dañadas.

CSMA/CD significa Acceso Múltiple por Sensado de Portadora con Detección de Colisión (Carrier Sense Multiple Access / Collision Detection). En este protocolo, las computadoras escuchan el bus mientras transmiten para detectar una posible colisión, en caso de ser positiva la colisión abortan la transmisión y esperan un tiempo aleatorio antes de volver a sensar el bus.

Además de la información de utilidad para el usuario que se transmite a través de la red debe de contener otros campos requeridos para proporcionar servicio, a estos datos se le llama trama. La trama no debe ser menor a 64 bytes eliminado el preámbulo y el delimitador de inicio de trama. La trama contiene lo siguiente:

a) Preámbulo. Es un campo de 7 bytes con el código 10101010. Al transmitir este byte en codificación Manchester, se genera una señal cuadrada que sirve para sincronizar a los receptores en la red.

b) Delimitador de inicio de trama (SFD, Start Frame Deliniter 1 byte). Un byte formado por el patrón 10101011. El último par de bits interrumpe la onda cuadrada formada por el preámbulo y los primeros bits de este byte. Esta interrupción sirve para indicar dónde en realidad inician los campos con información útil.

c) Dirección Destino (2 ó 6 bytes). Cada computadora en la red tiene un identificador único que es su dirección. Este campo contiene la dirección del sitio al que se envía la trama. Se utilizan 6 bytes para los campos de direcciones. Los 2^{48} valores garantizan que ningún otro dispositivo en el mundo pueda tener la misma dirección.

d) Dirección Fuente (2 ó 6 bytes). Contiene la dirección de la computadora que generó la trama.

e) Tipo de trama o longitud (2 bytes). En la mayor parte de los casos esto permite identificar cómo interpretar el campo de datos, pues indica el tipo de protocolo de capas superiores que lleva la trama. También este campo puede indicar el número de bytes válidos en el campo de información, tomando valores de 0 a 1500.

f) Información. En este campo viaja la información que se transfiere de un sitio a otro. Como se ha mencionado anteriormente las tramas deben tener un tamaño mínimo de 64 bytes. Para determinar el tamaño de la trama se toman en cuenta todos los campos excepto el preámbulo y el delimitador, por lo que para LAN con direcciones de 6 bytes, el tamaño mínimo del campo de información debe ser de 46 bytes. En caso de que la computadora transmisora quiera enviar menos de 46 bytes, se deben agregar caracteres de relleno conocidos como padding.

g) Secuencia de Verificación de la trama (FCS, Frame Check Secuence, 4 bytes). El último campo es un código cíclico de redundancia (CRC) de 32 bits, calculado con los campos de dirección fuente y destino, tipo o longitud e información para tratar de predecir la integridad de la trama: la computadora calcula el CRC y lo añade a la trama cuando lo envía por la red, la computadora destino recibe la trama y repite el cálculo, si el valor calculado es diferente al recibido, se supone que la trama ha sido alterada por el ruido en el medio, por lo que se descarta.

Estándar IEEE 802.4, token bus.

El token es transmitido de un equipo al siguiente, cuando un equipo recibe el token éste puede transmitir datos en un tiempo límite. Una vez terminada su transmisión o terminado su tiempo pasa el token al siguiente equipo. Si el equipo que recibe el token no tiene datos que transferir pasa el token de inmediato al siguiente equipo y así sucesivamente. Como la red físicamente es un bus, lógicamente se configura como un anillo para el paso de testigo de un equipo al otro.

a) Preámbulo (1-n bytes). Cada trama comienza con un preámbulo que es usado por la estación receptora para la sincronización. La longitud varía dependiendo del método de modulación.

b) Delimitador inicial (SD, Starting Delimiter). Su longitud es de un byte y contiene una señal de un dato.

c) Control de Trama (FC, Frame Control). Es de 1 byte e identifica el tipo de trama que está siendo enviada

d) Dirección destino (DA, Destination Address). La dirección destino puede ser tan sólo una dirección o un grupo de direcciones. Mide de 2 a 6 bytes.

e) Dirección Origen (SA, Source Address). De un tamaño de 2 a 6 bytes, la dirección origen a diferencia de la destino es única.

f) Campo de Información (IF, Information Field, 0-819 Bytes). Este campo puede contener una unidad de enlace, un controlador de señal, un manejador de datos o datos de propósito especial.

g) Secuencia de Verificación de Trama (FCS, Frame Check Sequence). Cuando el equipo emisor conjunta una trama realiza una revisión redundante cíclica (CRC), esta fórmula fue descrita por la IEEE la cual requiere 4 bytes para el resultado. El equipo emisor almacena el resultado en este campo. Cuando el equipo receptor recibe la trama realiza la misma fórmula que el emisor y el resultado lo compara con el de este campo (FCS). Si los resultados no son iguales, el equipo receptor asume que la transmisión tuvo algún error y puede nuevamente requerir que la trama sea retransmitida.

h) Delimitador final (ED, Ending Delimiter). Es similar al delimitador inicial y marca el fin de la trama.

Estándar IEEE 802.5, token ring.

La IEEE en 1989 dió a conocer el estándar 802.5 describiendo las características para una LAN del tipo Token Ring como: la topología en anillo, el cable par trenzado, el tipo de codificación Manchester diferencial, con velocidad de 4 a 16 Mbps y protocolo de acceso paso de token. [MAR89]. Existen diversas técnicas para que en una topología en anillo se pueda controlar el acceso al medio, la más popular es Token Ring.

Las unidades de datos son transmitidas de una computadora a la próxima en una secuencia física a lo largo del anillo. Cada computadora transmite la unidad de datos a la próxima computadora actuando como un repetidor. La transmisión de unidades de datos está controlada por un token. Cuando una computadora recibe el token, puede transmitir unidades de datos mientras un tiempo límite está corriendo. Un token con una configuración que indica libre para transmitir es llamado free token (testigo libre). Cuando una computadora recibe un free token y tiene unidades de datos para

transmitir, cambia la configuración del token de free token a busy token (testigo ocupado) e incluye el propio busy token con la unidad de datos que transmita. La unidad de datos viaja de computadora en computadora alrededor del anillo. Cada computadora que recibe una unidad de datos realiza una verificación, analizando el campo de control (FC) en caso de tratarse de una unidad de datos de tipo MAC, la información será procesada por todas las computadoras del anillo, si es de tipo LLC y coincide con la dirección destino es procesada. En caso de que no vaya dirigida a la computadora que en instancia tenga el busy token, lo retransmite a la siguiente computadora, y así sucesivamente hasta que encuentre su destino.

Para evitar que una computadora transmita indefinidamente, se establece un tiempo máximo de posesión del token, después del cual deberá emitirse un nuevo token que proporcione la oportunidad de transmitir tramas a las demás computadoras. Cuando una unidad de datos retorna a la computadora que originalmente lo mandó, ésta remueve la unidad de datos de la red y manda un free token a través de la red a la siguiente computadora.

a) Delimitador inicial (SD, Starting Delimiter). Es una señal única que identifica el inicio de la trama y está formado por la siguiente secuencia de bits: "JK0JK000".

b) Control de Acceso (AC, Access Control). Este byte está formado por los siguientes bits "PPPTMRRR". Los bits PPP indican la prioridad del token o de la trama. El bit T tiene un valor de "0" en caso de tratarse de un free token. Cuando una computadora desea transmitir espera el free token y cambia el valor del bit a "1". El bit M se transmite con un valor de "0". Cuando el monitor activo lo retransmite, le cambia el valor a "1". Esto ayuda a la detección y corrección de situaciones en las que una trama de alta prioridad se transmite indefinidamente. Los bits RRR sirven para reservar la prioridad del siguiente free token.

c) Control de Trama (FC, Frame Control). Está formado de la siguiente cadena de bits "FFZZZZZZ". Los bits FF indican el tipo de trama (00=MAC, 01=LLC), los bits ZZZZZZ identifican un subtipo de la trama.

d) Dirección destino (DA, Destination Address). Todas las direcciones de la red deberán tener la misma longitud. El primer bit indica si se trata de una dirección individual o de grupo, mientras que el segundo bit, para el caso de direcciones de 6 bytes indica si se trata de una dirección local o universal.

e) Dirección Origen (SA, Source Address). Debe tener la misma forma y longitud que la dirección destino, y el primer bit debe valer 0.

f) Campo de Información (IF, Information Field). Contiene la información destinada a las capas MAC o LLC. La longitud máxima es de 4.48 Kbytes en el caso de una red que soporte 4 Mbps y de 17.984 Kbytes para una red de 16 Mbps.

g) Secuencia de Verificación de Trama (FCS, Frame Check Sequence). Contiene información para detectar errores de transmisión. La secuencia se calcula a partir del FC, DA, SA y de información.

h) Delimitador de final (ED, Ending Delimiter). Contiene la siguiente cadena de bits "JK1JK1IE", El bit I puede usarse para determinar el fin de transmisión de una estación. Para la transmisión de una serie de tramas el bit I=1 en la primera trama e intermedias; I=0 cuando se transmita una sola trama o cuando se transmite la última trama. El bit E se transmite con valor igual a 0, cuando se detecta un error el bit E=1.

i) Estado de la Trama (FS, Frame Status). Contiene los bits de reconocimiento de la dirección y la trama copiada que son usados para indicar si la trama fue o no exitosamente recibida por la computadora destino.

ATM.

ATM quiere decir Asynchronous Transfer Mode, Modo de Transferencia Asíncrono, sirve para transmisión de datos a alta velocidad como por ejemplo: voz, datos, vídeo en tiempo real y sonido; usando como medio cables de fibra óptica.

Una red ATM contiene conmutadores ATM, los cuales son generalmente dispositivos multipuertos que realizan conmutación de celdas. Cuando una celda llega a un puerto, el conmutador ATM examina la información del destino de la celda y la envía por el puerto de salida apropiado. Las funciones de conmutación se realizan por hardware incrementando así la velocidad.

Originalmente se definió ATM como parte de la Red Digital de Servicios Integrados de Banda ancha (B-ISDN Broadband-Integrate Services Digital Network) que desarrolló el CCITT.

En 1991 se formó el Forum ATM el cual definió dos métodos de interfaz física:

a) Interfaz de usuario a red (UNIs, User-to-Network Interfaces). El UNI es el punto de conexión de las estaciones finales a una red ATM.

b) Interfaces de red a red (NNIs, Network-to-Network Interface) Consisten en interfaces de interoperabilidad de conmutadores ATM.

El modelo de referencia B-ISDN.

El nivel físico define las interfaces eléctricas y físicas, las velocidades de las líneas y otras características físicas. Lo interesante es que no se define ningún medio físico, pero se recomienda FDDI (100 Mb/s), canal de fibra (155 Mb/s), OC3 SONET (155 Mb/s) y T3 (45 Mb/s). El nivel ATM define el formato de celda, el canal virtual, el trayecto y el control de errores. Las celdas tienen una longitud de 53 bytes, 48 de ellos los ocupa el campo de datos y los otros 5 la información de la cabecera.

El nivel de adaptación ATM (AAL, ATM Adaptation Layer) define el proceso de conversación de información de los niveles superiores en las celdas ATM, es decir, divide los paquetes de los niveles superiores en celdas de 48 bytes para su transporte. Este nivel se divide en dos subniveles: El subnivel de convergencia (CS, Convergence Sublayer) que acepta los datos del nivel más alto y los pasa al subnivel de segmentación y reensamblado (SAR, Segmentation and Reasembly), que es el responsable de la segmentación de la información en celdas ATM de 53 bytes. Si llegan celdas, el SAR reensambla los datos y los pasa al nivel superior. Existen varios tipos de AAL:

. Tipo 1 es un servicio de capacidad constante isócrono para las aplicaciones de audio y vídeo.

. Tipo 2 es una aplicación isócrona de capacidad variable como vídeo comprimido.

. Tipo 3-4 soporta ráfagas de datos de capacidad variable tipo LAN que soporta interfaces Frame Relay e SMDS.

. Tipo 5 soporta un subconjunto de funciones del tipo 3-4, proporciona modo de mensajes y operaciones no seguras.

Servicios ofrecidos.

. Clase A, es un servicio orientado a la conexión que proporciona una capacidad constante. Las compensaciones de sincronización lo hacen adecuado para aplicaciones de vídeo y voz.

. Clase B, es un servicio orientado a la conexión y se sincroniza para la transmisión de capacidades variables de voz y vídeo.

. Clase C, es un servicio orientado a la conexión de capacidad variable sin sincronismos adecuados para los servicios como X.25, Frame Relay y TCP/IP.

. Clase D, es un servicio sin conexión, con un tráfico de datos de velocidad variable que no requiere relaciones de sincronización entre los nodos finales.

FDDI.

FDDI significa interfaz de datos distribuida por fibra (Fiber Distributed Data Interface) [SAN93] y es un estándar que fue desarrollado por el Instituto Nacional de Estándares de América (ANSI: American National Standards Institute). Está basado para el uso de fibra óptica, control de acceso por paso de testigo y con capacidad para transportar 100Mbps.

Las ventajas principales de este protocolo son:

a) La interconexión de mainframes.

b) Gran velocidad en la transferencia de imágenes y gráficos.

c) Una red con alta velocidad que pueda servir para interconectar redes locales de menor capacidad.

Protocolo de acceso MAC.

Se controla por la posesión de un token. Si ninguna estación en la red quiere transmitir, el token circula sobre el anillo. Cuando una estación quiere transmitir, espera el token, lo remueve del anillo y transmite su trama. La alta velocidad de transmisión de FDDI, el tamaño máximo que pueden tener las tramas y la extensión física que puede alcanzar la red hacen que el tiempo de transmisión de una trama puede ser pequeño en comparación al retardo de propagación de la trama alrededor del anillo. Por esta razón para aumentar la eficiencia en el uso de la red, la estación que transmite pone en circulación un nuevo token inmediatamente después de terminar una transmisión.

a) Preámbulo (P, Preamble, 16 símbolos idle o 64 bits), Es usado para sincronizar el reloj de cada computadora en la red con la transmisión. La trama envía este campo con 16 símbolos idle, los nodos que repiten la trama alrededor del anillo pueden cambiar la longitud de este campo de acuerdo a sus requerimientos de sincronización.

b) Delimitador de inicio (SD, Starting Delimiter, 2 símbolos idle), Es una señal que identifica el comienzo de la trama. Consta de un símbolo J y otro K.

c) Control de la trama (FC, Frame Control). Sirve para identificar el tipo de trama. Este tiene el siguiente formato de bits CLFFZZZZ, donde bit C específica si se trata de una trama asíncrona o síncrona, el bit L específica si las direcciones tienen una longitud de 16 ó 48 bits, los bits FF indican si la trama es un LLC o MAC, y los bits ZZZZ proporcionan el control de información para tramas de MAC.

d) Dirección destino (DA, Destination Address, 4 ó 12 símbolos idle). Especifica la estación a la cual va dirigida la trama. El anillo puede contener una mezcla de computadoras usando direcciones de 16 bits (4 idle) o 48 bits (12 idle). La dirección destino puede ser individual, de grupo o para todos.

e) Dirección fuente (SA, Source Address, 4 ó 12 símbolos idle). Identifica la computadora que envió la trama.

f) Información (IF, Information Field, 0-n símbolos idle). Contiene datos del usuario LLC o información de control MAC. La longitud máxima de una trama es de 4500 bytes.

g) Secuencia de verificación de trama (FCS, Frame Check Secuence, 8 símbolos idle). Este campo contiene un valor de 32 bits. El valor se calcula tomando como base los siguientes campos: Control de Trama, Dirección Destino, Dirección Fuente e Información. La computadora que recibe la trama realiza el mismo cálculo y compara con el valor almacenado en FCS.

h) Delimitador de fin (ED, Ending Delimiter, 1 ó 2 símbolos idle). Identifica el fin de una trama. Consiste de uno o dos símbolos de terminación (T).

i) Estado de la trama (FS, Frame Status, 1 símbolo idle). Como su nombre lo indica, contiene información, algunas banderas, sobre la trama como son si hubo detección de algún error, si las direcciones se reconocieron y si la trama fue copiada con éxito.

Frame relay.

Se utiliza principalmente para la interconexión de LANs y WANs sobre redes públicas o privadas y ofrece un ancho de banda entre 56 Kb/s y 1.544 Mb/s. Frame Relay proporciona circuitos virtuales permanentes a través de una trayectoria predefinida que conecta dos puntos finales. Los canales permanecerán activos continuamente y garantizan las especificaciones que se han negociado con el cliente. Frame Relay evita el control de flujo y la gestión de errores dentro de la propia red, cuando estos últimos se presentan, Frame Relay asume que los nodos finales son máquinas programables que pueden realizar su propia gestión de errores. En Frame Relay los nodos simplemente realizan una retransmisión de tramas a través de una trayectoria predefinida.

Servicios de red.

a) Mensajes de estado de los circuitos virtuales. Este servicio proporciona la comunicación entre la red y el cliente. Asegura que el PVC exista e informa sobre los PVCs eliminados.

b) Multidistribución. Este servicio es opcional y permite a un usuario enviar tramas a múltiples destinos.

c) Direccionamiento global. Este servicio opcional concede a una red Frame Relay capacidades semejantes a las de una LAN.

d) Control simple de flujo. Es un servicio opcional que proporciona un control de flujo XON/XOFF a los dispositivos que así lo requieran.

Cuando una red Frame Relay alcanza la congestión, las tramas pueden descartarse arbitrariamente siendo los nodos finales los responsables de la retransmisión.

Una red Frame Relay conecta dos LANs sobre una red pública de conmutación de paquetes. El proceso es muy simple: la trama procedente de la LAN se sitúa en una trama Frame Relay y se distribuye a través del substrato de red hacia el destino. Técnicas de mutiplexión estadística entrelazan los datos procedentes de múltiples orígenes en la ubicación del cliente sobre una única línea. Los indicadores delimitan la trama mediante una secuencia especial de bits.

El encabezamiento mantiene la información sobre direcciones y control de congestionamiento. Este campo contiene la siguiente información:

a) Identificador de conexión de enlace de datos (DLCI, Data Link Connection Identifier), que contiene el número de identificación para la conexión lógica multiplexada dentro del canal.

b) Capacidad de elección de descarte (DE, Discart Eligibility), que establece las prioridades si una trama puede descartarse durante un congestionamiento.

c) Notificación de congestión explícita hacia delante (FECN, Forward Explicit Congestión Notification), informa al ruteador que se recibe la trama que se ha experimentado congestión en el trayecto que ha atravesado ésta.

d) Notificación de congestión explícita hacia atrás (BECN, Backward Explicit Congestión Notification), añade información a las tramas que viajan en sentido contrario para ayudar a los protocolos con el fin de emprender la acción adecuada para el control de flujo.

e) El siguiente campo corresponde a la información.

f) Después se añade un campo para la secuencia de verificación de trama (FCS, Frame Check Sequence), se realiza un cálculo con cada trama recibida y se compara con el campo FCS, que calculó el emisor. El paquete se elimina si no hay coincidencia y las estaciones finales deben solucionar su pérdida.

1.3

Bases de datos

1.3.1 ¿ Qué es una base de datos ?

Aunque los datos de todos los orígenes y representaciones han estado presentes a lo largo de la historia de la humanidad, el concepto de bases de datos orientadas hacia computadoras, data de la década de los sesentas.

Una definición simple de lo que es un sistema de bases de datos es: un sistema cuyo objetivo es registrar y mantener información, formado por cuatro componentes fundamentales: datos, hardware, software y usuarios.

Datos.

Se pueden almacenar en varias bases de datos, mismas que cumplen con las características de integración y compartición. Por integrada se entiende una unión de varios archivos, reduciendo la redundancia. Compartida se refiere a que la base de datos puede ser accesada simultáneamente por diferentes usuarios.

Hardware.

Este componente lo forman los medios de almacenamiento secundario donde reside físicamente la base de datos.

Software.

Existe una capa de software entre la base de datos física y los usuarios, que se denomina "Sistema de Administración de Bases de Datos" o DBMS por sus siglas en inglés (Data Base Management System).

Usuarios.

Los usuarios se clasifican en tres categorías:

* Programador de aplicaciones. Su función es escribir programas que exploten la información de las bases de datos.

* Usuarios finales. Explota los datos basándose en los programas escritos por el programador de aplicaciones o usando los que se proporcionan como parte del DBMS.

* DBA (Data base administrator). El administrador de la base de datos se encarga de controlar los datos y las operaciones que sobre ellos realice cualquier elemento de la empresa.

Los datos tienen tres clasificaciones: de entrada, de operación y de salida. Los datos de entrada pueden ocasionar modificaciones a los mismos. Los datos de salida se derivan de los datos de operación, pero al igual que los de entrada son de carácter transitorio, por lo tanto no forman parte de la base de datos.

Entidades y atributos.

Una entidad es cualquier objeto que pueda representarse en la base de datos y del cual se desea registrar y consultar información. Un atributo es una característica de una entidad. Por ejemplo, una entidad es un alumno y algunos atributos son su número de cuenta, su nombre, su sexo, etcétera.

1.3.2 Ventajas de tener un control centralizado de los datos.

La época que estamos viviendo se le denomina "la era de la información". Hoy en día resulta evidente el hecho de que una empresa deba tener un control centralizado de sus datos de operación, mismos que le permitan estar preparada para responder con oportunidad, calidad y a bajo costo a las exigencias de una economía global en la que vivimos.

1.- Reducir la redundancia. Al existir una sola copia de los datos se evitan desperdicios innecesarios ocasionados por tener varias copias de la misma información. En algunos casos la redundancia es intencional y se deben tener mecanismos de actualización en todos los archivos, para lograr lo que se denomina redundancia controlada.

2.- Evitar la inconsistencia. Cuando la redundancia existe, es probable que los datos no se actualicen en todas partes y al efectuar consultas arrojen resultados diferentes. A este fenómeno se le denomina inconsistencia y se resuelve evitando la redundancia o al menos controlándola.

3.- Compartir los datos. Todas las aplicaciones pueden usar la información almacenada, y en caso de que se creen una nuevas aplicaciones, pueden operar con los datos existentes.

4.- Fijación de estándares. Resulta más sencillo unificar los formatos de los datos que auxiliarán en caso de intercambio o migración de datos entre sistemas.

5.- Seguridad. El DBA puede diseñar un esquema de seguridad que permita el acceso a los datos dependiendo de la categoría de cada usuario, protegiendo así la información.

6.- Exactitud de la información. Es posible implantar mecanismos de verificación de datos que consecuenten la inexistencia de datos absurdos, v.gr. más de 24 horas al día, empleados inexistentes, etcétera.

7.- Independencia de los datos. "Es la inmunidad de las aplicaciones a los cambios de la estructura de almacenamiento y a la estrategia de acceso a los datos." [DAT81] Cuando no existe independencia, es necesario invertir fuertes cantidades de recursos para adaptar las aplicaciones a algún cambio en la organización de los datos y la forma de acceso. La independencia de los datos es un objetivo central en el diseño de los sistemas de bases de datos.

1.3.3 Diccionario de datos.

Entre las funciones de un diccionario de datos se encuentran el control de campos de datos, reducción de redundancia e inconsistencia, determinación del impacto de los cambios en los campos sobre el total de la base de datos, centralización del control de los campos de datos como ayuda en el diseño y expansión de la base de datos, registro de los programas que se usan con la base de datos y mantenimiento de información respecto a códigos de seguridad.

"Un diccionario de datos es un depósito central de información acerca de las entidades, los campos de datos que representan a las entidades, las relaciones entre éstas, sus orígenes, significados, usos y formatos de representación." [SHA90]

1.3.4 Arquitectura de un sistema de base de datos.

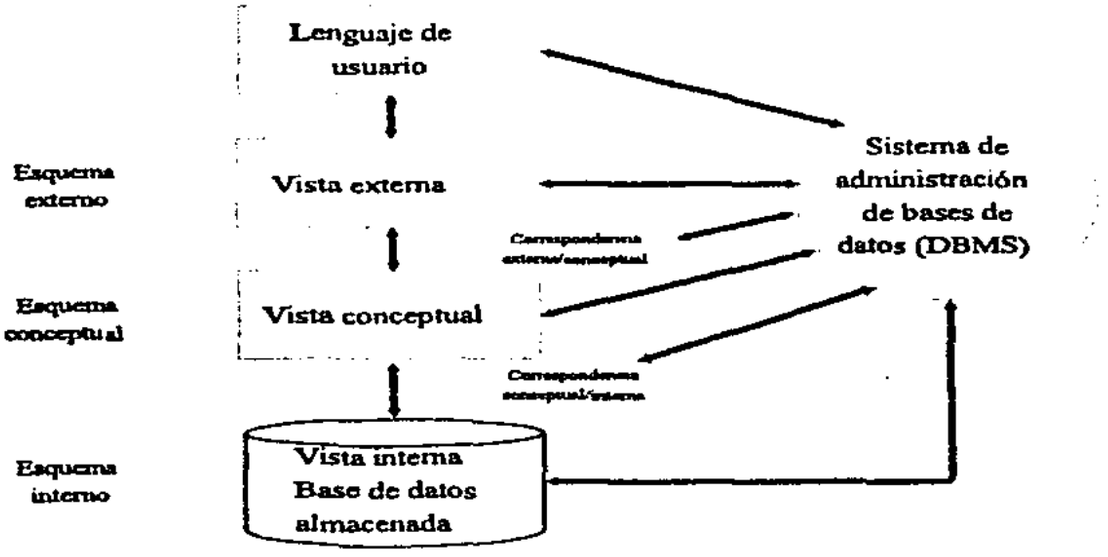


Fig. 1.3.4.1 Arquitectura del sistema de base de datos.

El modelo de arquitectura estudiado se divide en tres niveles: interno, conceptual y externo. El nivel interno se refiere a la manera en que los datos son almacenados físicamente, el externo es el más cercano a la forma en que los usuarios ven los datos y el conceptual se sitúa como mediador entre estos dos.

Elementos de la arquitectura de base de datos.

1.- Lenguaje de usuario.

Para los usuarios del sistema de base de datos existe un lenguaje a su disposición, que puede ser uno convencional para el caso de los programadores de aplicaciones, y uno especial para los usuarios finales.

El lenguaje de usuario incluye un sublenguaje de datos (DSL) que está inmerso en un lenguaje anfitrión. El DSL es una combinación de un lenguaje de definición de datos (DDL, Data Definition Language) que permite el manejo de la información de la base de datos tal como la aprecia el usuario, y un lenguaje de manipulación de datos (DML, Data Manipulation Language) que apoya el procesamiento de esa información.

2.- Vistas de la arquitectura.

A) Vista externa.

Un usuario sólo ve la parte de la base de datos que le interesa. Una vista externa es el contenido de la base de datos como la aprecia un usuario, está formada por numerosas ocurrencias de varios registros externos. Para definir esta vista se usa el esquema externo, que se compone de las definiciones de los tipos de registros externos de esa vista. Este esquema se crea con el DDL externo.

B) Vista conceptual.

Es una representación abstracta del contenido total de la base de datos. El esquema conceptual que define a esta vista se crea utilizando el DDL conceptual, que incluye definiciones conceptuales de los registros. Para lograr independencia en la BD es preciso que no se hagan consideraciones en este nivel sobre el almacenamiento y los métodos de acceso. El esquema conceptual incluye además la definición de los controles de autorización y procedimientos de validación.

C) Vista interna.

Es la representación a bajo nivel del total de la base de datos. Está compuesta de varias ocurrencias de registros internos. La vista interna se describe por medio del esquema interno, donde se especifican los diversos tipos de registros almacenados, los índices, la forma de representar los campos almacenados y la secuencia física en que se encuentran los registros. Este esquema se escribe utilizando el DDL interno.

3.- Correspondencias.

Entre las vistas existe lo que se denomina correspondencia. La conceptual/interna especifica la manera en que los registros conceptuales y los campos corresponden con sus contrapartes almacenadas. Cuando se da un cambio en la estructura de almacenamiento, debe reflejarse en el esquema conceptual.

La externa/conceptual define la correspondencia de las vistas con ese nombre, donde las diferencias que se presentan, son las mismas que en el caso anterior. Existe la posibilidad de que varias vistas externas utilicen la misma correspondencia externa/conceptual.

4.- Sistema de administración de base de datos (DBMS)

Es el software que maneja todos los accesos a la base de datos. La forma de operar es la siguiente: un usuario valiéndose de un DML emite una solicitud de acceso, el DBMS la capta y la interpreta, el DBMS inspecciona el esquema externo, la correspondencia externa/conceptual, el esquema conceptual, la correspondencia conceptual/interna y la definición de la estructura de almacenamiento, finalmente el DBMS realiza las operaciones requeridas sobre la base de datos. Todo este proceso de interpretación es usualmente precompilado para ganar tiempo de respuesta.

5.- Administrador de la base de datos.

Entre sus funciones tenemos:

a) Decidir el contenido de la base de datos.
b) Definición de la estructura de almacenamiento y las estrategias de acceso.
c) Proveer disponibilidad de los datos y controlar el desempeño.
d) Definición del esquema de seguridad.
e) Establecimiento de estrategias de respaldo y recuperación.

6.- Interfaz con el usuario.

Se localiza en el nivel externo y es el elemento tangible con el cual el usuario interactúa con la base de datos.

1.3.5 Modelo de datos.

Un sistema de manejo de base de datos utiliza un modelo de datos para definir la estructura fundamental de los mismos. Las entidades de una empresa y las relaciones entre ellas se pueden representar por un modelo de datos.

Relaciones dentro de un modelo de datos.

Una relación es una unión entre dos conjuntos de datos. Las posibles combinaciones son: "uno a uno", "uno a muchos" y "muchos a muchos". Las relaciones pueden darse entre entidades o entre atributos.

* Relación "uno a uno". Cada elemento (entidad o atributo) tiene sólo una correspondencia en el otro elemento.

* Relación "uno a muchos". Las entidades o atributos pueden tener varias correpondencias en el otro conjunto.

* Relación "muchos a muchos". Los elementos de ambos conjuntos pueden tener varias correspondencias entre sí.

1.3.5.1 Modelo de datos jerárquico

Está formado por una entidad dominante y una o varias entidades subordinadas. En este caso se da una relación "uno a varios", así, para una del tipo dominante puede haber varios tipos de subordinadas.

1.3.5.2 Modelo de datos de red.

Se tiene una ampliación del concepto jerárquico, donde cualquier entidad puede ser dominante o subordinada, que en este caso se denominan propietarias y miembros. Incluso las entidades pueden caer en ambas categorías simultáneamente incidiendo en un número ilimitado de relaciones.

1.3.5.3 El modelo relacional.

Como me baso en este modelo para el desarrollo de esta tesis, es el que tiene un estudio más profundo.

Este enfoque para el estudio de las bases de datos se funda en el concepto matemático de relación. "Lo anterior se basa en el hecho de que los archivos que obedecen ciertas restricciones se pueden considerar relaciones matemáticas y, por tanto, la teoría elemental de las relaciones se puede aplicar a varios problemas prácticos de vérselas con datos de esos archivos." [DAT81]

Relaciones.

La definición matemática de relación es: "Dada una serie de conjuntos D1,D2,...,Dn, no necesariamente distintos, se dice que R es una relación sobre estos n conjuntos si es un conjunto de n tuplas ordenadas d1,d2,...,dn, tales que d1 pertenece a D1, d2 pertenece a D2,..., dn pertenece a Dn. Los conjuntos D1, D2,..., Dn son los dominios de R. El valor de n es el grado de R."

En este modelo las tablas se denominan relaciones, los renglones o registros se conocen como tuplas, y se llama dominio al depósito de valores del cual se extraen los que aparecen en una columna específica. Las asociaciones entre tuplas se representan por valores de datos en columnas que provienen de un dominio común.

Base de datos relacional.

Una base de datos relacional es un conjunto de relaciones normalizadas de diversos grados que varían con el tiempo.

1.3.5.3.1 Dominios y atributos.

Un atributo representa el uso de un dominio dentro de una relación.

Una relación debe estar normalizada, es decir, en cada intersección de un renglón y una columna de la tabla siempre hay exactamente un valor, nunca un conjunto de valores. Aunque dependiendo del contexto se permite la existencia de valores nulos.

Un dominio simple es aquel donde todos los elementos son atómicos. Para que una relación esté normalizada es requisito que todos sus dominios sean simples.

1.3.5.3.2 Llaves.

Una llave primaria es un atributo cuyos valores son únicos y se pueden usar para identificar las tuplas de esa relación. Es posible que la llave primaria se forme con la combinación de más de un atributo. Toda relación tiene una llave primaria que es además no redundante.

En algunas relaciones puede haber más de un atributo que posea la propiedad de identificación única, en estos casos se afirma que existe más de una llave candidata. Cuando una llave candidata no es llave primaria recibe el nombre de llave alterna.

1.3.5.3.3 Reglas de integridad.

Integridad de la entidad.

Las llaves primarias realizan la función de identificación única en un modelo relacional de base de datos, por lo tanto ninguno de sus componentes puede tener un valor nulo.

Integridad de referencia.

Un dominio se considera primario si y sólo si existe alguna llave primaria de un solo atributo definida sobre ese dominio.

"Sea D un dominio primario, y sea R1 una relación con un atributo A que se define sobre D. Entonces, en cualquier instante dado, cada valor de A en R1 debe ser o bien nulo, o bien igual a V, donde V es el valor de la llave primaria de alguna tupla de alguna relación R2 con llave primaria definida sobre D." [SHA90]

1.3.5.3.4 Extensiones y comprensiones.

La extensión de una relación es el número de tuplas que aparecen en la relación en un instante específico, por lo tanto varían con el tiempo en la medida en que se crean, eliminan o actualizan. La comprensión es invariable con el tiempo y constituye lo que se especifica en el modelo relacional. Esto es, el nombre de la relación y de los atributos.

1.3.5.3.5 Ventajas del modelo relacional.

Fundamentos matemáticos. El modelo se basa en la teoría matemática de las relaciones. El método de diseño usando la normalización le da un fundamento sólido que no existe para los otros dos modelos.

Simplicidad. El usuario final observa la información desde la perspectiva del modelo de datos, no de la forma en que están almacenados.

Consultas no planeadas. Gracias a que no hay dependencia de posición entre las relaciones, las consultas no requieren tener una estructura dada.

Independencia de los datos. Este modelo elimina los detalles relativos a la estructura de almacenamiento y la estrategia de acceso desde la interfaz de usuario.

1.3.6 Diseño de bases de datos.

1.3.6.1 Modelo conceptual.

El primer paso en el diseño de las bases de datos es generar el modelo conceptual. El modelo conceptual representa las entidades de la empresa y las relaciones entre ellas, basadas en las necesidades de procesamiento de datos de la misma.

El modelo conceptual es independiente de aplicaciones individuales, del DBMS, del hardware usado para almacenar datos y del modelo físico en el medio de almacenamiento. Este diseño se basa en conceptos de la teoría relacional, aunque es independiente del enfoque de realización final que se aplique, ya sea jerárquico, de red o relacional.

1.3.6.2 Metodología de diseño de bases de datos.

1.- Recopilación de información respecto a los datos para aplicaciones existentes.

El equipo de trabajo responsable del diseño de la base de datos debe obtener, por algún medio, los datos que se necesitan en cada nivel de la organización, así como el tratamiento que debe darse a los mismos, ya sea que se procesen o simplemente se almacenen.

En esta etapa debe reunirse:

a) Nombre y descripción de entidades y campos.
b) Atributos (tipo y límites asociados).
c) Fuentes de los datos.
d) Seguridad requerida.
e) Valor de los datos.
f) Relaciones de campos y entidades.
g) Flujo de datos.

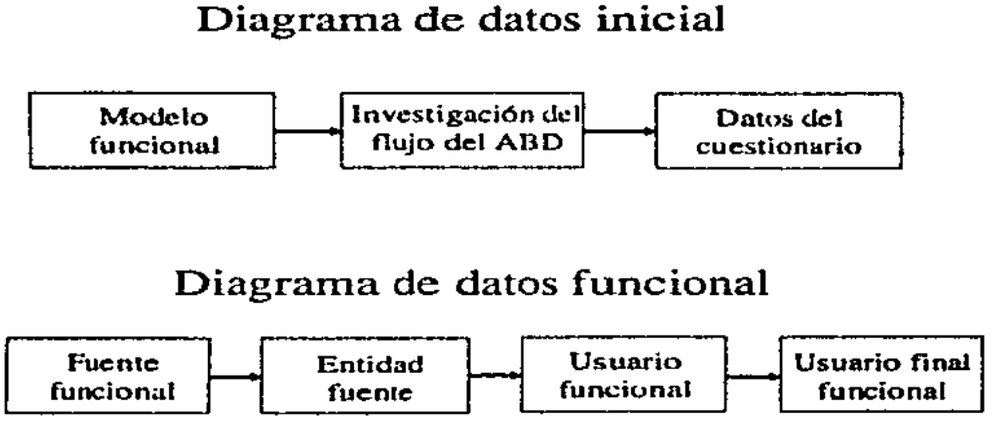


Fig. 1.3.6.2.1 Diagrama de datos.

Con la información descrita anteriormente se realiza el análisis del depósito de los datos. En este estudio se deben mostrar las relaciones entre entidades y campos, indicando las fuentes de información y los usuarios de la misma.

2.- Recopilación de información para aplicaciones futuras.

Un buen diseño de la base de datos debe considerar que a medida que los usuarios de la información asimilen el valor que ésta les proporciona, solicitarán nuevas relaciones, entidades y campos. Es decir, la base de datos debe diseñarse pensando en las demandas futuras de datos que la empresa necesite.

3.- Normalización de la base de datos.

Normalizar consiste en agrupar a los campos de datos en tablas que representan a las entidades y sus relaciones. La teoría de la normalización se basa en el hecho de que un buen diseño de relaciones tiene mejores propiedades para insertar, actualizar y eliminar datos, que otros.

Primera forma normal.

Consiste en transformar los campos de datos a una tabla de dos dimensiones. Cada relación es una tabla, en donde en cada intersección de un renglón y una columna sólo puede haber un valor en la tabla. Esto significa que no se permiten conjuntos de valores en las intersecciones.

En la primera forma normal todos los atributos que no son clave en la relación son funcionalmente dependientes de la clave primaria, esto significa que dado el valor de la clave primaria, quedan determinados de manera única los valores tomados por los atributos que no son clave. En esta forma pueden darse fallas de inserción, de actualización y de supresión.

Segunda forma normal.

Una relación está en la segunda forma normal, cuando todo atributo que no sea clave es completamente dependiente de manera funcional de la clave primaria. De esta manera, todo atributo que no es clave necesita de la clave primaria completa para poder ser identificado de manera única.

De cualquier manera, siguen existiendo fallas de inserción, supresión y actualización, a consecuencia de la dependencia del atributo que no es clave de otro atributo que tampoco es clave. A esto se le denomina "dependencia transitiva".

Tercera forma normal.

Para que se considere que una relación esté en la tercera forma normal es necesario que no exista ninguna dependencia funcional transitiva entre los atributos que no son clave. Si se diera el caso de que un atributo que no es clave se pudiera determinar con uno o más atributos que tampoco lo sean, se dice que existe una dependencia funcional transitiva entre ambos.

Cuando se llega a la tercera forma normal, se eliminan los problemas de inserción, actualización y borrado de información.

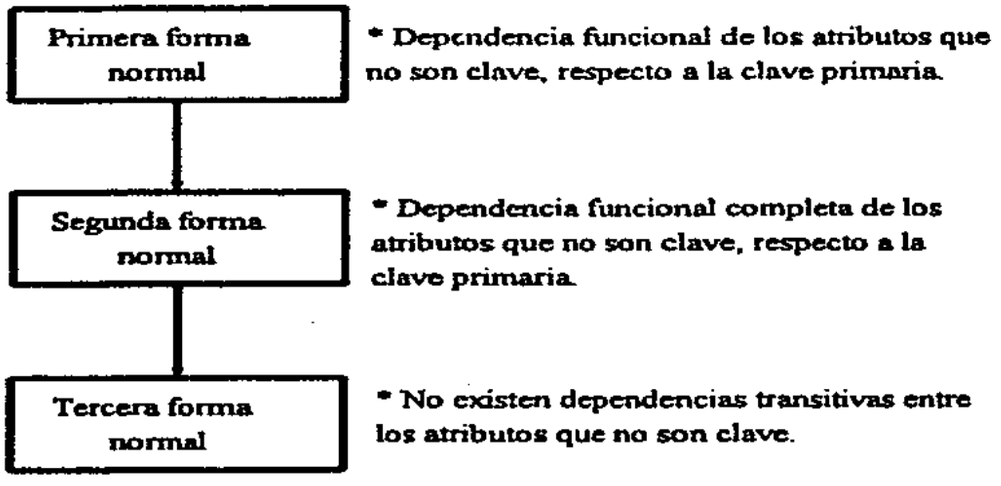


Fig.1.3.6.2.2 Normalización de bases de datos.

1.3.6.3 Modelo lógico.

En esta fase del diseño, se debe decidir qué modelo de datos aplicará para la base de datos, ya sea relacional, jerárquico o de red.

Transformación a un modelo relacional.

Puesto que para llegar al modelo conceptual me apoyo en la teoría relacional, la transformación a un modelo lógico es directa. Por lo tanto, esta actividad se reduce a convertir cada cuadro del modelo conceptual en una relación o tabla.

1.3.6.4 Modelo físico.

Esta fase se subdivide en dos: la primera parte es el diseño del modelo físico en sí y la segunda es la evaluación del funcionamiento del mismo.

El modelo físico es una estructura de base de datos que finalmente se debe almacenar en dispositivos físicos, donde para efectos de un mejor desempeño de los sistemas de bases de datos, se debe evaluar su funcionamiento.

Los detalles de esta etapa dependen del manejador escogido para el diseño. La conversión del modelo lógico al físico debe comprender la selección de los siguientes puntos:

a) Métodos de acceso.
b) Indices secundarios.
c) Asignación de dispositivos de almacenamiento.

Definidos los puntos anteriores, se debe hacer una estimación de escritorio del desempeño que tendrá la base de datos en dos direcciones: espacio físico de almacenamiento y tiempos de respuesta. Para este efecto, se requiere contar con información acerca de los volúmenes y tipos de datos esperados. que permitan estimar con precisión los espacios promedios a utilizar así como los tiempos de acceso que se deben esperar.

En caso de que el modelo físico planteado no alcance las expectativas de desempeño existentes y dado que el sistema de base de datos se encuentra aún en su etapa de diseño, es posible cambiar algunos aspectos del mismo. Hacerlo una vez que el sistema está operando en producción, implica un costo demasiado alto.

1.3.7 Requerimientos para una base de datos.

Al momento de escoger la base de datos que soportará nuestra aplicación debemos considerar una serie de elementos que nos brinden seguridad respecto a la elección. Estos aspectos son:

Independencia de plataforma.

Cuando por alguna razón hemos decidido actualizar el hardware, se debe tener la capacidad de que el manejador de base de datos siga funcionando como lo venía haciendo antes del cambio. Bajo este enfoque los datos sólo requieren respaldarse a priori y recuperarse a posteriori. El software debe ser compatible entre plataformas y los cambios requeridos deben ser mínimos.

Capacidad de realizar rollbacks.

Una transacción es un conjunto de operaciones que se llevan a cabo para completar una tarea. Las transacciones las genera el cliente y se mandan al servidor para su pocesamiento. Se considera una transacción exitosa si todas las operaciones se concluyeron. Si uno de los movimientos tuvo alguna falla todos las demás deben deshacerse.

Al tiempo que se ejecuta una transacción el DBMS posee una memoria de movimientos denominada "log". Si la transacción concluye con éxito el sistema guarda los cambios permanentemente, lo cual se conoce como "commit". En caso de errores, el sistema se apoya en el log para restablecer la base de datos al estado anterior a la ejecución, lo que se identifica como "rollback". El sistema de base de datos debe tener la capacidad de realizar rollbacks hasta el último commit.

Conectividad.

El servidor de base de datos requiere brindar acceso a múltiples fuentes de datos de diversa procedencia.

Procedimientos almacenados (stored procedures).

Son un conjunto de instrucciones SQL que residen en el servidor. Sirven entre otras cosas para mantener la integridad de los datos y manejar las reglas del negocio. Tienen la ventaja de que una vez ejecutados por primera vez, residen en memoria y sólo son llamados cuando se les necesite, lo que les brinda una gran velocidad.

Disparos (triggers).

Son stored procedures especiales que se ejecutan automáticamente cuando el DBMS lo solicite. Uno de los usos principales es garantizar la integridad referencial de la información.

Optimizador.

Es un software que analiza una instrucción SQL apoyándose en los índices estadísticos de distribución y el tamaño de las tablas para determinar el camino más eficiente en tiempo y costo para procesar lo solicitado.

Herramientas de prueba y diagnóstico.

Puesto que la información almacenada en una base de datos constituye la columna vertebral para algunas organizaciones, resulta indispensable tener utilerías que diagnostiquen los problemas con la base de datos y ofrezcan alternativas de solución.

Confiabilidad.

Cuando se tienen aplicaciones de misión crítica, es preciso contar con recursos adicionales que nos permitan seguir operando sin problemas. Estos elementos van desde un respaldo de datos hasta centros operativos espejo, es decir, lugares que tienen todos los elementos humanos y computacionales, por si el centro original sufriera algun desastre.

Mecanismos de respaldo y recuperación.

Se debe diseñar toda una estrategia de respaldo y recuperación de información, que nos permita no detener la operación de alguna institución. Estos respaldos regularmente se hacen en cintas magnéticas o en otros discos o servidores. La frecuencia de la operación varía dependiendo de la empresa y puede ir desde minutos, horas, días o semanas.

1.3.8 Comunicación cliente/servidor.

Los tres tipos de comunicación más usados son:

. Modelo conversacional.
. Llamadas de procedimiento remoto.
. Modelo de mensajes.

Modelo conversacional.

Se recomienda para sistemas que involucren muchas interacciones por transacción entre dos puntos, cuenta con rutinas que permiten ejecuciones traslapadas. Un producto comercial que utiliza este modelo es el APPC de IBM.

Modelo de mensajes.

Cuando un cliente necesita datos de un servidor de base de datos, formatea la petición en un mensaje y por su parte el servidor contesta en ese medio. Al llegar los mensajes, tanto a clientes como a servidores, se administran por una cola, con esquemas FIFO (First In First Out) o LIFO (Last In First Out). Como ejemplos de productos que utilizan este modelo están CICS, Presentation Manager y Windows de Microsoft.

Llamadas de procedimiento remoto (Remote Procedure Calls (RPC's)).

Un RPC permite a una computadora ejecutar un proceso sobre otro sistema en una máquina remota. El procedimiento sabe qué hacer cuando llega un mensaje de ejecución, las tareas a ejecutar y la respuesta que debe dar.

Para hacer realidad los RPC's se utilizan un lenguaje, un compilador y un runtime. Para crear uno, se escribe su contenido en código fuente, se compila y el runtime envía los datos por la red.

1.3.9 Sistemas manejadores de bases de datos.

Entre la base de datos física y los usuarios del sistema existe un nivel de programas conocido como el Sistema Manejador de Bases de Datos, o por sus siglas en Inglés DBMS (Data Base Management System). Entre sus componentes están las utilerías, las herramientas para desarrollar aplicaciones y los generadores de información, principalmente.

El DBMS maneja todo el acceso a la base de datos. Las acciones que ejecuta para ello son:

1. Un usuario solicita acceso por medio de algún sublenguaje de datos.

2. El DBMS inspecciona el esquema externo de ese usuario, la correspondencia externa/conceptual,

el esquema conceptual, la correspondencia conceptual interna y la definición de la estructura de almacenamiento.

3.- Finalmente, el DBMS ejecuta las operaciones necesarias sobre la base de datos física.

Dentro de las funciones de un DBMS se tienen:

a) Conversión de versión fuente a objeto.

Para mejorar el desempeño de las bases de datos se suele compilar las solicitudes de acceso, de modo que no se realicen al momento de ejecución. Por lo tanto, debe aceptar las definiciones de datos en versión fuente y convertirlas a la versión objeto apropiada. Se debe contar con procesadores para los diversos lenguajes de definición de datos.

b) Manipulación de datos.

Debe poseer un lenguaje de manipulación de datos que haga posible incorporar, eliminar o actualizar datos.

En las aplicaciones reales, existen solicitudes planeadas y no planeadas. Las primeras tienen la característica de que su necesidad se previó tiempo antes de que tuvieran que ejecutarse, para lo cual se tomaron las medidas pertinentes de funcionamiento. Las solicitudes no planeadas surgen de improviso y pueden afectar el desempeño global del sistema.

c) Seguridad e integridad de los datos.

El DBMS debe hacer que se cumplan las medidas de seguridad e integridad definidas por el administrador de la base de datos.

d) Recuperación y concurrencia.

Se deben cumplir los controles de recuperación y concurrencia establecidos.

e) Diccionario de datos.

El DBMS debe incluir un diccionario de datos, el cual tendrá información acerca de los propios datos. En él se almacenan los diversos esquemas y correspondencias, relaciones programas-datos y referencias usuarios-reportes entre otras.

f) Desempeño.

El DBMS debe procurar que se opere de la forma más eficiente posible.

1.4 Ingeniería de software

1.4.1 Historia.

Como toda disciplina humana, a medida que transcurren los años y se acumula experiencia y conocimiento, la tendencia es hacia la realización del trabajo de una forma menos artesanal. El desarrollo de software no es la excepción, y precisamente la ingeniería de software tiene por objeto la construcción de grandes y complejos sistemas de información de una forma rentable.

Para que un proyecto de desarrollo de sistemas tenga éxito se requieren técnicas formales de especificación y diseño, documentar cada etapa del mismo y una eficiente administración.

El concepto ingeniería de software surgió a finales de los sesentas en una conferencia para analizar la crisis del software. Esta crisis apareció como consecuencia del surgimiento de la tercera generación de computadoras, las cuales permitían el desarrollo de grandes sistemas de software.

Al desarrollar esos grandes sistemas se observaron retrasos en su construcción, incremento desmedido en su costo, complicado mantenimiento y rendimiento pobre, sumado a esto, los costos de hardware bajaban mientras que los de software crecían. Todo esto propició la creación de metodologías para el desarrollo de importantes sistemas de cómputo, naciendo así la ingeniería de software.

1.4.2 Introducción al ciclo de vida del software.

Se denomina ciclo de vida del software al conjunto de etapas por las que atraviesa un sistema desde su desarrollo hasta su operación. Las grandes fases en que se divide el ciclo son:

1.- Análisis y definición de necesidades. De manera conjunta con los usuarios del sistema deben fijarse los objetivos, restricciones y servicios que se deben ofrecer.

2.- Diseño del sistema y del software. El diseño se divide en sistemas de software y de hardware. "El diseño de software es el proceso de representar las funciones de cada sistema a fin de poderlo transformar con facilidad en uno o más programas de computación." [SOM88]

3.- Codificación y pruebas de unidades. Esta etapa comprende la creación de programas en algún lenguaje de programación, y el proceso de verificación de que cada unidad cumpla con su especificación.

4.- Pruebas del sistema. Los programas individuales se integran y se prueban como un sistema completo como medida de aseguramiento de que se cumplen o se superan las necesidades del software.

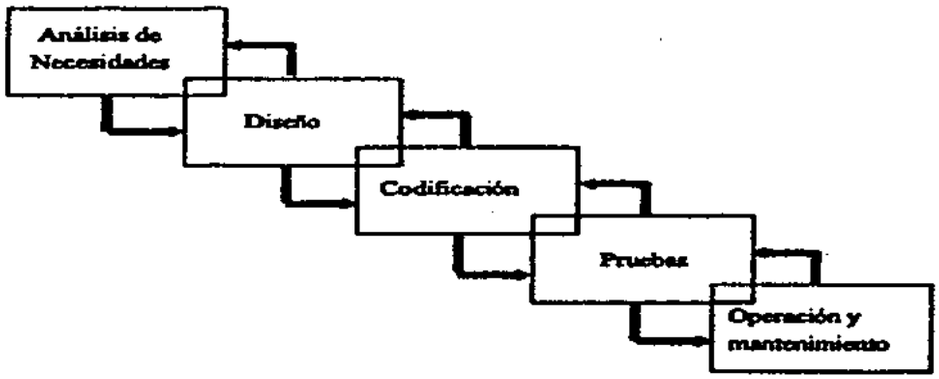


Fig. 1.4.2.1 El ciclo de vida del software.

5.- Operación y mantenimiento. Es la fase final y la más larga del ciclo de vida del software. La operación consiste en instalar y poner en uso el sistema desarrollado, y en el mantenimiento se corrigen fallas que no se detectaron en las etapas predecesoras, se mejoran las unidades del sistema y se incrementan los servicios que éste ofrece a los usuarios.

De acuerdo a análisis estadísticos realizados hasta el momento, los costos de desarrollo del software son mayores al principio y al final del ciclo de vida. Lo anterior da la pauta a pensar que una reducción de costos totales se logra mediante un diseño efectivo y unas pruebas bien planeadas y exhaustivas. La fase de mantenimiento puede traer consigo elevados costos que en algunos casos no se derivan de errores del sistema si no de cambios en las necesidades de los usuarios. Para disminuir estos costos resulta prioritario establecer con mayor precisión las necesidades a resolver por el problema.

1.4.3 Evolución del software.

El software es un producto en constante cambio. En la medida en que el ambiente donde el software se utiliza cambia, el sistema deberá irse adaptando, o de lo contrario desecharse. Este proceso de cambio se denomina "evolución del software".

Al proceso de corregir errores del sistema y de reflejar los cambios que el ambiente presenta se le conoce como mantenimiento del software. Desde la perspectiva de Lehman, existen cinco leyes de la evolución de los programas:
1.- Cambio continuo. Un programa que naturalmente se usa en el mundo real deberá ir cambiando con éste o se le condenará a ser menos útil en ese ambiente.

2.- Complejidad creciente. A medida que un programa en evolución cambia, su estructura se hace cada vez más compleja, a menos que se lleven a cabo esfuerzos para evitar este fenómeno.

3.- Evolución del programa. La evolución del programa es un proceso autorregulador, y mediante técnicas estadísticas es posible estimar características invariantes, v.gr. el tiempo entre versiones, el número de errores o el tamaño de los archivos ejecutables.

4.- Conservación de la estabilidad organizativa. Durante el tiempo de vida de un programa, su rapidez de desarrollo es casi constante e independiente de los recursos dedicados al desarrollo del sistema.

5.- Conservación de la familiaridad. Durante el tiempo de vida de un sistema la evolución del cambio en cada versión es aproximadamente constante.

Desde luego estas leyes no tienen un carácter universal, pero son aplicables a muchos desarrollos.

Dentro de las implicaciones de las leyes de Lehman en el ciclo de vida del software están:

Los costos de mantenimiento del software no se pueden eliminar, a lo más que se puede aspirar es a la adopción de técnicas que permitan la fácil incorporación de los cambios, sin repercutir en una afectación dramática en la estructura del sistema.

Otro punto es que no se deben planear modificaciones muy grandes en un solo incremento. Es preferible hacerlo en pequeñas versiones.

La tercera conclusión es que, la manera más rentable de desarrollar un software es utilizar la menor cantidad de personas posibles en los equipos de trabajo, considerando que mientras más gente trabaje, menos productivo será cada miembro del proyecto.

Es posible observar que la mayoría de los puntos abordados van encaminados al incremento de la rentabilidad. Esto es muy claro si recordamos la función esencial de la ingeniería de software: "Realizar grandes sistemas de una forma rentable".

1.4..4 Confiabilidad del software.

La característica más importante de los sistemas de software una vez que éstos están en uso es su confiabilidad. La confiabilidad de cualquier sistema depende de lo correcto de su diseño, lo adecuado de su correspondencia con la aplicación y la confiabilidad de sus componentes.

En los sistemas de software toda la confiabilidad depende de lo correcto de su diseño y de su aplicación. Someramente se acepta que la confiabilidad del software se da si se cumple con las especificaciones iniciales y se comporta según las expectativas.

1.4.5 El ciclo de vida del software.

1.4.5.1 Definición de requisitos.

Como toda obra de ingeniería, el software requiere de especificaciones claras y precisas. El análisis y la definición de requisitos es la primer etapa del ciclo de vida, y en las palabras de Sommerville es: "El proceso de establecer los servicios que debe proporcionar el sistema y las restricciones con las cuales debe operar." [SOM88]

Es necesario analizar la información acerca del problema que se va a resolver y generar una definición. La descripción precisa de los requisitos de un sistema se plantea en el documento de los requisitos del software. En él deben quedar asentadas todas las características que debe satisfacer el sistema de software.

Las características que debe cumplir el documento de requisitos son:

a) Completo. Es necesario especificar todo lo que ha de hacer el sistema.

b) Consistente. Ningún requisito debe entrar en conflicto con los demás.

Este documento tiene valor tanto para los desarrolladores como para los programadores de mantenimiento, dado que estos últimos lo explotarán para saber lo que el sistema debe hacer.

Los puntos que debe contener este documento son:

1.- Introducción. Debe ubicar al sistema en contexto y describir las funciones que éste realizará.

2.- Hardware. Se deben describir las características de hardware donde el sistema se ejecutará, especificando las configuraciones mínima y óptima.

3.- Modelo conceptual. Es una visión de alto nivel del sistema donde se muestran los principales servicios que proporcionará el software. Una técnica útil para esta parte son los diagramas de flujo de datos descritos por Yourdon. En esta técnica se hacen representaciones gráficas de los servicios que el sistema ofrece y puede irse detallando hasta donde sea necesario.

4.- Requisitos funcionales. Aquí se plantean los servicios que el usuario espera del sistema. Para su especificación se puede usar el lenguaje natural o estructurado.

5.- Requisitos de la base de datos. En esta parte debe quedar planteada la definición lógica de la base de datos (descrita en el punto 1.3 del presente trabajo).

6.- Requisitos no funcionales. Estos son las restricciones u obligaciones bajo las cuales operará el software. V.gr. tiempos de respuesta, limitaciones de memoria y velocidad de tranmisión. Este punto es especialmente vulnerable a los cambios en el hardware en donde se alojará el sistema.

7.- Información para mantenimiento. Se debe aclarar las suposiciones tecnológicas sobre las cuales se construirá el sistema, y los cambios esperados debidos a la evolución del hardware o a las necesidades del usuario.

Una vez establecidos los puntos anteriores, es de suma importancia su confirmación. Un visto bueno oportuno constituye un factor crítico de éxito. Durante la confirmación debe buscarse que los requisitos sean consistentes, completos y realistas. Esta labor debe realizarse entre los usuarios y los ingenieros de software.

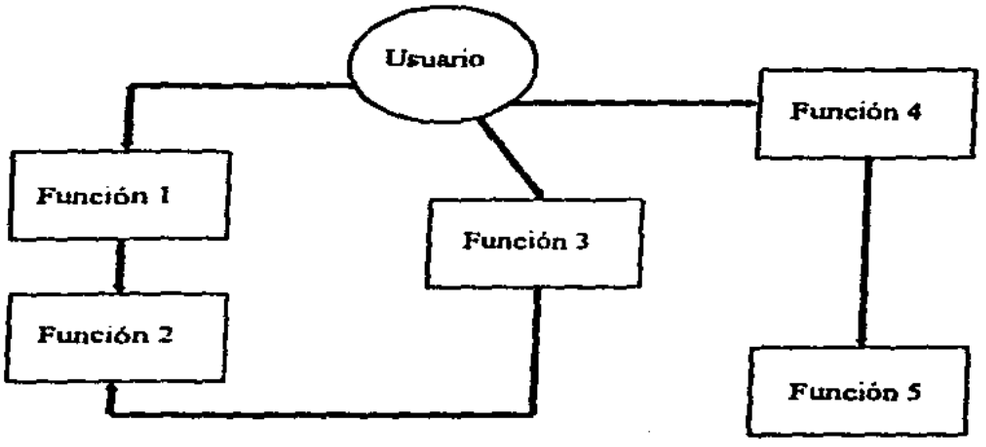


Fig. 1.4.5.1.1 Modelo conceptual.

1.4.5.2 Diseño.

Especificación de requisitos.

En esta fase se analiza la definición de requisitos y se diseñan los componentes del software para proporcionar los servicios que el usuario solicitó, de tal suerte que con la información acumulada se puedan crear esos componentes en algún lenguaje de programación.

El diseño se divide en tres etapas:

1.- Asociar componentes de software con los servicios establecidos en la definición de requisitos, y construir especificaciones precisas para esos componentes.

2.- Construir un diseño de alto nivel que muestre la relación recíproca de los componentes abstractos de software.

3.- Crear un diseño detallado para cada componente abstracto.

Especificación precisa de los componentes de software.

Como el documento de especificación precisa de los componentes de software contiene definiciones abstractas, está destinado al diseñador de software, no al usuario. Es recomendable que esta etapa preceda al diseño, dado que aquí se clarifica lo que debe hacer el software para posteriormente en el diseño plasmar la forma de realizar esas funciones.

La especificación debe realizarse en una notación preferentemente matemática, formalizando la sintaxis y la semántica del lenguaje de especificación.

Algunas de las ventajas de una especificación formal son:

a) Puede demostrarse que un programa cumple con las expectativas, con base en su especificación y la definición de la semántica del lenguaje.

b) Al basarse en notaciones matemáticas, cabe la posibilidad de especificaciones equivalentes.

Existen tres tipos de especificaciones a saber:

1.- De interfaces. Especifica las restricciones de entrada y salida de cada componente de software.

2.- Operacionales. Define el cálculo que realiza la transformación de la entrada en salida, expresada de una forma abstracta de alto nivel.

3.- De abstracciones de datos. Define el significado de los tipos de datos mediante la especificación del comportamiento de las operaciones de los tipos. Aquí se utiliza un enfoque algebraico donde se tiene una parte de interfaz que nombra las operaciones y los tipos de parámetros, y una parte de axiomas que define el comportamiento de esas operaciones.

La especificación de un sistema se divide en un proceso de tres etapas:

1.- Especificaciones operacionales para las abstracciones de más alto nivel, explicitadas en función de abstracciones más simples y tipos abstractos.

2.- Creación de especificaciones para los tipos de datos abstractos.

3.- Construcción de especificaciones de interfaces para las abstracciones simples.

Diseño detallado del software

La definición de requisitos se debe utilizar para crear el diseño del sistema de software que cumpla con los mismos.

Esta es la parte más importante del proceso de desarrollo de software y se afirma que un buen diseño es la clave de una ingeniería de software efectiva. Cuando se logra un diseño confiable es más fácil programarlo y darle mantenimiento. Una especificación precisa es una parte vital del proceso de diseño.

El proceso de diseño sigue en términos generales las siguientes etapas:

1.- Se deben establecer los subsistemas que conforman el sistema de software.

2.- Cada subsistema debe dividirse en componentes individuales definiéndose la operación de esos componentes.

3.- Cada programa se debe diseñar con subcomponentes que interactúen entre sí.

4.- Es necesario refinar cada componente, lo que conduce a la especificación de componentes como una agrupación de subcomponentes.

5.- Se deben precisar los algoritmos a utilizar en cada componente.

6.- Se deben crear las estructuras de archivos y de datos a existir en el sistema.

7.- Se debe realizar la matriz de pruebas para los programas.

Se acepta que un buen diseño se caracteriza por ser de fácil mantenimiento. Esto implica una reducción de costos de cambios al sistema debido a que las modificaciones tienen un efecto local.

Un diseño de software debe ser muy coherente y poco acoplado. Una unidad de programa es coherente si los elementos muestran un alto grado de relación funcional, es decir, cada elemento de la unidad de programa debe ser esencial para que la unidad completa cumpla su propósito. El poco acoplamiento se refiere a que las unidades de programa no deben depender una de la otra, lo ideal es tener unidades independientes.

La ventaja principal de la coherencia y el acoplamiento es que cualquier unidad puede ser reemplazada por una equivalente sin cambiar las otras unidades, tanto a nivel de diseño como de programación.

1.4.5.2.1 Metodologías de diseño.

Existen tres fundamentalmente:

1.- Diseño funcional descendente. El sistema se diseña partiendo de una visión de alto nivel que se va bajando hasta llegar a un diseño detallado, todo desde una óptica funcional. Esta metodología se conoce además como estructurada y un autor importante en esta área es Edward Yourdon.

2.- Diseño orientado a objetos. El sistema se ve como una colección de objetos que van pasando mensajes entre ellos. Cada objeto tiene un conjunto de operaciones asociadas, y se basa en el ocultamiento de información. Esta técnica fue propuesta por primera vez por Parnas en 1972.

3.- Diseño controlado por los datos. Metodología propuesta por Jackson en 1975, contempla que la estructura de un sistema debe reflejar la estructura de datos que éste procese. El diseño se obtiene de un análisis de los datos de entrada y salida.

Debido a que no hay una metodología aceptada universalmente como buena, se pueden utilizar una combinación de las mismas dependiendo de las características del software a realizar.

1.4.5.2.2 Herramientas gráficas de apoyo al análisis.

Se tienen tres herramientas principales:

1.- Diagramas de flujo de datos: Muestran como se transforman los datos al pasar de un componente del sistema a otro. Es de utilidad en los diseños de alto nivel. Se conforman de tres elementos: burbujas que representan centros de transformación, flechas que indican el flujo de los datos hacia adentro y afuera de los centros de transformación, y operadores que unen a las flechas.

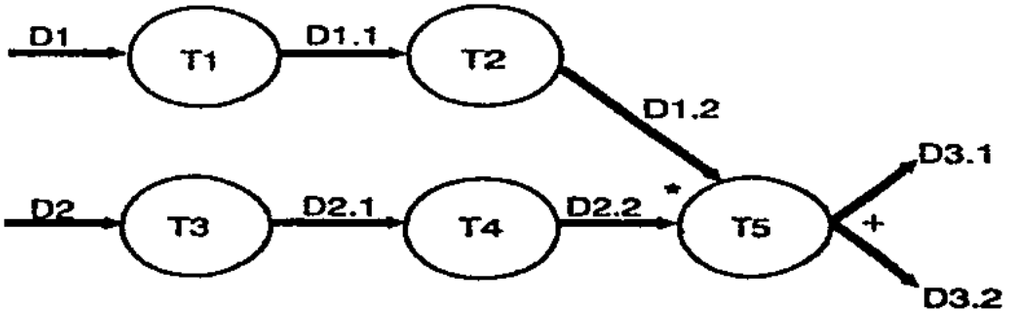


Fig. 1.4.5.2.2.1 Diagrama de flujo de datos.

2.- Diagramas de estructura. Describen al sistema de programación como una jerarquía de partes y se representan gráficamente como un árbol. También sirve para los diseños de alto nivel. En estos diagramas se muestra las relaciones entre las unidades de programa usando tres elementos:

a) Un triángulo con el nombre de la unidad.
b) Una flecha que conecta los triángulos.
c) Una flecha con un círculo con el nombre de los datos que se pasan entre los elementos.

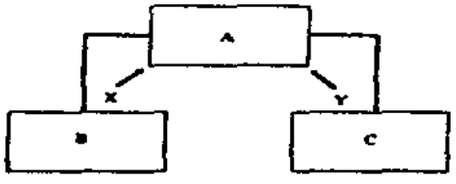


Fig. 1.4.5.2.2.2 Diagrama de estructura.

3.- Lenguaje para descripción del diseño. Se usa para el diseño detallado utilizando un lenguaje formal de programación, v.gr. Pascal o C. Existen lenguajes exprofeso para descripción del diseño que utilizan las estructuras de control de los lenguajes de alto nivel.

Diseño descendente

Se basa en el hecho de que la estructura del problema debe determinar la estructura de la solución de software. Utiliza una premisa esencial del pensamiento humano: la abstracción.

Las etapas que comprende el diseño descendente son:

1.- Estudiar y comprender el problema.

2.- Identificar las características generales de las posibles soluciones y elegir la más simple.

3.- Construir un diagrama de flujo de datos.

4.- Basándose en el DFD, construir un diagrama de estructura que muestre las unidades de programa relacionadas con la solución.

5.- Describir cada abstracción mediante un lenguaje de descripción.

Una vez que se formule una solución de alto nivel, el proceso de solución debe repetirse hasta llegar a una especificación de bajo nivel.

Transformación de diagramas de flujo de datos a diagramas de estructura.

El primer paso es identificar las unidades de entrada y salida de más alto nivel, rastreando las entradas hasta encontrar una burbuja cuya salida sea tal que su entrada no se pueda deducir del examen de la salida. Para la burbuja de salida de más alto nivel se sigue un criterio análogo. Las transformaciones que no son de entrada o de salida se denominan centrales.

El primer nivel del diagrama de estructura se forma con la unidad de entrada y las transformaciones centrales. La caja en la raíz del diagrama se designa como unidad de control.

1.4.5.3 Pruebas.

La prueba de programas es la técnica de confirmación de sistemas más efectiva, y aunque existe una etapa especial para ello, se da también durante la construcción." La prueba consiste en ejercitar el programa utilizando datos similares a los datos reales que habrán de ser ejecutados por el programa, observar los resultados y deducir la existencia de errores o insuficiencias del programa a partir de las anomalías de ese resultado." [SOM88] Me es posible afirmar que las pruebas y la depuración constituyen un proceso de control de calidad del software.

Es preciso distinguir entre prueba y depuración, la primera es establecer la existencia de errores en el programa y depuración se refiere a localizar donde se produjeron esos errores y corregir el código o los datos que los ocasionen.

Las pruebas deben diseñarse pensando que el comportamiento de un programa sea distinto del que se pretendía en el diseño o la construcción. El personal adecuado para conformar el equipo de pruebas debe estar constituido por las personas que participaron en la construcción y recursos exclusivos para las pruebas.

1.4.5.3.1 Proceso de prueba.

Los autores coinciden en que se identifican cinco etapas durante las pruebas:

1.- Prueba de funciones. Es el nivel básico donde se prueban las funciones de un módulo para garantizar su correcta operación. Regularmente esta prueba se realiza por los mismos programadores durante la construcción del sistema.

2.- Prueba de módulos. Una vez que se prueban las funciones independientemente es necesario examinar su interacción, donde lo deseable es verificar módulos aislados.

3.- Prueba de subsistemas. Aquí se hace necesario probar la agrupación de módulos cooperando entre sí.

4.- Prueba del sistema o prueba de integración. Esta se lleva a cabo cuando se integran los subsistemas para conformar el sistema completo. El objetivo aquí es corroborar que todo el sistema cumple con lo especificado en la definición de requisitos.

5.- Prueba de aceptación. En esta fase se debe probar todo el sistema con datos reales, donde lo correcto es que el sistema cumpla la funcionalidad y rendimiento previstos.

La fase de pruebas se debe planear al término del diseño con la intención de tener todo el tiempo necesario para ajustar el plan, e ir probando lo que esté concluido a medida que se va terminando la programación.

1.4.5.3.2 Estilos de pruebas.

Existen dos estilos, pruebas descendentes y ascendentes. En ambos existen varios elementos a saber: funciones, módulos, subsistemas y el sistema completo.

La prueba descendente empieza al nivel de los subsistemas, representando a los módulos con creaciones más simples que imitan la funcionalidad real. Después de probar los subsistemas se realiza la misma labor con los módulos y finalmente se reemplazan las funciones con código real. La ventaja de este estilo es que los errores de diseño se pueden detectar tempranamente, puesto que se va de lo general a lo particular.

La prueba ascendente es el proceso inverso. Se inicia probando las funciones, posteriormente se integran para formar un módulo y se prueba. Al terminar con todos los módulos se integran y se verifica el subsistema. Como se avanza de menos a más, es probable que se encuentren errores de diseño o programación demasiado tarde.

1.4.5.3.3 Diseño de casos de prueba.

Es necesario construir una matriz de pruebas que sea lo suficientemente amplia para contemplar todas las funciones del sistema, considerar entradas válidas e inválidas, tomar en cuenta que el sistema se comporte de acuerdo a las especificaciones y que no afecte datos o funciones de otros sistemas con los cuales interactúe. Lo más importante es considerar el conjunto mínimo de casos de prueba requeridos para garantizar un nivel de confianza razonable.

En ocasiones es necesario probar sistemas de tiempo real, en los cuales resulta vital que el sistema responda con la oportunidad demandada.

1.4.5.3.4 Verificación de programas.

Esta actividad es para demostrar con métodos matemáticos la correspondencia entre un programa y sus especificaciones.

1.4.5.3.5 Inspección del código.

En algunos proyectos de ingeniería de software resulta enriquecedor formar un equipo especializado encargado de revisar la calidad del código, y en su caso recomendar posibles mejoras que consecuentarán en un mejor desempeño del sistema.

1.4.5.3.6 Depuración de programas.

La depuración de programas consiste en identificar las partes de los programas que causan errores y modificarlas para su corrección. Este proceso implica por una parte localizar las partes incorrectas del código, de las bases de datos o de cualquier parte del sistema, y por otra efectuar dichas correcciones y volver a probar. Esta etapa es el complemento de las pruebas.

La depuración puede tener dos vertientes, corregir errores de codificación o variantes al diseño. Los cambios al diseño resultan muy costosos y se deben evitar por medio de revisiones a lo largo del ciclo de vida.

Cuando se rediseñan algunas partes del sistema se debe analizar el impacto que tendrá en otros lados, y de ser necesario se tienen que modificar.

1.4.5.4 Documentación.

Los sistemas creados deben ser interpretados y modificados no solo por sus creadores, si no por cualquier persona capacitada que consultando la documentación, pueda tener elementos para realizar un cambio o corrección.

La documentación se puede definir como un escrito donde se plantea la manera de utilizar los programas, las técnicas utilizadas en su construcción y el objetivo que obedece su creación.

El mantenimiento se refiere a las modificaciones que se hacen una vez que el sistema se libera. Esto requiere un profundo entendimiento de los programas y de todo el material contenido en la documentación.

Se clasifica como documentación de usuario y de sistema. La primera son los documentos relacionados con las funciones del sistema, y la segunda describe todos los aspectos relacionados con el diseño, construcción, pruebas e implantación del software.

1.4.5.4.1 Documentación de usuario.

Debe proporcionar una perspectiva amplia y precisa acerca del sistema, y según Sommerville, la deben conformar al menos cinco documentos:

1.- Una descripción funcional del sistema. Debe contener en términos generales las características del sistema.

2.- Guía de instalación. Describe la manera de instalar el sistema y la configuración requerida en un ambiente dado.

3.- Manual introductorio. Se necesita que explique cómo iniciar el sistema, el uso de funciones comunes y cómo salir de problemas frecuentes.

4.- Manual de referencia. Debe detallar las funciones y ventajas de utilizar el sistema. Es el documento completo sobre el uso del sistema.

5.- Guía del operador (en caso de que exista).

En nuestros días los manuales electrónicos o ayuda en línea están sustituyendo a los tradicionales manuales impresos. Por otro lado, en algunas organizaciones se tienen áreas de soporte en cómputo que llevan a cabo la instalación del software, por lo que esta guía pasa a la documentación del sistema.

1.4.5.4.2 Documentación de sistema.

Comprende desde la especificación de requisitos hasta los documentos de liberación a producción del sistema. Se acepta que la documentación de sistema esté integrada por:

1.- Definición de requisitos.

2.- Especificación general de la descomposición de los requisitos en programas.

3.- Descripción de la división de los programas en componentes y sus características.

4.- Descripción de la operación de cada unidad. Resaltando que las acciones del programa se documentan a manera de comentarios dentro del código.

5.- Matriz de pruebas que describa cómo se debe probar cada unidad de programa.

6.- Matriz de pruebas de integración de todas las unidades.

7.- Plan de liberación a producción.

Es recomendable que dentro de las organizaciones existan estándares tanto para la documentación de usuario como de sistema. Así mismo, al ritmo que se van modificando los sistemas o migrando de ambientes es necesario ir actualizando la documentación.

1.4.5 Mantenimiento.

Se conoce con este nombre a las actividades que involucran modificaciones a los sistemas, comprenden la corrección de errores de programación, cambios por errores de diseño, modificaciones drásticas debido a errores de especificación o inclusión de nuevos requerimientos.

El mantenimiento tiene tres clasificaciones:

1.- De perfeccionamiento. Cambios solicitados por el usuario u oportunidades de mejora detectadas por el programador.

2.- Adaptativo. Se deriva de cambios en el ambiente del programa.

3.- Correctivo. Corrección de errores no detectados durante las pruebas.

Estudios acerca de la evolución del software reflejan que los mantenimientos eran 65% de perfeccionamiento, 18 % adaptativos y 17 % correctivos. Así mismo, se afirma que alrededor del 50 % de la programación dentro de las organizaciones se debe a mantenimientos de sistemas existentes.

Como mencioné en capítulos anteriores, toda obra de ingeniería de software requiere de mantenimiento debido a que en sí mismo el sistema modifica el medio donde se usa, o éste puede variar debido a factores externos.

Existen múltiples razones por las que un sistema requiera mantenimiento, como la aparición de nuevas tecnologías de hardware, creación de nuevos lenguajes de programación o alteración de factores económicos.

1.4.6 Análisis y diseño estructurado.

1.4.6.1 Análisis.

El modelo esencial.

"Es un modelo de lo que el sistema debe hacer para satisfacer los requerimientos del usuario, diciendo lo mínimo posible acerca de cómo se implantará. " [YOU89]. Esto conduce a suponer que se tiene una tecnología perfecta y disponible para su uso.

El modelo esencial se compone de dos elementos:

1.- El modelo ambiental.
2.- El modelo de comportamiento.

1.4.6.1.1 El modelo ambiental.

El modelo ambiental plasma la frontera entre el sistema y el exterior. La idea es determinar qué está en el interior del sistema y qué no. En términos de sistemas de información lo que se pretende es saber qué entra al sistema desde el ambiente exterior y qué información se debe producir como salida al ambiente externo.

El modelo ambiental está integrado por:

Declaración de propósitos.
Diagrama de contexto.
Lista de acontecimientos.

1.4.6.1.1.1 Declaración de propósitos.

Es una declaración breve del propósito del sistema, está dirigida a los niveles superiores de la organización. Se recomienda que esta declaración no extrapole un párrafo, aun cuando queden algunas interrogantes al respecto, mismas que se resolverán con los siguientes elementos del modelo esencial.

1.4.6.1.1.2 Diagrama de contexto.

Es un caso especial del diagrama de flujo de datos (DFD) en el cual una sola burbuja representa todo el sistema y su relación con entidades externas. Los elementos que debe contener el diagrama son:

a) Los terminadores con los que se comunica el sistema, abarcando sistemas organizaciones o personas.

b) Los datos recibidos del exterior.

c) Los datos que el sistema produce para el mundo exterior.

d) Los almacenes de datos que el sistema comparte con los terminadores.

e) La frontera entre el sistema y el ambiente.

La parte central del diagrama de contexto es una burbuja que contiene el nombre de todo el sistema.

Los terminadores se representan por rectángulos que se comunican con el sistema a través de flujos de datos, de control o de almacenes externos. Como convención los terminadores no se comunican entre sí.

Los flujos en el diagrama de contexto representan datos que entran y salen del sistema y señales de control. Otro uso de los flujos es para representar la transportación de datos entre terminadores por medio del sistema.

1.4.6.1.1.3 Lista de acontecimientos.

Son una serie de eventos provenientes del exterior a los cuales el sistema debe responder. Se clasifican de tipo flujo, temporal y de control.

En algunos casos se recomienda se construya como parte del modelo ambiental un diccionario de datos inicial y un modelo entidad/relación de los almacenes externos. Lo anterior es conveniente sobre todo cuando el medio en que se encuentra el sistema es muy cambiante.

1.4.6.1.2 El modelo de comportamiento.

El modelo de comportamiento describe lo que se demanda del sistema para que interactúe con el ambiente. Las partes que conforman este modelo son:

a) Diagramas de flujo de datos.

b) Diagramas de entidad/relación.

c) Diagramas de transición de estados.

d) Diccionario de datos.

e) Especificaciones de procesos.

Enfoque de partición por acontecimientos.

Contempla los siguientes pasos:

1.- Se dibuja una burbuja para cada acontecimiento de la lista.

2.- La burbuja se nombra con la respuesta que el sistema debe dar al acontecimiento.

3.- Se dibujan las entradas y salidas, y se dibujan los almacenes para la comunicación entre burbujas.

4.- Se compara el borrador con el modelo ambiental para asegurar consistencia.

1.4.6.1.2.1 Diagrama de flujo de datos.

Un DFD permite visualizar al sistema como una red de procesos funcionales, conectados entre sí por conductos y tanques de almacenamiento de datos.

Los componentes de un DFD son:

a) El proceso es una parte del sistema que transforma entradas en salidas, se representa gráficamente por un círculo y tiene un nombre corto de una frase.

b) El flujo se utiliza para describir el movimiento de datos de una parte del sistema a otra. Se identifica gráficamente como una flecha que entra o sale de un proceso. Los flujos son datos en movimiento y los almacenes datos en reposo.

c) El almacén representa un conjunto de datos en reposo. Gráficamente son dos líneas paralelas con el nombre del almacén en el centro escrito en plural.

d) El terminador representa entidades externas con las cuales el sistema se comunica. La figura con la que se representa es un rectángulo.

Es recomendable construir DFD no muy complejos que sea difícil su interpretación. Así mismo, el proceso de construcción de DFD es iterativo al grado que sea necesario y lógicamente consistente.

1.4.6.1.2.2 Diagrama de entidad/relación.

Es un modelo de red que describe la distribución de datos almacenados en un sistema. Sus cuatro componentes son:

a) Tipos de objetos. Se representa por un rectángulo y es una colección de objetos del mundo real.

b) Relaciones. Los objetos se conectan entre sí por medio de relaciones, que son un conjunto de conexiones entre objetos representados por un rombo. Se da el caso de que existan múltiples relaciones entre objetos.

c) Indicadores asociativos de tipo de objeto. Representa algo que funciona como objeto y como relación.

d) Indicadores de subtipo y supertipo. Son tipos de objeto de una o más subcategorías.

Análogamente al DFD la construcción de un DER es un proceso que requiere múltiple refinamiento que considera ampliaciones y reducciones a las entidades presentes o modificación a la definición de las mismas.

1.4.6.1.2.3 Diagramas de transición de estados.

Muestra el comportamiento en el tiempo del sistema. Sus elementos que lo conforman son:

a) Estados. Identificados como rectángulos, representa el estado en que se puede encontrar un sistema en un momento dado.

b) Cambios de estado. Se muestran como flechas y plasman los cambios de estado que se pueden dar y el orden en que sucede.

c) Condiciones y acciones. Muestran la condiciones que causan un cambio de estado y las acciones que el sistema efectúa cuando se cambia de estado. Gráficamente es una línea horizontal que contiene la condición en la parte superior y la acción en la inferior.

La manera de construirse que se recomienda es empezar por identificar todos los estados y proseguir con la conexión de los mismos. Nuevamente es un ejercicio iterativo.

1.4.6.1.2.4 Diccionario de datos.

Es un listado de todos los datos, con definiciones precisas para poder entender entradas, salidas, componentes de almacenes y cálculos intermedios. Para considerarse completo un diccionario de datos debe contener:

a) Significado de flujos y almacenes del DFD.

b) Especificar valores y unidades de la información contenida en los flujos y almacenes.

c) Describir las relaciones entre almacenes del modelo E/R.

Notación del diccionario de datos:

= está compuesto de	** comentario
+ y	@ campo llave de un almacén
( ) optativo	| separa opciones alternativas en la construcción
{ } iteración	[ ] seleccionar de varias alternativas

1.4.6.1.2.5 Especificaciones de proceso.

Es lo que debe hacerse para transformar entradas en salidas, también conocidas como miniespecificaciones. Es lo que sucede en las burbujas de nivel más bajo de un DFD.

Los dos lineamientos sobre los que debe girar la especificación de procesos son:

a) Debe expresarse de una forma sencilla para que la comprendan usuario y analista.

b) Debe especificarse de forma que pueda ser comunicada a cualquier auditorio ajeno al proyecto.

Las herramientas más comúnmente utilizadas para escribir especificaciones son: lenguaje estructurado, tablas de decisiones, pre/post condiciones, diagramas de flujo y diagramas de Nassi/Shneiderman. La más recomendable es la de lenguaje estructurado por su sencillez y consistencia.

1.4.6.2 Diseño.

La labor de diseño contempla el desarrollo de los modelos de implantación de sistemas y el modelo de implantación de programas.

Modelo de implantación de sistemas.

Este modelo se divide en un modelo del procesador y uno de tareas.

1.4.6.2.1 Modelo del procesador.

El objetivo de este modelo es decidir la asignación del modelo esencial a los procesadores con que se cuente y el esquema de comunicación que debe seguirse. Las opciones que existen son asignar todo el modelo a un solo procesador (computadora principal), asignar ciertas burbujas del DFD a procesadores distintos en mini y microcomputadoras (solución distribuida) y una combinación de ambas. De la misma forma, es preciso asignar los almacenes de datos a los procesadores.

Las recomendaciones generales que deben considerarse, se centran en:

a) Costo. Escoger el esquema de asignación que mejor resulte en el análisis beneficio/costo.

b) Eficiencia. Enmarcarse en las expectativas de tiempo de respuesta de los clientes.

c) Seguridad. La organización por lo regular cuenta con un esquema de protección de datos el cual debe considerarse o en caso de que no exista, darle alta protección a la información de los procesos clave del negocio.

d) Confiabilidad. Existen medidas de los requerimientos de servicio de un sistema especificados por:

MTBF (medium time between faults): tiempo promedio entre fallas.

MTTR (medium time to reparation): tiempo promedio de reparación

La disponibilidad de un sistema es el porcentaje de tiempo que está disponible y se calcula como:

$$\text{Disponibilidad} = \text{MTBF} / (\text{MTBF} + \text{MTTR})$$

Dependiendo de esos requisitos se hará la asignación y si es el caso, se tendrán elementos paralelos de hardware e inclusive redundantes, que permitan mantener el nivel de servicio que se requiere.

e) Restricciones políticas y operacionales. Son aspectos dictados por cada organización que deben contemplarse, inclusive desde el modelo esencial.

1.4.6.2.2 Modelo de tareas.

En cada uno de los procesadores debe asignar procesos y almacenes a las tareas individuales de cada uno.

Modelo de implantación de programas.

En una tarea individual sólo se puede realizar una actividad a la vez, para organizar la actividad se utilizan los diagramas de estructura, en el cual se aprecian la organización jerárquica de módulos dentro de una tarea.

Calidad del diseño.

Para evaluar la calidad del diseño existen reglas que permiten ubicar un diseño:

a) Cohesión. Grado en el cual los componentes de un módulo son necesarios y suficientes para llevar a cabo una función bien definida. Lo que se busca es tener módulos funcionalmente cohesivos, donde cada instrucción es necesaria para poder llevar a cabo una tarea. Lo que se debe evitar son los

coincidentalmnte cohesivos que son aquellos cuyas instrucciones no tienen relación. En conclusión se debe evitar fragmentar procesos esenciales en módulos y no juntar aquellos que no tengan relación.

b) Acoplamiento. Grado en el cual los módulos se interconectan o se relacionan entre ellos. Lo que se busca es tener acoplamiento bajo, que permita en caso de una modificación sólo afectar un módulo y lo menos posible a los demás.

c) Tamaño del módulo. Lo ideal es que sea lo más pequeño posible, una página por ejemplo.

d) Alcance del control. Es el número de módulos subordinados que se mandan llamar, seis es el número buscado.

e) Alcance del efecto/alcance del control. Es una premisa que plantea que el módulo afectado por una decisión debe estar en la jerarquía de quien la tomó.

1.4.7 Análisis y diseño orientado a objetos.

Las metodologías de análisis y diseño orientadas a objetos son de reciente creación y despiertan un especial interés dentro del ambiente de la computación. Algunos autores las identifican como la vanguardia y la mejor técnica para el desarrollo de sistemas de información.

1.4.7.1 Análisis orientado a objetos (AOO).

El análisis orientado a objetos es equivalente, en su parte de definición y partición de un problema, al análisis de requerimientos descrito anteriormente. Un objeto puede verse como un elemento de información y una operación como un proceso que se aplica a los objetos.

La metodología de análisis consta de los siguientes elementos:

1.- El sistema se describe usando una estrategia informal, que consiste en una narración en lenguaje natural de la solución del problema a resolver.

2.- Los objetos se determinan subrayando nombres y cláusulas nominales y se van acomodando en una tabla. Puede darse el caso que el objeto esté dentro del espacio del problema o dentro de la solución.

3.- Los atributos de los objetos se identifican subrayando los adjetivos y luego se asocian con sus respectivos nombres.

4.- Las operaciones se identifican subrayando los verbos, frases verbales y predicados; y relacionando cada operación con el objeto apropiado.

5.- Los atributos de las operaciones se obtienen al subrayar los adverbios y asociándolos con sus operaciones respectivas.

1.4.7.2 Diseño orientado a objetos (DOO).

"El diseño orientado a objetos crea una representación del dominio del problema en el mundo real y lo transforma en un dominio de solución que es software. El DOO da como resultado un diseño que interconexiona los objetos de datos (elementos de datos) y las operaciones de procesamiento, de forma que modulariza la información y el procesamiento en vez de sólo el procesamiento." [PRE93]

Al usar DOO se pretende que el software diseñado tenga las características de abstracción, ocultación de la información y modularidad.

1.4.7.2.1 Historia del DOO.

Al inicio de la programación, los lenguajes ensambladores utilizaban instrucciones máquina u operadores para manipular los elementos de datos. En aquel entonces el nivel de abstracción era muy bajo.

Cuando aparecieron lenguajes de programación como FORTRAN y COBOL, los objetos y operaciones podían ser modelados mediante datos y estructuras de control predefinidas, que formaban parte del lenguaje de programación. El diseño de software se enfocaba sobre la representación del detalle procedimental usando el lenguaje de programación que correspondiera. En esta época se incorporaron conceptos como refinamientos sucesivos de una función, modularidad procedimental y programación estructurada.

En la década de los setentas se introdujeron conceptos como abstracción y ocultación de la información, y los métodos de diseño eran conducidos por los datos. Por su parte, los lenguajes tenían ya una variedad más rica de tipos y estructuras de datos.

Al aparecer lenguajes como SIMULA y Smalltalk, los problemas del mundo real se representaban mediante objetos de datos a los que se adicionaban operaciones y la abstracción de datos tenía una gran importancia. Estos lenguajes marcaron la primer diferencia con los convencionales.

En la década de los ochentas surgieron lenguajes como ADA que llevaron a un interés en el DOO. En las palabras de Abbott " El análisis de sentencias en lenguaje natural del problema y su solución, pueden usarse como guía para desarrollar la parte visible de un paquete que tenga los datos y procedimientos que operan sobre ellos, y el algoritmo particular para un problema dado".

Booch, una importante personalidad del medio de la computación, coadyuvó a la popularización de los conceptos de análisis y diseño orientado a objetos. Los primeros intentos por crear una metodología para el diseño orientado a objetos surgieron a inicios de los ochentas.

1.4.7.2.2 Conceptos de DOO.

Objetos, operaciones y mensajes.

Un objeto es un componente del mundo real que se transforma en el dominio del software. V.gr. archivos, cadenas alfanuméricas u órdenes.

Cuando un objeto se transforma en software, consta de una estructura de datos privada y procesos denominados operaciones o métodos, los cuales están facultados para transformar la estructura de datos y poseen las construcciones procedimentales.

Un mensaje es una petición al objeto para que ejecute una de sus operaciones.

Un objeto posee una parte privada y una compartida, la primera es la estructura de datos y el conjunto de operaciones para la estructura de datos. La compartida es su interfaz. Los mensajes se mueven a través de la interfaz y especifican qué operaciones del objeto se desean, sin mencionar cómo se realizarán.

Al definir un objeto con partes privadas y dando mensajes para llamar al procesamiento adecuado, se consigue el ocultamiento de información. Los objetos y sus operaciones dan modularidad inherente, esto es, los datos y procesos se agrupan junto con los mecanismos de interfaces o mensajes.

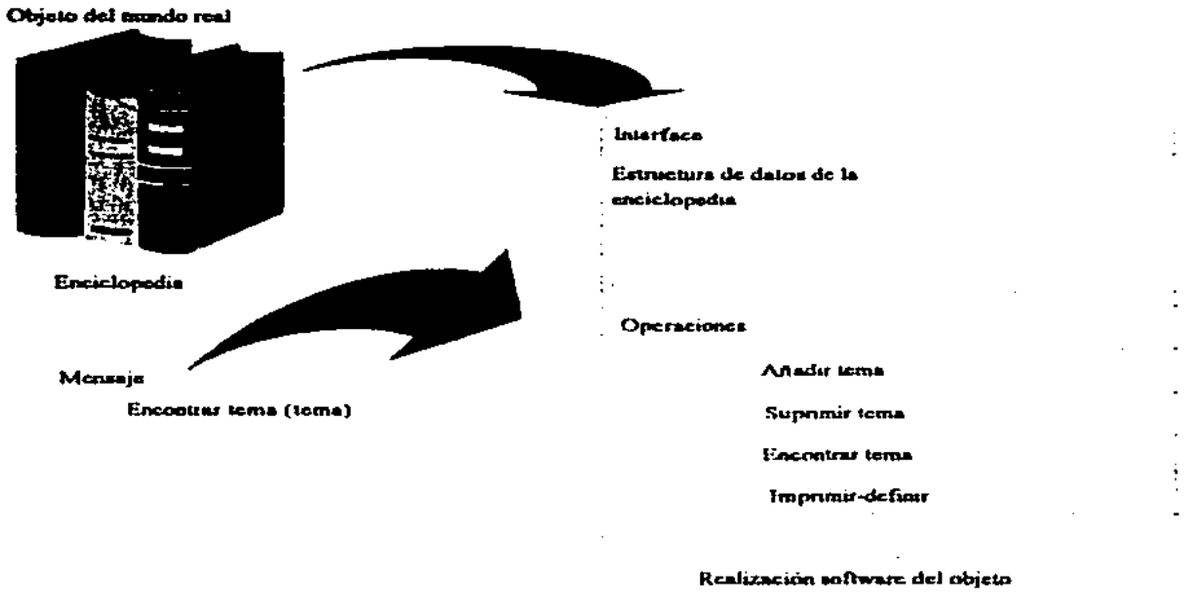


Fig. 1.4.7.2.2.1 Ejemplo de objetos en el mundo real.

Clases, instancias y herencia.

Una clase es un conjunto de objetos que tienen las mismas características, y se dice que un objeto individual es una instancia de una clase más amplia. Todos los objetos son miembros de una clase más amplia y heredan la estructura de datos privada y las operaciones que se han definido para esa clase.

Un concepto clave en la ingeniería de software es la reutilización de código. Esto se consigue creando objetos que se construyen sobre los atributos y operaciones existentes que se heredan de una clase, para sólo definir las nuevas características del objeto. La implantación de clases y objetos varía dependiendo del lenguaje de programación usado.

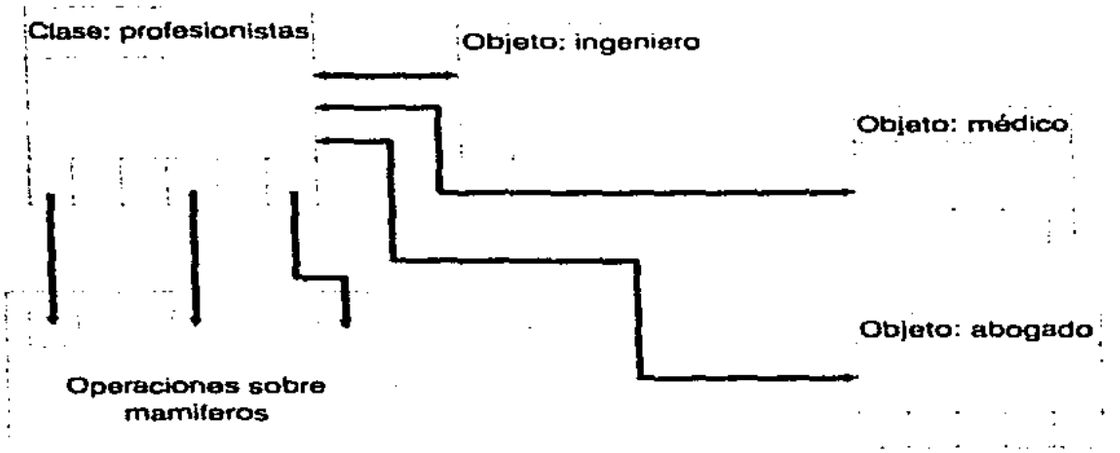


Fig. 1.4.7.2.2.2 Objetos, clases y herencias.

Encapsulamiento.

Cuando se diseña un objeto se debe incluir: 1) el nombre del objeto y referencia a la clase, 2) la especificación de la estructura de datos privada con una indicación de los elementos y tipos de datos, y 3) una descripción procedimental de cada operación. El encapsulamiento se observa cuando alguien distinto al diseñador use el objeto y no necesita los detalles de la implantación, sólo una forma estandarizada y controlada de llamarlo.

1.4.7.2.3 Metodología de diseño orientado a objetos.

1.4.7.2.3.1 Metodología de Booch.

Esta metodología se basa en el trabajo de Booch, la cual se conforma de los siguientes puntos:

1.- Definir el problema.

2.- Desarrollar una estrategia informal para la realización software del dominio del problema en el mundo real.

3.- Formalizar la estrategia mediante los pasos:

a) Identificación de objetos y sus atributos.

b) Identificar las operaciones que pueden aplicarse a los objetos.

c) Establecer interfaces para mostrar las relaciones entre los objetos y las operaciones.

d) Decidir los aspectos del diseño detallado que harán una descripción de la implantación para los objetos.

4.- Repetir los pasos 2, 3 y 4 recursivamente hasta tener un diseño completo.

Los dos primeros pasos deben realizarse durante el análisis de requerimientos.

1.- Definición del problema. Se refiere al análisis de requerimientos. Se deben ejecutar dos subpasos: a) establecer el problema de forma clara y sencilla y b) analizar las ligaduras conocidas. Para estos puntos, puede ser de utilidad primero una narración corta conceptual en lenguaje natural del problema y diagramas de flujo de datos en sus primeros niveles.

2.- Estrategia informal. Consiste en una descripción en lenguaje natural de la solución al problema establecido en la definición del problema. Debe ser algo sencillo, que conserve el nivel de abstracción y enfocarse a lo que debe de hacerse en vez del cómo.

3.- Formalización de la estrategia. Es aquí donde inicia verdaderamente el diseño, y lo que se espera es identificar objetos, operaciones y sus interrelaciones, que conduzcan a un diseño.

a) Identificación de objetos y sus atributos. Esto es la esencia del DOO y de una buena identificación se desprende el éxito de todo el diseño. En este contexto, un nombre común representa una clase de objetos, un nombre propio es para una instancia de una clase y un nombre abstracto indicará las agrupaciones específicas para los objetos o clases. Los adjetivos sirven para fijar los atributos de los objetos. Finalmente, se recomienda plasmar este trabajo en una tabla que contenga nombre del objeto, espacio al que corresponde y sus atributos.

b) Operaciones aplicadas a los objetos: Basándose en los verbos, frases verbales y predicados de la estrategia informal se identifican las operaciones que actúan sobre los objetos.

Objeto	Espacio	Atributo	Operación	Comentario

Tabla 1.4.7.2.3.1.1 Tabla de objetos, atributos y operaciones.

c) Componentes e interfaces del programa: Un aspecto importante a cuidar es la modularidad, esto es, la especificación de los componentes del programa (módulos) que se combinan para formar un programa completo. También se deben identificar las interfaces que existen entre los objetos y su estructura global.

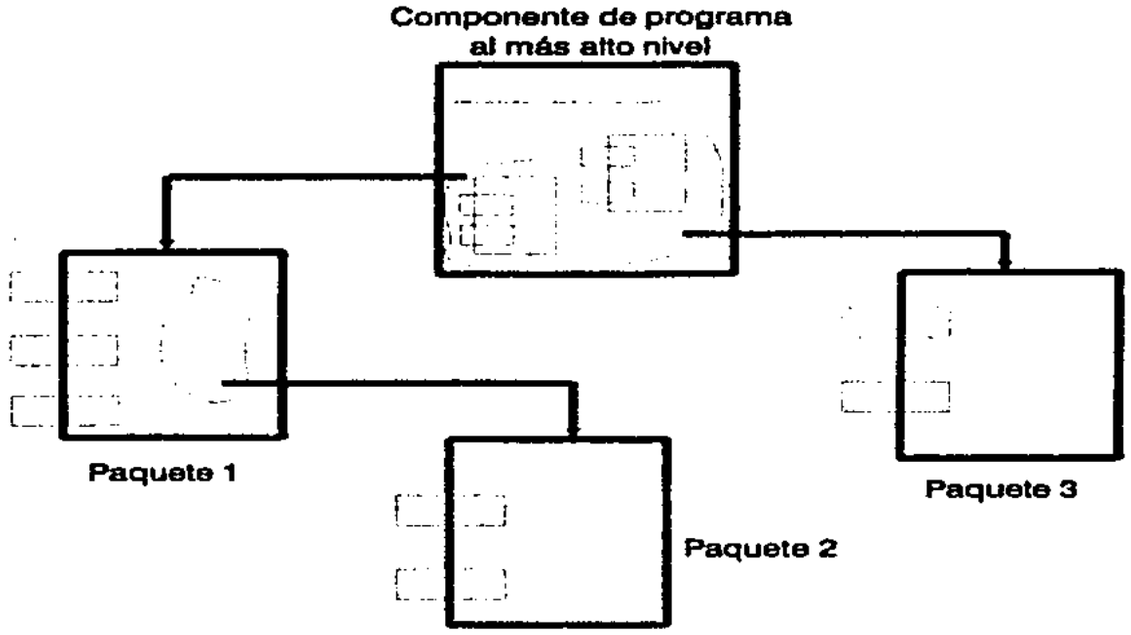


Fig. 1.4.7.2.3.1.1 Diagrama de Booch.

Es importante señalar que a medida que se avanza en el diseño, van surgiendo nuevos objetos que se tienen que incorporar de la forma descrita.

d) Detalle de implantación. Esta parte es similar para cualquier metodología de diseño de software, en el sentido de que se tienen interfaces en detalle, se especifican y refinan las estructuras de datos y se diseñan los algoritmos para cada unidad de programa. El punto de diferencia es que, tanto la estrategia informal como la formalización de la misma, se deben aplicar recursivamente.

1.4.7.2.3.2 Metodología de Lorensen.

1.- Identificar las abstraciones de datos para cada subsistema. Esto significa identificar las clases del sistema las cuales frecuentemente corresponden con objetos físicos dentro del sistema.

2.- Identificar los atributos de cada abstracción. Son las variables de la instancia para cada clase.

3.- Identificar las operaciones de cada abstracción. Son los métodos o procedimientos de cada clase.

4.- Identificar la comunicación entre los objetos. Aquí se deben definir los mensajes que los objetos mandan a los otros. Además, se debe definir una correspondencia entre los métodos y los mensajes que llaman a los métodos.

5.- Probar el diseño con los escenarios. Son los mensajes a los objetos que validan la medida en que se cumplen los requerimientos del sistema.

6.- Aplicar la herencia donde sea apropiada. Si las abstracciones se aplican de forma ascendente, aquí debe aplicarse la herencia, en caso contrario, se hace al momento de identificar las abstracciones para cada subsistema.

1.5

Ambiente Cliente/Servidor

1.5.1 ¿ Qué es cliente/servidor ?

La computación cliente/servidor permite a los desarrolladores de soluciones basadas en sistemas de información que las aplicaciones se dividan en tareas. Cada tarea puede correr en una diferente plataforma, bajo diferente sistema operativo y con un diferente protocolo de red. Cada tarea puede desarrollarse y mantenerse por separado, acelerando así el desarrollo del sistema. Por otro lado, los usuarios están más cerca de los datos y pueden accesarlos por medio de interfaces amigables. Todo lo anterior consecuenta el uso más eficiente del equipo existente y el incremento de la productividad de los trabajadores. Las soluciones cliente/servidor son el producto de la evolución de varias tecnologías, pero sin duda lo que las hace posibles es el software.

Una definición simple de cliente/servidor es: El software del servidor recibe peticiones de datos del software del cliente y regresa los resultados al cliente. El cliente manipula los datos y presenta los resultados al usuario, o actuando como servidor, los envía al cliente que los haya solicitado. Una buena parte del procesamiento de la aplicación se hace en una computadora de escritorio, la cual obtiene los servicios de otra computadora en una configuración maestro/esclavo.

El procesamiento que se da en la red puede ser distribuido o cooperativo. El primero distribuye los datos en dos o más computadoras, pudiendo estar distantes geográficamente, y el usuario tiene acceso transparente a la información. El procesamiento cooperativo reparte una aplicación en dos o más computadoras en una relación "peer-to-peer". La mayoría de las estructuras de red se basan en acceso distribuido, no computación distribuida. La arquitectura cliente/servidor usa una configuración maestro/esclavo donde el procesamiento puede realizarlo tanto el cliente como el servidor. El uso de sistemas abiertos en hardware, software, sistemas operativos, bases de datos y redes, permiten la computación cliente/servidor.

Funciones de una aplicación.

Una aplicación se puede dividir en las siguientes funciones:

1.- Interface de usuario. Lo que el usuario ve.
2.- Lógica de presentación. Lo que sucede cuando el usuario interactua con la interface.
3.- Lógica de aplicación. Las funciones propias del sistema.
4.- Petición de datos y aceptación de resultados.
5.- Integridad de los datos. Validación, seguridad y que estén completos.
6.- Administración física de los datos. Actualizaciones, eliminación y adición de datos.

Un producto denominado EASES el cual emulaba terminal en conjunto con Windows de Microsoft, determinaron el nacimiento de las interfaces gráficas de usuario o GUI's por sus siglas en inglés (Graphical User Interface). Debído a que el procesamiento en una computadora personal PC es más barato que en un mainframe y el procesamiento de la presentación se realiza mejor en esta plataforma. Al dividirse las funciones de aplicación entre el host y una PC nació la idea de cliente/servidor.

La primera generación de aplicaciones C/S operaba en un servidor de archivos. Los problemas típicos eran que para consultar sólo ciertas partes de un archivo era necesario enviar el archivo completo, quedaba bloqueado para otros usuarios y no se podía diferenciar entre consulta y actualización.

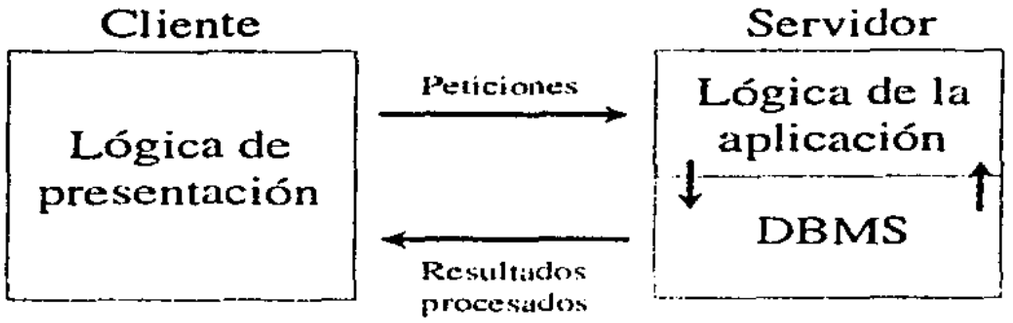


Fig. 1.5.1.1 Primeras aplicaciones cliente/servidor.

En atención a la deficiencia de transmisión de archivos completos, los DBMS de red soportaban múltiples accesos a la misma base de datos y pueden manipular registros o campos individuales, liberando a la red de tráfico de información innecesaria.

Al ocurrir lo anterior, la computación cliente/servidor tomó cierta tendencia donde el procesamiento y presentación de los datos están en el dominio del cliente. Lo que hace el servidor es atender un requerimiento de información y regresar los resultados. Las actividades de validación y seguridad estaban en el DBMS y por tanto eran controladas por el servidor.

En el momento en que el software se volvió más robusto y se incrementó el poder de hardware de las PC's, algunas validaciones y revisión de errores de datos se trasladaron a la parte cliente. Esto tiene la ventaja de que al host sólo llegan peticiones de datos válidas.

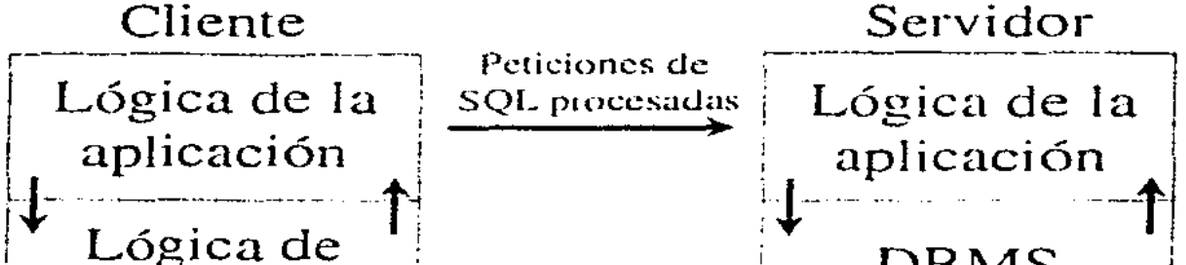


Fig. 1.5.1.2 Distribución del procesamiento en el modelo cliente/servidor.

1.5.2 Arquitecturas de cómputo.

Al menos en México todavía predominan los mainframes con múltiples terminales conectadas a él. Sin embargo a raíz del explosivo crecimiento de las microcomputadoras, se han desarrollado otros tipos de arquitecturas de cómputo, los cuales describo a continuación.

1.5.2.1 Arquitectura céntrica mainframe.

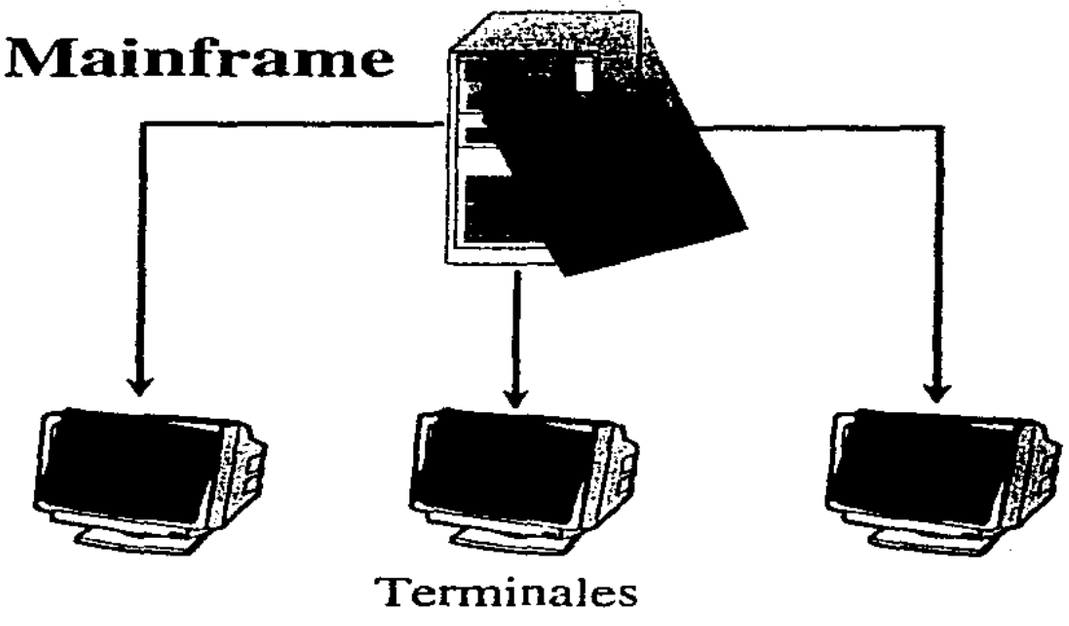


Fig. 1.5.2.1.1 Arquitectura céntrica mainframe.

El procesamiento se realiza en computadoras centrales denominadas mainframes con las que los usuarios interactúan por medio de terminales o emuladores de terminal basados en computadoras personales. Las principales características de esta arquitectura son:

Las interfaces son en su mayoría de sólo caracter y existen excepciones como Xwindows que es una alternativa gráfica.

Todo el procesamiento se realiza en el mainframe, por ello las terminales se denominan tontas puesto que su única función es realizar la transmisión de texto y comandos que el usuario demande a la computadora central. Las terminales se conectan a controladores y los emuladores de PC por medio de modems o redes.

La principal ventaja es la gigante capacidad de procesamiento y almacenamiento de un mainframe, sin embargo su talón de aquiles es el alto costo del tiempo máquina y de los dispositivos auxiliares.

1.5.2.2 Arquitectura céntrica PC-Servidor.

Las PC's comparten las aplicaciones y datos que se almacenan sobre servidores basados en PC. Sus peculiaridades principales son:

El servidor es usado para compartir periféricos, archivos y aplicaciones además de cierto procesamiento que lleva a cabo, esto representa las ventajas principales de este modelo. En algunos casos los archivos se pueden compartir entre aplicaciones.

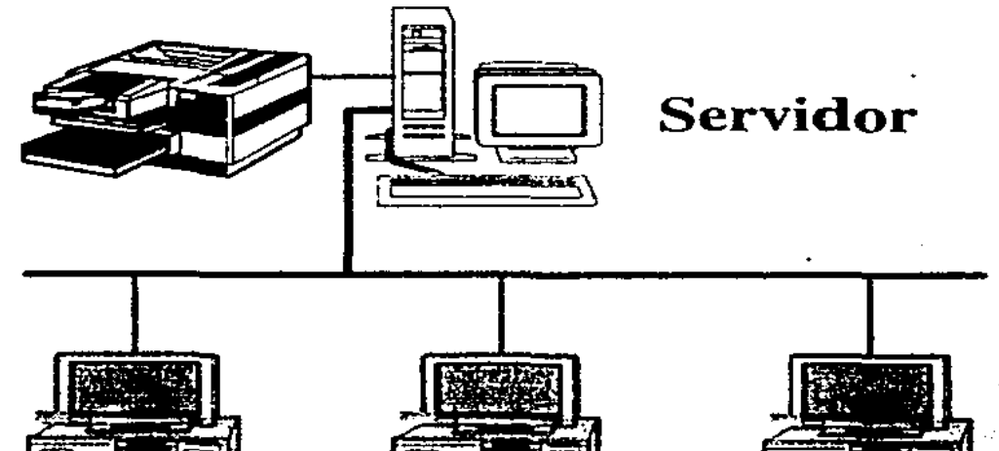


Fig. 1.5.2.2.1 Arquitectura céntrica PC-servidor.

El procesamiento se hace tanto en la PC como en el servidor, por lo que se le exige a las computadoras tener cierto nivel de memoria y capacidad de disco así como un procesador razonablemente rápido. Esto ocasiona un incremento en el costo de operación.

Las aplicaciones pueden accesar al servidor para demandarle datos, esta comunicación se hace vía red con las implicaciones de tráfico que conlleva.

1.5.2.3 Arquitectura punto a punto.

Es una de las opciones más modernas en la distribución del procesamiento, al tratarse de computadoras similares, éstas proporcionan y brindan servicios a cualquier máquina actuando como cliente o como servidor según convenga.

En adición comparten los periféricos que se tengan instalados y recursos como memoria y capacidad de disco de alguna computadora en particular.

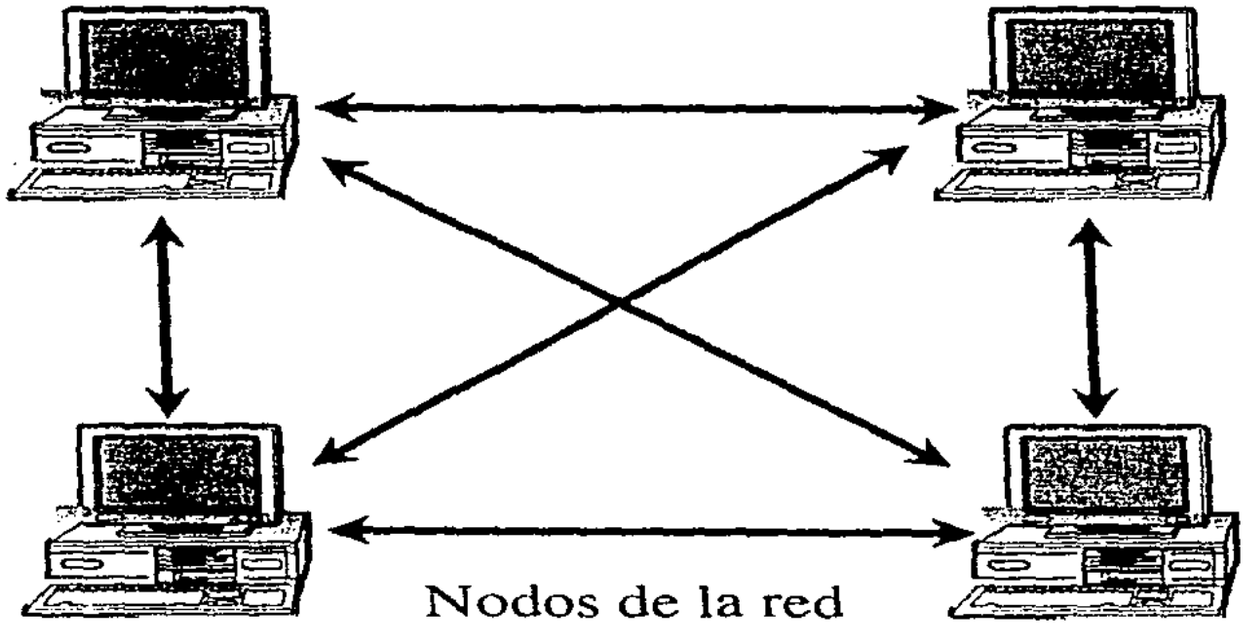


Fig. 1.5.2.3.1 Arquitectura punto a punto.

En el mejor de los casos, este ambiente debe proporcionar transparentemente un procesamiento cooperativo distribuyendo la carga de trabajo sobre diferentes servidores.

1.5.3 Rightsizing, downsizing, upsizing y smartsizing.

Rightsizing.

Rightsizing significa diseñar las aplicaciones para la plataforma que mejor convenga, a diferencia de usar la misma siempre. Una aplicación debe correr en el ambiente en el cual es más eficiente. El modelo cliente/servidor permite que las aplicaciones se dividan en tareas que pueden ejecutarse en diferentes plataformas. Como desarrolladores de aplicaciones cliente/servidor debemos analizar todas las funciones que debe cumplir un sistema y decidir cuáles conviene ejecutarlas en el servidor y cuáles en el cliente.

Downsizing.

Se hace downsizing a una aplicación de host cuando se rediseña para correr en un ambiente más pequeño como una red de área local.

Upsizing.

Se hace este proceso cuando las aplicaciones se rediseñan para ejecutarse en ambientes más grandes, como mainframes.

Smartsizing.

También conocida como "reingeniería de procesos de negocio", esta actividad implica cambios radicales a los procesos de negocio dentro de la organización y en su caso, el uso de la tecnología para apoyar esos rediseños.

1.5.4 Evolución de la computación cliente/servidor.

Tendencias de hardware.

Las terminales reemplazaron a las tarjetas perforadas, las PC's que emulaban terminal sustituyeron a las terminales, las minicomputadoras están desplazando a los mainframes y las LAN junto con la computación cliente/servidor están reemplazando a las minicomputadoras y a los mainframes. Esto se debe al alto costo de los mainframes y al decremento en el mismo y el aumento en la capacidad de las PC's. Las empresas tratan de utilizar los host para procesos que sólo los host pueden desempeñar.

Los procesadores que utilizan los equipos para aplicaciones C/S son frecuentemente de la familia Intel y se producen en masa. La capacidad de memoria se ha incrementado a un ritmo impresionante desde 1986. Los equipos destinados para operar como servidores manejan arreglos de discos y alta capacidad de memoria con la intención de incrementar su desempeño.

Tendencias de software.

Algunos manejadores de bases de datos que soportan la tecnología C/S como Sybase, Informix o Ingress usan SQL, el lenguaje de acceso a datos estándar para aplicaciones C/S. Algo muy importante es que no todas las bases de datos en este modelo son relacionales y algunas se heredan de ambientes anteriores.

Las GUI's proveen un ambiente operativo sobre el sistema operativo de las máquinas de escritorio el cual ha coadyuvado a la aceptación de C/S.

Tendencias de las redes.

Las primeras redes se usaban para compartir impresoras o plotters, tiempo después permitían el intercambio de archivos completos a través de la red. Los últimos avances han logrado que sólo se transmitan los registros que se requieren, evitando el tráfico de la red con información innecesaria. La mayoría de los desarrollos C/S se basan en redes de área local.

1.5.5 Beneficios del modelo cliente/servidor.

Los beneficios del modelo cliente/servidor se pueden observar en la siguiente tabla.

Beneficios del modelo cliente/servidor
Ahorro de dinero
Incremento en la productividad del usuario final
Incremento en la productividad del desarrollador
Flexibilidad y escalabilidad
Mayor utilización de recursos
Control centralizado
Sistemas abiertos

Tabla 1.5.5.1 Beneficios del modelo cliente/servidor.

Ahorro de dinero. Los ambientes de mainframe son costosos de mantener si se considera hardware, software y el personal especializado para operarlo. En el caso de cliente/servidor estos costos son considerablemente menores. En caso de que la organización donde el modelo operativo crezca, al usar tecnologías escalables es posible adaptar el equipo a un precio razonable; en el caso de los mainframes se tendrían que reemplazar por completo. Adicionalmente, en algunos casos se puede aprovechar el equipo existente además de que el costo de los desarrollos para cliente/servidor son menores.

Incremento en la productividad del usuario final. Dado que esta tecnología se apoya en GUI se tienen bondades como el uso por medio de mouse, colores, dibujos, menús, etc. Al tener este tipo de ambientes, el usuario trabaja de forma más agradable que con una terminal de sólo carácter y tiende por tanto a ser más productivo. Según estadísticas esta proporción es del 30%.

Incremento en la productividad del desarrollador. Debido a que los ambientes de desarrollo y las herramientas son más sencillas de utilizar, y en ocasiones tienen automatizadas o preconstruidas algunas funciones, los desarrolladores son más productivos.

Flexibilidad y escalabilidad. Como la computación cliente/servidor es modular, nos permite añadir, sustraer o reemplazar módulos de una forma prácticamente transparente.

Mayor utilización de recursos. Si se centra el uso de los clientes en las interfaces de usuario y procesamiento de aplicaciones, y los servidores en almacenamiento y procesamiento de mayor nivel, se logra un uso eficiente de los recursos de cómputo. Otro aspecto interesante es el hecho de que el cliente sólo hace peticiones prevalidadas al servidor, evitando así el tráfico innecesario a través de la red.

Control centralizado. Esta arquitectura hace posible tener un control centralizado de los datos en el servidor que incluyen procedimientos de validación de acceso a la información.

Sistemas abiertos. Para lograr un uso efectivo se deben soportar múltiples ambientes.

Al trasladar el procesamiento de los mainframes a C/S, las empresas pueden posponer la compra o la actualización de un equipo grande. Dentro de este modelo, las aplicaciones requieren menos hardware, software y tiempo de mantenimiento, todo esto se traduce en ahorro de dinero, incremento en la productividad y capacidad de responder con oportunidad a las demandas de un mercado global cada vez más exigente.

1.5.6 Componentes del modelo cliente/servidor.

Una aplicación cliente/servidor tiene tres componentes: un cliente, un servidor y una red. Cada uno de éstos a su vez, tienen una parte de software y otra de hardware.

1.5.7 El cliente.

El usuario del software y hardware del cliente interactúa con la lógica de presentación, ésta a su vez con la lógica de aplicación y con el software de red del cliente, el cual manda la petición al servidor,

a su regreso, recibe el resultado del servidor y lo transfiere al software del cliente. En algunos casos el software del cliente procesa la información que viene del servidor antes de pasarla a la componente de presentación del software.

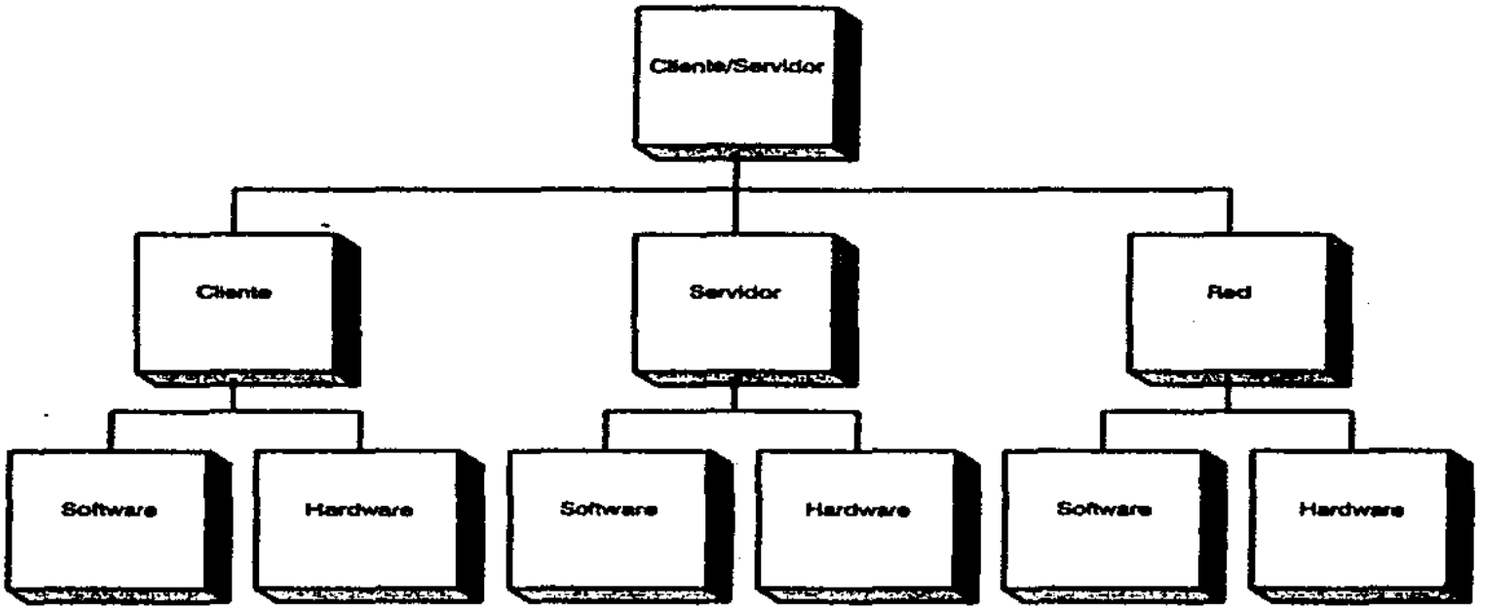


Fig. 1.5.6.1 Componentes de una aplicación cliente/servidor.

Dawna Trevis recomienda que, contrario al flujo de selección tradicional, en un desarrollo C/S primero se elige la plataforma y el sistema operativo para la parte cliente. Posteriormente se debe seleccionar el software de desarrollo y el de aplicación.

1.5.7.1 Hardware del cliente.

El hardware del cliente es la computadora que ejecuta el software del cliente y los elementos de comunicación de red, pudiendo ser una computadora personal o una estación de trabajo (workstation). Este hardware tiene que ser lo suficientemente robusto para soportar los requerimientos de presentación y el procesamiento desarrollado en el cliente de la aplicación.

La cantidad de memoria y capacidad del disco que el cliente requiera depende de la clase de procesamiento (basado en: host, cliente o cooperativo) y de la complejidad de la solución. Las dos últimas requieren suficiente memoria para ejecutar la lógica de la aplicación y mayor velocidad de procesamiento para obtener buenos tiempos de respuesta. La demanda de espacio se deriva de la cantidad de información que se necesite almacenar localmente.

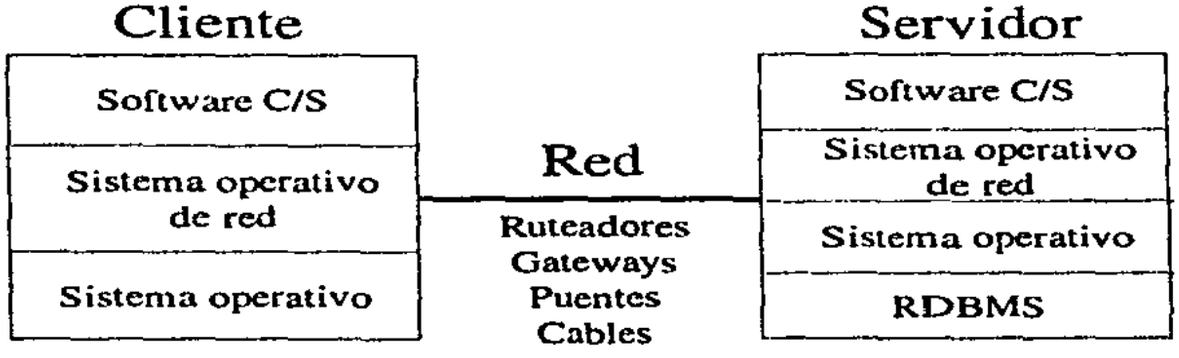


Fig. 1.5.7.1.1 Componentes de la computación cliente/servidor.

1.5.7.2 Software del cliente.

Las cuatro categorías de software que puede ejecutar una máquina cliente son: el componente de presentación, la lógica de aplicación, el sistema operativo y el sistema operativo de red.

Componente de presentación

El componente de presentación genera la interface con la que el usuario interactúa. Este componente no sabe dónde se encuentran los datos, sus funciones son aceptar solicitudes de información del usuario, enviar las peticiones al servidor, recibir los resultados del servidor y manipularlos y formatearlos de acuerdo a lo establecido.

En la mayoría de los casos es una interface gráfica de usuario (GUI) que como su nombre lo dice provee un ambiente gráfico y simula multitarea. La información se presenta en áreas conocidas como ventanas y las tareas como dibujos denominados íconos. Los tiempos de respuesta deben ser breves para satisfacer las expectativas de los usuarios y coadyuvar a la alta productividad.

Las interfaces más populares son Windows de Microsoft, Presentation Manager de IBM, Motif de Open Systems Foundation (OSF) y OpenLook de USL. La intención de las GUI's es reducir para el usuario el uso del teclado y simplificar la administración del sistema.

Es conveniente que cada organización fije estándares para la construcción de las interfaces gráficas como medida de reducción de las curvas de aprendizaje de los usuarios, e incidiendo por tanto en una mayor productividad. Por otro lado, los diseñadores de interfaces deben situarse siempre en la posición del usuario, considerar que una GUI bien diseñada elimina la necesidad de documentación, simplificar el uso de funciones frecuentes, indicar en todo momento al usuario lo que el sistema está haciendo, ser consistentes en la apariencia de las funciones comunes, contener ayuda en línea y evitar un intercambio constante entre el uso de mouse y teclado. Otra característica importante que debe satisfacer una GUI es ser portable, ya que de lo contrario la aplicación tampoco podrá serlo.

Cada GUI tiene sus propias interfaces de programación de aplicaciones (API- Application Programming Interface) que son una serie de rutinas para hacer algunas funciones y ligar diferentes tipos de software. Es importante destacar que no todas las interfaces son gráficas, de hecho en algunos casos conviene que sean de sólo caracter.

Lógica de aplicación.

El cliente ejecuta aplicaciones creadas con herramientas de desarrollo para C/S que son la esencia del software de la parte cliente debido a que son con las que el usuario trabaja y para ellos suelen ser todo el sistema. En algunos casos un cliente es un servidor actuando como cliente. A esto se le denomina agente.

Sistema operativo cliente.

Otro elemento es el sistema operativo que corre en el hardware del cliente que puede ser distinto del S.O. del servidor. Este sistema operativo debe ser lo suficientemente robusto para soportar el procesamiento de la presentación y de la lógica de la aplicación. Las interfaces gráficas se montan sobre los S.O., de los productos mencionados, Windows corre bajo DOS, Presentation Manager bajo OS/2 y Motif y OpenLook bajo UNIX.

Tres elementos que incorporan los sistemas operativos clientes son:

a) Librerías de ligado dinámico (DLL, Dynamic link libraries). En ellas se codifican rutinas y se ligan a las aplicaciones que las requieran en forma de módulos.

b) Intercambio dinámico de dátos (DDE, Dynamic Data Exchange). Estos elementos se usan para intercambiar automáticamente datos entre aplicaciones que corren bajo Windows.

c) Ligado y unión de objetos (OLE, Object Linking and Embedding). Se usa para crear reportes con una serie de objetos con las ligas al software en que fueron creados.

Software de comunicaciones.

Un elemento adicional es el software de comunicaciones que se ejecuta en el hardware del cliente. Sus funciones son manejar las transmisiones de peticiones hacia el servidor y sus respuestas. Todo lo relacionado a redes y software de comunicaciones lo cubro con amplitud en el punto 1.2.

Funciones del cliente	Funciones del servidor
* Interface gráfica de usuario.	* Manejo de archivos, impresión y servidor de base de datos.
* Procesamiento distribuido de la aplicación.	* Procesamiento de la aplicación distribuido.
* Correo electrónico.	* Correo electrónico.
* Aplicación local.	* Comunicaciones.
* Emulación de terminal.	* Administración de la red.
	* Administración de recursos.

Tabla 1.5.7.2.1 División de actividades en el modelo cliente/servidor.

1.5.8 El servidor

El término "servidor" se aplica a una computadora que provee de servicio a otras máquinas en una red. Este abarca administración de datos, aspectos de red, control de versiones de aplicaciones, procesamiento de una parte de la lógica de aplicación, compartición de archivos, envío de peticiones a otras máquinas y dar respuesta a las peticiones que le hacen los clientes entre otras funciones.

Un servidor es la máquina que ejecuta el software de administración de datos y se diseña para funcionalidad de servicio. Comparado con una microcomputadora, un servidor tiene más memoria, mayor capacidad de almacenamiento, mucho más poder de procesamiento que en algunos casos llegan a procesadores en paralelo, fuentes ininterrumpibles de energía y discos espejo para respaldos, principalmente.

1.5.8.1 Hardware del servidor.

En la arquitectura cliente/servidor este último se encarga de administrar los datos, haciendo funciones de almacenamiento, actualización, bloqueo de registros, ordenamiento, consolidación y protección de los mismos. El servidor atiende las peticiones de datos y de ser necesario, genera y envía peticiones a otros servidores.

El servidor puede ser para usos específicos de una organización o atender a una organización completa. Puede caer dentro de la categoría de simple, un superservidor, un microservidor o un servidor de base de datos.

Puntos de referencia (Benchmarks).

Existen organizaciones encargadas de realizar medidas de desempeño de los servidores. Una de estas organizaciones es la TPC (Transaction Processing Counsel), la cual integra compañías de hardware y software de todo el mundo, con el objetivo de crear puntos de referencia respecto a bases de datos y procesamiento. TPC creo tres áreas de interés, denominadas A, B y C. Estas áreas se concentran en medir el desempeño en tres direcciones: número de transacciones por segundo, desempeño de procesamiento de transacciones tipo batch y capacidad de procesamiento de todo el sistema cliente/servidor respectivamente.

1.5.8.2 Categorías de servidores.

El concepto de servidor evolucionó cuando las organizaciones tuvieron la necesidad de compartir periféricos de alto costo. Sin embargo hoy en día existen servidores que comparten datos en adición a la premisa original.

Los seis tipos de servidores son:

1.- De archivos.
2.- De aplicaciones.
3.- De datos.
4.- De cómputo.
5.- De bases de datos.
6.- De comunicaciones.

1.5.8.2.1 Servidores de archivos.

Este tipo de servidores manejan aplicaciones para trabajo en grupo y archivos de datos que se pueden compartir por ese grupo. Estas máquinas hacen un uso extensivo de la entrada/salida y de los medios de almacenamiento secundario. Una peculiaridad es que cuando se hace una petición de información, el servidor transmite por la red todos los registros de un archivo y su índice. El cliente por su parte, selecciona los registros basándose en un criterio de consulta o carga la información en memoria, y de ahí se revisa. Por estas características de operación, estos servidores requieren discos duros de gran capacidad y velocidad.

El bloqueo de archivos se hace bloqueando todo el archivo o un rango de bytes. En caso de que los usuarios quieran accesar el mismo archivo compartido simultáneamente, el motor del servidor revisa la contención. En caso de que exista al nivel de bloqueo de archivos, es necesario esperar hasta que esté desocupado.

Debido a que no hay un motor disponible para toda la información, no hay administración de caché ni de bloqueo y poco control de concurrencia en el DBMS. El hecho de tener que enviar todo el archivo por la red para hacer operaciones de filtrado, ordenamiento o depuración, consecuenta un alto tráfico. Dos técnicas que se usan para minimizar este efecto son organizar la información en bloques contiguos y hacer varias copias de los datos.

1.5.8.2.2 Servidores de aplicaciones.

En algunos casos es un host o una máquina que actúa como tal. Cuando se bajan las aplicaciones de host, una opción es instalarlas en equipos más pequeños que ejecuten el mismo software que en el ambiente original con la ventaja de no requerir modificaciones a las aplicaciones que corrían en la primer plataforma.

1.5.8.2.3 Servidor de datos.

Estos servidores se usan exclusivamente para almacenamiento y administración de datos en conjunto con uno o varios servidores de cómputo. El servidor de datos no realiza ninguna labor de procesamiento de la lógica de aplicación. Es preciso tener procesadores rápidos, gran memoria y capacidad de disco, dado que se realizan búsquedas y actualizaciones de cantidades masivas de datos. En contraparte, se tiene un tráfico muy pequeño por la red.

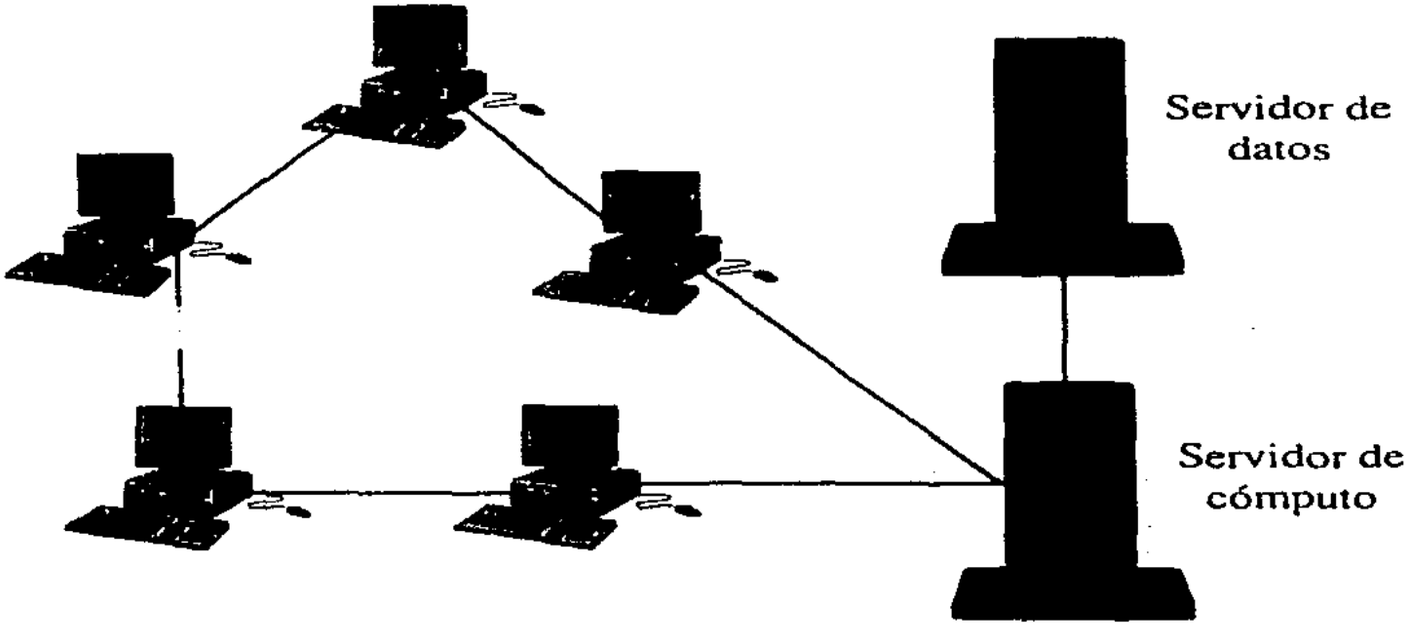


Fig. 1.5.8.2.3.1 Servidor de datos y servidor de cómputo.

1.5.8.2.4 Servidores de cómputo.

Un servidor de cómputo hace la petición de datos a un servidor de datos y envía de regreso los resultados a los clientes. Son capaces de desempeñar lógica de aplicación a los resultados de datos antes de pasarlos a los clientes.

Estos servidores exigen potentes procesadores y grandes cantidades de memoria RAM, en contraparte no se necesita un disco duro muy poderoso.

1.5.8.2.5 Servidores de bases de datos.

Este es el tipo de uso más común en la arquitectura cliente/servidor. La mayor parte del procesamiento y la lógica de presentación se hace en el cliente y las funciones de administración de datos en el servidor. Estos servidores reciben la petición de datos , recuperan la información y la hacen llegar al cliente. En este sentido, su funcionamiento es equivalente a un servidor de cómputo junto con uno de datos.

Gracias a que el motor de base de datos se encuentra separado del cliente, se puede tener manejo de bloqueos, administración de caché multiusuario y no es necesario redundancia en los datos.

La forma de operar es recibiendo una petición SQL desde el cliente y regresando sólo los datos que satisfacen su petición. Esto lo hace mucho más eficiente en el aspecto de tráfico en la red y debido a las bondades de agrupación de acciones de las sentencias SQL, incluso las peticiones mismas.

La demanda de recursos de los servidores de bases de datos depende del volumen de la base de datos manejada, la velocidad que se requiera para actualizar los datos, el número de usuarios y el tipo de red.

1.5.8.2.6 Servidores de comunicaciones.

Proveen gateways a redes, computadoras medias y mainframes. Su demanda de recursos mayor se centra en procesadores rápidos y múltiples slots para soportar los protocolos de red.

1.5.8.3 Rasgos distintivos de los servidores.

1.5.8.3.1 Multiprocesamiento.

Tener múltiples procesadores permite a los servidores realizar multiprocesamiento simétrico o funcional. El primero hace que una tarea se pueda asignar dinámicamente a cualquier procesador logrando una maximización de recursos. Sin embargo, para que esto sea realidad el sistema operativo de red, el sistema operativo del servidor y el software de aplicación deben soportar multiprocesamiento.

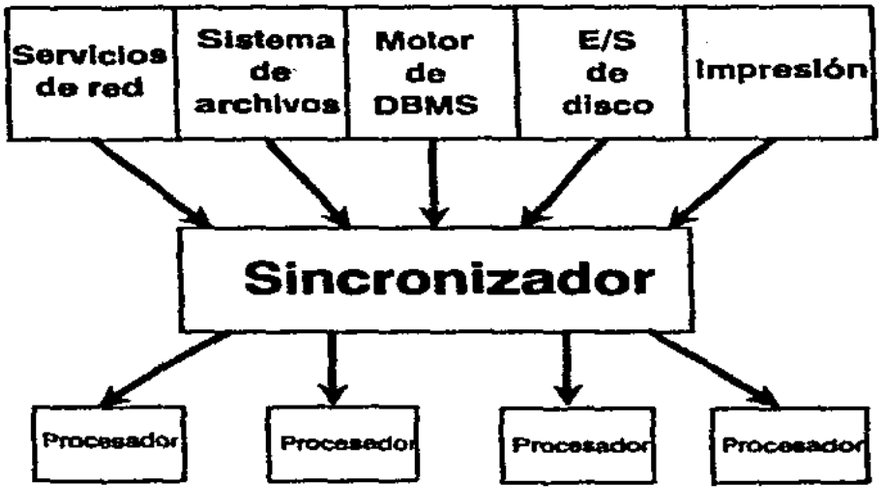


Fig. 1.5.8.3.1.1 Multiprocesamiento simétrico.

El multiprocesamiento funcional asigna permanentemente un conjunto de tareas a un procesador. Algunos superservidores con buses EISA (Extended Industry Standard Arquitecture) soportan masterización de bus, que es una forma de multiprocesamiento funcional.

Los procesadores pueden tener acoplamiento ajustado o por separado, en la primer categoría cada uno tiene su memoria por separado y en la segunda la comparten por medio de un bus especial. El multiprocesamiento soporta multitarea.

1.5.8.3.2 Multitarea.

Una tarea es la mínima unidad de ejecución que el sistema puede preparar para su ejecución. Cada tarea tiene un stack, un apuntador de instrucción, una prioridad, el status del CPU y un acceso en la lista de actividades a ejecutar. Una tarea puede encontrarse bloqueada, lista para ejecutarse o en ejecución.

Las tareas se comunican enviándose mensajes entre ellas y compiten por los semáforos, los cuales organizan la asignación de recursos de cómputo entre las tareas individuales. Las tareas hacen la petición al sistema para ejecutar una instrucción, en caso de que ninguna esté lista, la tarea se suspende hasta que tenga algo que hacer. Cuando la instrucción está lista, se ejecuta y se hace una nueva petición de trabajo al sistema.

En un ambiente de multitarea, un proceso se particiona en tareas ejecutables independientemente, y en conjunto desempeñan el trabajo de un programa. Esto posibilita que una aplicación ejecute varias tareas simultáneamente.

1.5.8.3.3 Arreglos de discos.

Existen arreglos de discos tolerantes a fallas de forma estándar en los superservidores, identificados como arreglos de discos redundantes y baratos (RAID, Redundant Arrays of Inexpensive Disks). Para su operación trabajan en conjunto con un servidor de archivos y software para el control del acceso a los drivers individuales. El sistema operativo del servidor lo ve como un solo drive lógico.

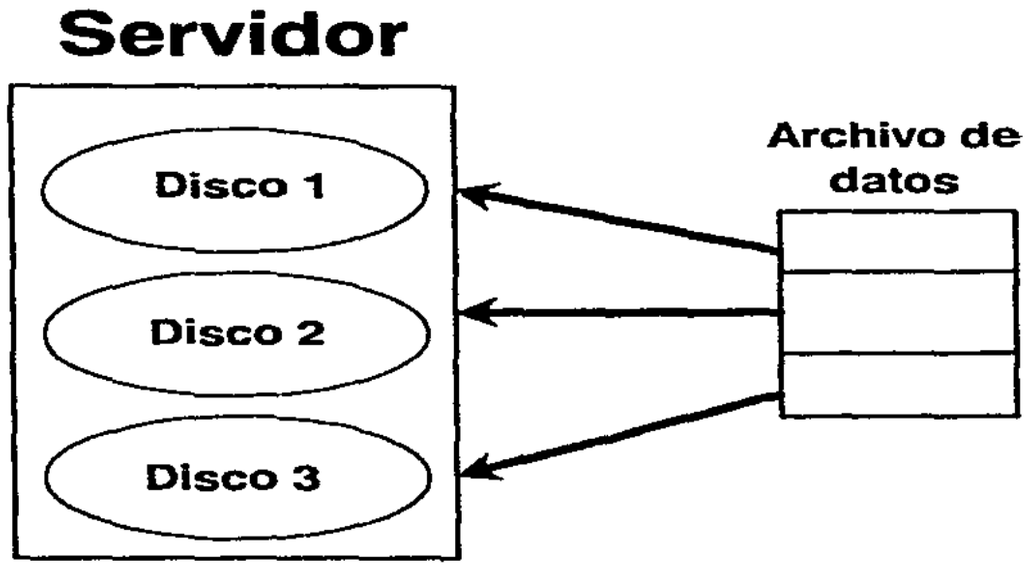


Fig. 1.5.8.3.3.1 Arreglos redundantes y baratos de discos. (RAID's)

Los RAIDS pueden recuperarse transparentemente a la falla de cualquier disco y permiten que se cambie el disco dañado manteniendo en línea al servidor. Los bloques de datos se particionan en pequeñas partes que se escriben simultáneamente a varios discos. Cuando un disco falla, los datos se pueden reconstruir recolectando esas partes. El desempeño del servidor se afecta cuando se da este proceso de reconstrucción, pero se tiene la ventaja que es transparente al usuario. Existen cinco niveles de protección y corrección de datos, conocidos como RAID 1 al 5.

1.5.8.3.4 Subsistemas de memoria.

Para prevenir el daño de los datos cuando viajan por el servidor se usa el código de corrección de errores de memoria y verificación de paridad (ECC, Error Correction Code). La memoria ECC es un subsistema que corrige automáticamente errores en bits individuales y detecta errores en múltiples bits.

Cada que se escriben datos al arreglo de discos, se escribe un ECC a un disco extra o a un disco de almacenamiento primario. El código para un ECC se calcula por medio de una fórmula de corrección de error que se aplica a los datos escritos a los otros drivers. Si un dato falla, la información que se pierde se reconstruye del ECC, los datos intactos de los otros drivers y la fórmula. Es posible que se presenten errores en la codificación, pero sólo afectan un bit de información dado que cada bit se escribe en un chip diferente.

1.5.8.3.5 Componentes redundantes.

Es una característica opcional en los microservidores y obligatoria en los superservidores que consiste en tener componentes dobles de discos, fuentes de poder y facilidades de recuperación automática.

1.5.8.4 Clases de servidores.

Existen cinco clases de servidores: microservidores, superservidores, servidores de bases de datos, computadoras de tamaño medio y máquinas tolerantes a fallas.

1.5.8.4.1 Microservidores.

Un microservidor es una computadora personal que se ha habilitado para realizar funciones de servidor. Los elementos que se adicionan para desempeñar estas funciones son un procesador más veloz de lo usual, mayor cantidad de memoria y un mayor disco duro.

1.5.8.4.2 Superservidores.

Son máquinas creadas específicamente para la arquitectura cliente/servidor. Se caracterizan por tener procesadores múltiples, grandes cantidades de memoria, espacio en disco y alta velocidad. Aparecieron en el mercado desde 1989.

Las características principales de un superservidor son:

a) Mayor poder de procesamiento. Gracias al uso de múltiples procesadores.

b) Alta capacidad de manejo de las funciones de entrada/salida. Derivado del uso de varios buses o procesadores de entrada/salida.

c) Gran capacidad de disco. Se usan arreglos de varios discos, que se ven como un solo drive lógico.

d) Mejor administración de memoria. Al usar chips de memoria más rápidos, mejores arquitecturas de memoria y memoria caché.

e) Mayor confiabilidad. Al tener componentes redundantes.

f) Mejor mantenimiento. Esto se logra al incorporar software para administración y corrección de errores remotamente.

Dos ejemplos de superservidores son el IBM Server 295 y el Compaq System Pro.

1.5.8.4.3 Servidores de bases de datos.

Pueden desempeñarse como servidores de bases de datos de alta velocidad o como servidores de aplicaciones. Pueden tener cientos de procesadores en paralelo que les permitan hacer búsquedas, y manipulación de datos en general a super alta velocidad, además de alojar bases de datos gigantescas.

1.5.8.4.4 Máquinas tolerantes a fallas.

Cuando una aplicación se migra de un mainframe a un servidor, resulta crítico para las organizaciones que sigan operando de manera normal para los usuarios finales. La tolerancia a fallas se define como la habilidad de un componente para resistir una falla grave sin dañar los datos o interrumpir el servicio. Para ello, este tipo de equipos tienen hardware redundante, v.gr. dobles procesadores, sistemas de memoria, discos, elementos de comunicación y fuentes de energía alternas.

Estas máquinas tienen sistemas de hardware por separado operando simultáneamente, una como sombra del otro. Constantemente se intercambian mensajes que reflejan el status de la operación y la sincronización de ambos equipos. En caso de falla, el sistema de respaldo entra en acción inmediatamente, mientras el otro puede repararse sin interrumpir la operación. Es recomendable que el equipo de respaldo se localice físicamente en otro lugar, para prevenir algún evento catastrófico como terremoto, incendio o inundación. Debido a las características mencionadas, estos servidores tienen un alto costo.

1.5.8.5 Software del servidor.

Las ocho categorías de software que puede haber en un servidor son:

1.- Administración del ambiente de red.
2.- Ambiente y extensiones de red.
3.- Sistema operativo de red.
4.- Sistema operativo del servidor.
5.- Módulos de carga.
6.- Administración de la base de datos.
7.- Gateways de base de datos.
8.- Software de aplicación.

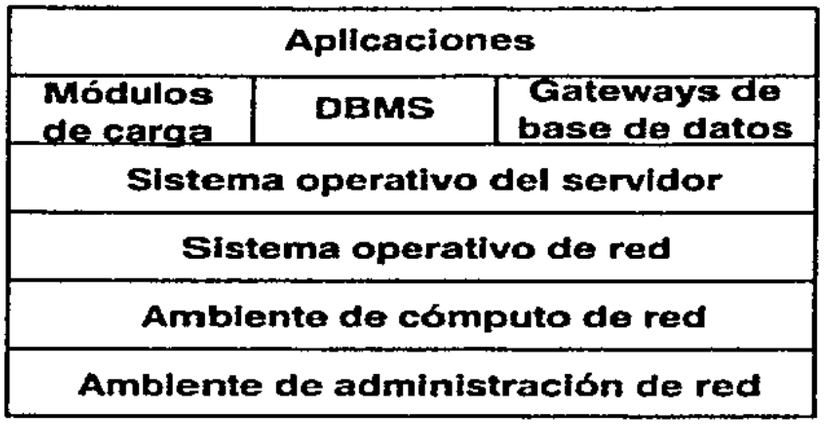


Fig. 1.5.8.5.1 Ocho categorías de software del servidor.

La aplicación manda una petición de datos que se envía por el sistema operativo de red a los gateways de bases de datos, administradores de bases de datos o a un módulo de carga. Estos elementos traducen esta petición a código máquina y la pasan al sistema operativo del servidor. Este último se guía por el software de ambiente de red, el cual maneja el servicio necesario para soportar varias aplicaciones en un ambiente heterogéneo. Las actividades completas de la red se manejan por el software de administración de ambiente de red.

1.5.8.5.1 Administración del ambiente de red.

Las redes y los sistemas de información interactúan de forma estrecha en la arquitectura cliente/servidor. Por lo tanto, su administración debe verse como un solo proceso. Existe software que se está desarrollando para administrar las tecnologías de información interconectadas. Algunos ejemplos son la administración de ambiente distribuido (DME, Distributed Management Enviroment), arquitectura de administración de objetos (OMA, Object Management Arquitecture) y UI-Atlas.

1.5.8.5.2 Ambiente de red.

Permite que las aplicaciones se puedan distribuir entre diferentes plataformas y distinto software. Los estándares son el DCE (Distributed Computing Enviroment) y el ONC de UNIX (Open Network Computing).

1.5.8.5.3 Extensiones.

La más importante extensión de DCE es la habilidad para administrar transacciones en línea a través de sistemas abiertos distribuidos. Una transacción se puede entender como una unidad básica de trabajo, que puede ser una petición SQL. En los ambientes distribuidos el manejo de información es más complejo debido a que los datos pueden residir en diversas máquinas y redes. Para asegurar la consistencia de los datos, el software de administración de datos usa comunicación servidor a servidor y entregas de dos fases.

1.5.8.5.4 Sistema operativo de red.

El sistema operativo de red administra los servicios que ofrece el servidor y evita que los programas de aplicación se comuniquen directamente con el hardware.

1.5.8.5.5 Módulos de carga.

Los módulos de carga son software que reside en el servidor cuyo objetivo es mejorar la funcionalidad del sistema operativo del servidor. Un ejemplo popular de estos módulos son los Netware Loadable Modules de Novell Netware.

1.5.8.5.6 Sistema operativo del servidor.

El sistema operativo del servidor maneja los recursos del servidor, interactúa con el sistema operativo de red y con el software de manejo de datos para recibir y responder a las peticiones de servicio por parte de los usuarios. Los sistemas operativos de servidor más populares son el OS/2, UNIX y Windows NT.

Estos son sistemas operativos de 32 bits que tienen como características principales respecto a procesadores de 16 bits como el 486, que soportan multitarea, un mayor espacio de direccionamiento, un bus físico más amplio, un operando de instrucciones y un tamaño de registros más grande.

Dos de los beneficios principales de los procesadores de 32 bits es el espacio de direccionamiento que supera los 4 gigabytes, y la eliminación de manejo de datos y programas en segmentos de 640 bytes. Estos aspectos hacen a los programas más rápidos y eficientes.

El sistema operativo de servidor que más me interesa es UNIX, puesto que es el que uso para el desarrollo de la tesis. Respecto a este sistema, se da una explicación amplia en el punto 1.1.

1.5.8.5.7 Requerimientos del servidor.

El software de base de datos es el elemento crítico en el ambiente cliente/servidor, considerando que administra los datos, procesa transacciones en línea, asegura la integridad referencial y realiza procedimientos de recuperación.

Algunas características importantes que debe cubrir un servidor son:

1.- Independencia de plataforma. El software debe ser independiente de la plataforma que se use. De esta manera, se garantiza que las migraciones hacia ambientes mayores o menores sean prácticamente transparentes.

2.- Procesamiento de transacciones. Una transacción es una o más operaciones que se realizan juntas para completar una tarea. Cuando el sistema se cae, el software de base de datos debe regresar las transacciones que estaban en proceso.

3.- Commits de dos fases. Un commit asegura la consistencia y el término del procesamiento de una transacción, cuando se usa más de una tabla.

4.- Esquemas de bloqueo. Aseguran que el registro que un usuario esté usando esté protegido, así cuando una transacción bloquea un registro, no puede ser actualizado hasta que se libere.

5.- Logs de transacción. Estos archivos se usan cuando es necesario recuperar una base de datos después de una caída del sistema.

6.- Conectividad. El software del servidor debe tener acceso a una gran variedad de fuentes de datos y no sólo a un vendedor.

7.- Llamadas de procedimiento remotas (Remote Procedure Calls). Son API's localizadas en la parte alta del mecanismo de comunicación de red, que permiten a los servidores comunicarse entre sí.

8.- Bases de datos inteligentes. Las aplicaciones C/S requieren que alguna parte de la lógica de aplicación se guarde junto con la base de datos. Esa lógica puede ser un verificador de integridad, un trigger o un procedimiento almacenado. El software de base de datos del servidor debe manejar integridad referencial que asegure que los datos relacionados en diferentes tablas sean consistentes.

9.- Procedimientos almacenados (stored procedures). Son una serie de instrucciones SQL que se compilan y guardan en la base de datos del servidor. Estos procedimientos se analizan y se revisa su sintaxis la primera vez que se utilizan, almacenando la versión compilada en memoria caché. De esta manera, las llamadas subsecuentes utilizan esta versión y son por tanto más rápidas.

10.- Disparadores (triggers). Son procedimientos de almacenamiento especiales ligados con tablas específicas que se llaman automáticamente por el software de base de datos del servidor. Se ejecutan cuando se hace un intento por modificar las tablas a las que están asociados. Usando RPC puede hacerse una revisión de integridad referencial en bases de datos separadas en una red.

11.- Mediación de la carga. En los desarrollos C/S debe decidirse cuidadosamente qué parte de la aplicación se procesa en el servidor y cuál en el cliente. Cuando se logra un buen balance se consigue minimizar el tráfico de la red.

12.- Optimizador. Es software que determina la manera más eficiente de procesar una transacción SQL.

13.- Herramientas de prueba y diagnóstico. Permiten recuperar datos de acciones impredecibles como grabado en sectores dañados del disco o causadas por otros datos.

14.- Mecanismos de respaldo y recuperación de datos. Los medios más comúnmente usados son en ese orden las cintas, discos duros y discos ópticos. Existe una variante que es usar al host como medio de respaldo y recuperación.

1.5.8.6 Aspectos para evaluar un servidor.

Al evaluar un servidor se deben tomar en consideración los siguientes tres aspectos:

a) Confiabilidad. Cuánto tiempo pasa entre fallas.

b) Disponibilidad. Cuánto tiempo transcurre para que el sistema vuelva a la normalidad después de una falla.

c) Flexibilidad y escalabilidad. Qué tan fácilmente pueden expandirse las capacidades del servidor.

El servidor contiene el software del sistema operativo, el software de administración de datos y una parte del software de red. El sistema operativo tiene que interactuar confiablemente con el software de red y ser lo suficientemente robusto para manejar la tecnología del software.

El software de administración de datos responde a las peticiones de información que se hacen desde los clientes. Las bases de datos relacionales y SQL se han convertido en los estándares de facto para la estructura y para el lenguaje de acceso a datos. Los DBMS tienen incorporada mucha funcionalidad heredada de los sistemas de mainframe, tales como rutinas de respaldo, de recuperación, herramientas de prueba y diagnóstico. Dos aspectos claves para tener éxito en las aplicaciones cliente/servidor son separar la administración de la presentación de otras aplicaciones y la distribución de la lógica de la aplicación entre el servidor y el cliente.

1.5.8.7 Administración de datos en el servidor.

El software de base de datos reside en el servidor y se le accesa por las herramientas front-end del cliente. Este software debe proveer de integridad, seguridad, funcionalidad y ser lo suficientemente robusto como su contraparte de mainframe.

1.5.8.7.1 SQL.

El SQL (Structured Query Language) se ha convertido en el lenguaje estándar para acceso a datos en las aplicaciones cliente/servidor. El SQL incluye:

a) Lenguaje de definición de datos. Define las estructuras de datos y la relación que guardan entre sí.

b) Lenguaje de manipulación de datos. Mueve y actualiza los datos.

c) Lenguaje de control de datos. Define acceso y seguridad.

El SQL procesa los datos en bloques. Esto permite que sólo los registros que satisfagan la petición sean enviados al cliente, lo que resulta en un ahorro considerable de tiempo de comunicación y por tanto de dinero.

La forma de operar de SQL en las aplicaciones C/S es que primero el cliente hace una consulta SQL que se envía al RDBMS del servidor. El servidor ejecuta la consulta y regresa los datos que satisfagan al cliente. Esto parte de que se usa un modelo de datos relacional. Cuando no es así, es necesario utilizar gateways de bases de datos.

Arquitectura de bases de datos distribuidas.

Algunas empresas requieren que sus datos y el DBMS estén dispersos entre múltiples sistemas, pero el acceso a los mismos debe ser transparente para los usuarios finales. El tener datos distribuidos exige que se cumplan las siguientes reglas: debe minimizarse las actualizaciones distribuidas y los datos distribuidos deben estar lo más cerca posible del procesador que los atiende.

1.5.8.7.2 Gateways de bases de datos.

Son interfaces traductoras que mueven datos, comandos SQL y aplicaciones de un tipo de base de datos a otra. Deben tener el detalle en cuanto a sintaxis, formato de datos, tipo de datos, convenciones de nombres de los productos que accesan y mantenerse al tanto de las actualizaciones de software. La desventaja de este mecanismo es la velocidad de respuesta, debido a que es necesario hacer traducciones entre ambientes. El gateway de base de datos puede ser para plataformas iguales o distintas. Entre sus funciones se encuentran:

a) Aceptar peticiones de una aplicación cliente que usualmente es vía SQL.
b) Traducir las peticiones a un formato específico de base de datos.
c) Mandar las peticiones para su ejecución a la base de datos.
d) Traducir los resultados para su regreso en un formato conocido.
e) Regresar los datos al cliente.

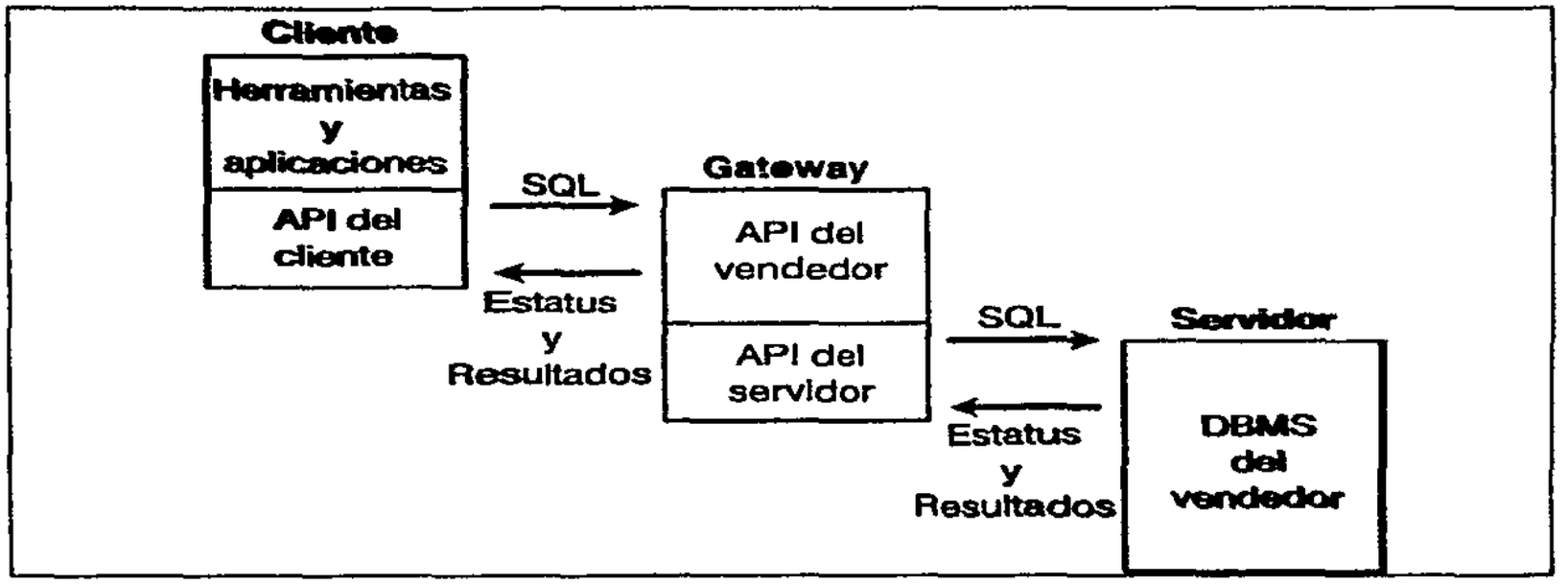


Fig. 1.5.6.7.1 Gateway de base de datos.

1.5.9 La red.

La red debe ser transparente y confiable para el usuario como si se estuviera corriendo en una sola computadora. Por otro lado, debe tener un diseño flexible que permita adecuarse a las necesidades de la organización donde se use.

La red transmite peticiones de información del cliente al servidor, canaliza las demandas entre los servidores y hace posible la comunicación entre los nodos.

1.5.9.1 Hardware de red.

El hardware de red son las tarjetas de comunicación, los cables y en general todos los dispositivos que conectan el servidor y a los clientes. Las conexiones deben permitir a los usuarios accesar la información desde cualquier nodo y lograr la comunicación entre servidores.

Existen cinco formas de interconectar redes de área local:

1.- Repetidores. Cuando una señal viaja a través de un cable pierde intensidad, los repetidores se usan para devolver a una señal su fuerza original. Otro uso es para ampliar el alcance de una red. Para que puedan usarse en ambos lados de la red deben existir las mismas características físicas de transmisión y los mismos protocolos.

2.- Puentes. Conectan redes con diferentes características físicas de transmisión y protocolos. Pasan tramas de datos de una red a otra basándose en el direccionamiento de la capa de ligado de datos del modelo OSI.

3.- Ruteadores. Se usan para conectar dos redes diferentes pero que usan el mismo protocolo. Administran la selección de rutas para el envío de paquetes de datos y minimizan las cargas de tráfico de las redes. Después de que el ruteador determina la mejor ruta para un mensaje, se pasa junto con dos direcciones adicionales, la del siguiente nodo y la ruta final. Trabajan en la capa de red del modelo OSI. Su desempeño se mide al igual que el de los puentes, en número de tramas por segundo.

4.- Gateways. Se usan para interconectar redes completamente diferentes. Hacen las funciones necesarias para ir de un protocolo a otro. Sus actividades son: conversión de formatos de mensajes, traducción de direcciones y conversión de protocolo.

5.- Backbone. Es una conexión entre redes que usa para lograr este propósito enlaces de microondas o fibra óptica. Requieren un gran ancho de banda y transmitir con alta confiabilidad. Un backbone permite a varias redes trabajar en paralelo.

1.5.9.2 Software de red.

La comunicación y el flujo de datos a través de la red es posible gracias al software de red. El sistema operativo de red administra los procesos de entrada y salida del servidor alusivos a la red. Cada

sistema operativo, como se explica en el punto 1.2, tiene su propio protocolo que es un conjunto de reglas que definen los formatos, el orden del intercambio de datos y cualquier acción que se dé en el proceso de recepción y transmisión de datos.

Es de suma importancia comprender perfectamente la parte de red en una aplicación C/S, debido a que causa la mayoría de los problemas de configuración.

1.5.9.3 Sistema operativo de red.

Las redes necesitan software para proteger a los programas de aplicación de tener una comunicación directa con el hardware. Un sistema operativo de red administra los servicios del servidor, de la misma forma que un sistema operativo común administra los servicios de hardware.

El sistema operativo de red reside en las capas de sesión y de presentación del software de administración de red de las máquinas cliente. Cuando se usa una red existente para aplicaciones C/S, no se requiere nuevo software.

Los sistemas operativos de red más comunes son:

* Netware de Novell
* LAN Manager de Microsoft
* OS/2 LAN Server de IBM
* VINES de Banyan

Netware de Novell.

En sistemas basados en servidor como Netware, el sistema operativo de red, las aplicaciones y los archivos se almacenan en una máquina dedicada que administra el tráfico a través de la red. En una instalación peer-to-peer, el sistema operativo de red se instala en cada nodo que tiene acceso a la red.

El Netware se puede usar en arquitecturas de bus o tokeng ring. Algunas de las funciones que soporta son: compartición de archivos y de dispositivos, correo electrónico, acceso remoto y comunicación entre redes por medio de un puente. Dentro de las utilerías se facilita administración de usuarios, cambio de passwords y administración de la seguridad del sistema. Un servidor Netware puede integrar en una red clientes de DOS, OS/2, Macintosh y UNIX. Netware soporta IPX/SPX como protocolo nativo y TCP/IP.

Los servicios de nombre de Netware simplifican la administración de múltiples servidores. Así, cada grupo de servidores, denominado dominio, tiene un nombre de servidor para dar seguimiento a los passwords de usuarios y privilegios de la red.

Como tolerancia a fallas, Novell ofrece discos espejos y duplex. Así mismo, posee una utilería para tener servidores espejo.

Dentro de las principales desventajas de Netware destacan que no fue diseñado para soportar múltiples redes de área local, y que el kernel no puede retomar el control si un módulo de carga falla.

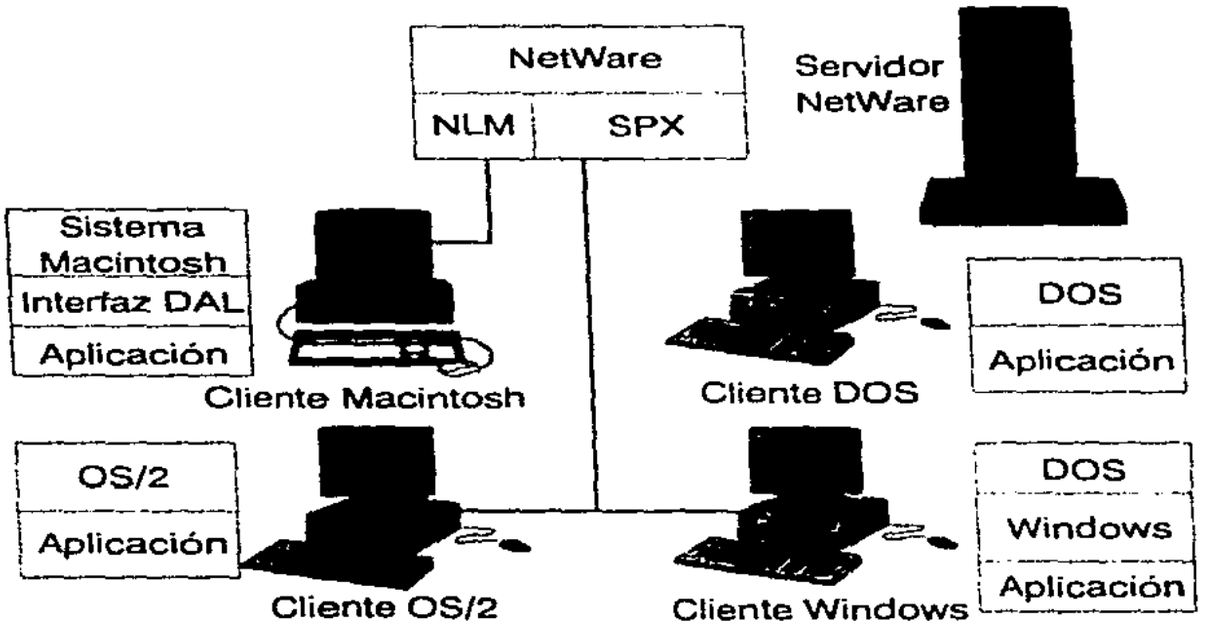


Fig. 1.5.9.3.1 Arquitectura de Netware.

1.5.10 Clases de procesamiento en las aplicaciones cliente/servidor.

Esta clasificación se basa en ubicar dónde se realiza la mayor parte del procesamiento.

Procesamiento basado en host.

En este tipo de aplicación se tiene una capa de aplicación corriendo en la microcomputadora y todo el procesamiento de la aplicación en el host como servidor. En este caso sólo se aprovecha el poder de procesamiento de la PC para generar una interfaz fácil de utilizar.

El procesamiento basado en host demanda menos funcionalidad en el cliente. Debido a que la aplicación en el host interactúa de la misma manera que con una terminal tonta y por ello necesita menos coordinación.

Lo que le da sentido a este tipo de aplicación es que el usuario es más productivo con la interface sencilla de utilizar.

Procesamiento basado en el cliente.

Aquí toda la lógica de la aplicación reside en la máquina del cliente, exceptuando las rutinas de validación de datos que se codifican en el DBMS en el servidor. Estas aplicaciones se crean para sacar provecho de las nuevas arquitecturas, en lugar de sólo simular terminales de host.

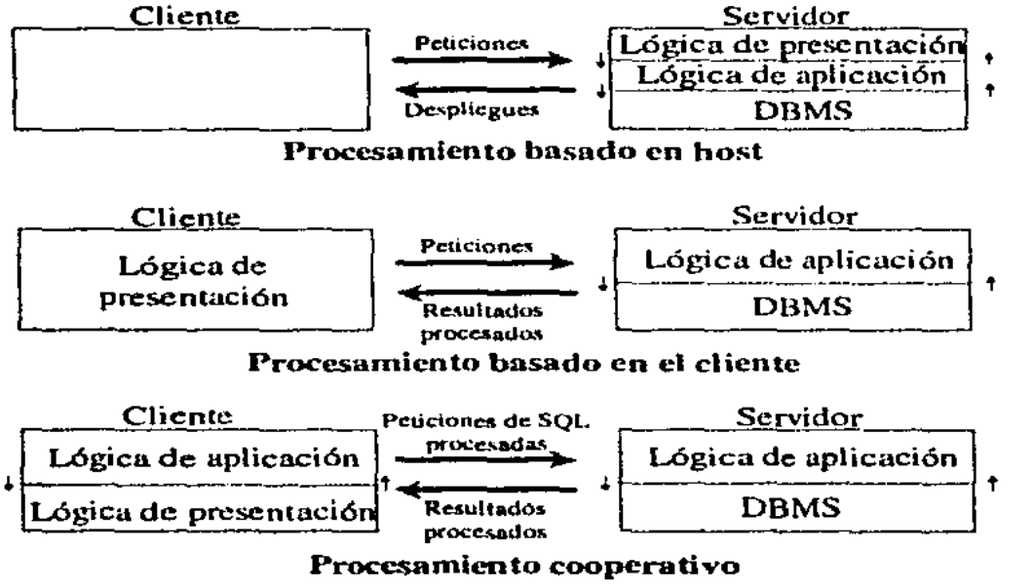


Fig. 1.5.10.1 Clases de procesamiento en las aplicaciones cliente/servidor.

Al requerir coordinación entre diferentes plataformas y su software, el uso de la red se vuelve más sofisticado. Además es necesario integrar redes, hardware y software de diferentes fabricantes. La mayoría de las aplicaciones C/S actuales corresponden a esta clasificación.

Procesamiento cooperativo.

En este caso se da un procesamiento cooperativo a la par. Todos los componentes del sistema son iguales y pueden exigir o proveer servicios a cualquier otro. El procesamiento se da en cualquier parte donde estén los recursos de cómputo. La manipulación de datos se puede dar tanto en el servidor como en el cliente, donde sea más apropiado. Este procesamiento requiere un alto nivel de coordinación.

1.5.11 Categorías de aplicaciones cliente/servidor.

Esta clasificación se basa en la función que la aplicación desempeña.

Sistemas de oficina.

En este tipo de aplicaciones se tiene un sistema para comunicación electrónica, en la cual se hace posible la comunicación entre usuarios independientemente de la configuración de la arquitectura de red que posean. Ejemplos de estas aplicaciones son el correo electrónico (E-mail) y software para trabajo en grupo.

Creación de interfaces para sistemas existentes.

Aquí sólo se convierten las pantallas de solo carácter a interfaces gráficas, con el beneficio mencionado de incremento en la productividad del usuario. Estas aplicaciones caen dentro del procesamiento basado en host.

Aplicaciones orientadas a bases de datos.

Algunas aplicaciones C/S se hacen para permitir el acceso vía GUI a los datos corporativos. Estas aplicaciones proveen una sola forma de interactuar a los datos de la organización. El incremento a la productividad que se logra con estas aplicaciones se estima midiendo la facilidad con la que los usuarios pueden accesar los datos para realizar su trabajo y secundariamente, la sencillez de uso de las GUI's.

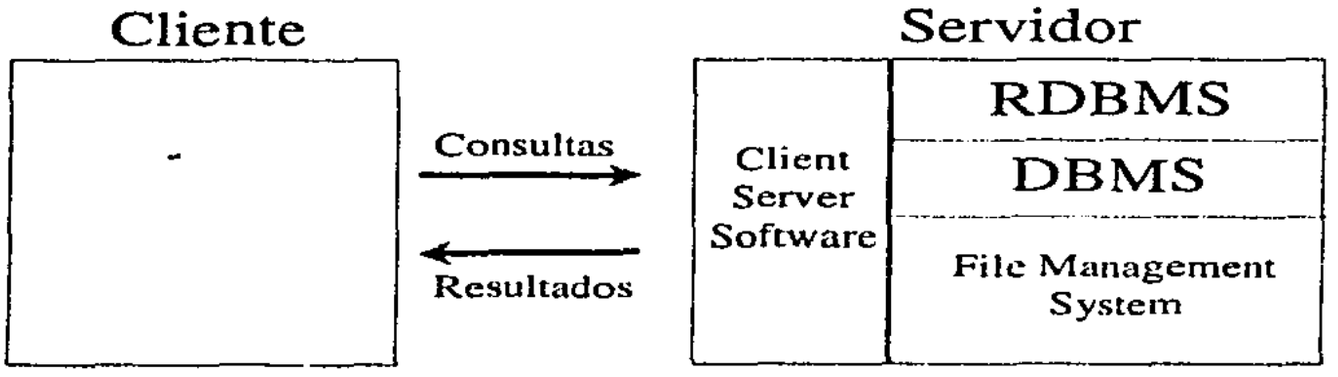


Fig. 1.5.11.1 Aplicaciones orientadas a bases de datos.

Aplicaciones de misión crítica.

También conocidas como de procesamiento de transacciones, estas aplicaciones deben estar funcionando de manera continua, en caso de que fallen por algún momento la organización sufre severas repercusiones. Algunos ejemplos de estas aplicaciones son: sistemas de punto de venta, sistemas de reservaciones aéreas y control financiero de aseguradoras y casas de bolsa.

Una transacción es una o más operaciones que se realizan en equipo, éstas se generan por los clientes y se envían al servidor para su procesamiento, quien a su vez puede enviar operaciones a otros servidores.

Para que una transacción se considere completa, se deben realizar exitosamente todas las operaciones. Si alguna operación de una transacción no se puede concluir, las operaciones que se han completado se deben regresar a su estado inicial.

Aplicaciones de investigación o de apoyo a toma de decisiones

A diferencia de los sistemas de procesamiento de transacciones en tiempo real, las aplicaciones de investigación trabajan con datos que pueden replicarse de los datos originales. Este tipo de aplicaciones se les conoce además, como sistemas de apoyo a la toma de decisiones o sistemas de información ejecutiva.

1.5.12 Aspectos a considerar alrededor del modelo cliente/servidor.

1.5.12.1 Mitos.

Como toda tecnología nueva, existen algunos mitos alrededor del modelo C/S:

Las aplicaciones C/S son fáciles de desarrollar.

Debido a que en la tecnología C/S se requiere integrar hardware y software que puede ser de distinto fabricante, no es una tarea fácil. Otro aspecto es que algunas personas consideran al modelo C/S como parte de la microcomputación, cuando en realidad se trata de complementar lo mejor de dos mundos, microcomputadoras y mainframes.

En esencia, se requieren equipos que sean expertos en mainframes, computadoras medias, microcomputadoras y redes. Se necesita visualizar todo el contexto e integrarlo.

Los equipos de escritorio con que se cuenta actualmente son suficientes.

Esto sólo es posible si los equipos existentes satisfacen las demandas de procesamiento que las interfaces gráficas y la lógica de la aplicación demandan.

Se requiere una mínima capacitación para los desarrolladores.

Como se mencionó anteriormente, la computación C/S está integrada por clientes, servidores y redes, con hardware y software para cada uno de estos elementos, y los desarrolladores deben dominar todos estos aspectos. Por lo tanto esta premisa es falsa.

Todos los datos se alojan bajo un ambiente relacional.

En los desarrollos C/S se heredan algunos aspectos de tecnologías anteriores, y se puede dar el caso de que se trabajen con modelos de datos no relacionales, tales como jerárquico o red invertido.

El tiempo de desarrollo es más corto.

Como algunas aplicaciones C/S se desarrollan para comunidades de usuarios más pequeñas, existen herramientas de desarrollo que lo hacen más fácil y rápido, se usan prototipos y se involucra al usuario más directamente en el diseño; el tiempo de desarrollo debe ser más corto que en el caso de mainframes. Esto se debe balancear y manejar con cautela hacia los directivos de la empresa por la existencia de las curvas de aprendizaje de las nuevas herramientas y el esfuerzo del alto nivel de integración de tecnologías y equipos de trabajo diferentes.

1.5.12.2 Obstáculos.

Costo.

El costo del hardware y software para las aplicaciones C/S es frecuentemente una décima parte del que representa el de los mainframes. Un ejemplo de los costos por MIPS (Millones de instrucciones por segundo) son:

Tipo de máquina	Costo por MIPS en dólares
Mainframes IBM	$100,000
Máquinas medianas	$50,000
Microcomputadoras	$300

Tabla 1.5.10.2.1 Costo por MIPS en dólares por tipo de máquina.

Sin embargo, hay que considerar que el costo de las soluciones C/S comprende, en caso de que se parta de que no existe infraestructura previa:

1. El software y hardware de red.
2. El software y hardware del servidor.

3. El software y hardware del cliente.
4. El software para desarrollo de aplicaciones.

En algunos casos, puede resultar más caro desarrollar una aplicación en C/S que basada en mainframes, por lo que antes del desarrollo de cualquier proyecto que apunte a esta dirección, debe hacerse un estudio de viabilidad financiera del mismo.

Plataformas mixtas.

Actualmente algunas empresas cuentan con una gama muy variada de elementos de cómputo, v.gr. varios sistemas operativos de red, diversos sistemas operativos de microcomputadora, diferentes topologías de red y más de una marca de mainframes. La meta es mantener todos los elementos de cómputo de diversos tipos trabajando para un fin común, lo cual dista mucho de ser una tarea fácil.

En el mercado nos encontramos con las siguientes competencias, por mencionar algunas:

a) Windows de Microsoft vs. OS/2 de IBM por las GUI's.
b) OS/2, UNIX y Windows NT en la parte de sistemas operativos para el servidor.
c) Topologías ethernet o tokeng ring en la parte de redes.
d) Novell Network o LAN Manager para los sistemas operativos de red.
e) SYBASE SQL Server, INFORMIX, INGRESS y ORACLE en el terreno de RDBMS.

Mantenimiento.

Gracias a que las aplicaciones C/S son modulares y en algunos casos se basan en tecnologías de desarrollo orientadas a objetos, el proceso de actualización de código es más directo y sencillo que en los sistemas de host.

Seguridad.

Los sistemas deben ser tan seguros como la organización donde se utilicen lo demande. Pero indistintamente deben contemplar la existencia de unidades de respaldo y recuperación de información, software de monitoreo y mecanismos ininterrumpibles de energía, por mencionar algunas medidas de seguridad.

Rediseño de los procesos de negocio.

Al poner más cerca y de una forma más fácil la información a los usuarios, y estando conscientes, usuarios y desarrolladores de aplicaciones, de usar el poder de cómputo para fines de resolver los problemas de la empresa, la tecnología C/S puede contribuir al rediseño y la mejora continua de nuevos procesos de negocio dentro de una organización.

1.5.12.3 Estándares y sistemas abiertos.

Los sistemas abiertos basados en estándares proveen la flexibilidad, portabilidad e interoperabilidad necesaria para lograr un máximo beneficio de la computación C/S.

Portabilidad significa que el software puede correr en plataformas distintas a donde fue creado sin necesitar modificaciones. Interoperabilidad se refiere a que el software pueda interactuar con otro software fuera de la plataforma inicial.

Los estándares deben afectar cuatro direcciones: plataformas, redes, middleware y aplicaciones. Algunos ejemplos de estándares son: UNIX, TCP/IP, OSI, RDA y DRDA.

Los sistemas abiertos inician con plataformas de operación de estándares y se componen de una serie de estándares para la computación distribuida, redes y desarrollo de aplicaciones. Los sistemas abiertos son una metodología para integrar tecnologías divergentes, creando un ambiente flexible para

desarrollar aplicaciones usando hardware y software abiertos. Los beneficios que nos ofrecen estos sistemas son plataformas escalables, interoperabilidad y una mejor administración y planeación de los recursos de cómputo.

Entre las organizaciones que actualmente se dedican a la fijación de estándares tenemos: Open Software Foundation, Unix International, X/Open, Object Management Group y SQL Acces Group.

1.5.12.4 Factores para el éxito.

Para que el ambiente C/S tenga éxito se debe cumplir la interconexión de redes (internetworking) y la interoperabilidad.

Internetworking.

La red de trabajo de una empresa (internetwork) es la topología de toda la estructura de información basada en cómputo.

El modelo C/S requiere una red bien diseñada que pueda crecer junto con las necesidades de la empresa y tenga un retorno de inversión atractivo y en poco tiempo. Una red debe ser segura y manejable para garantizar que las aplicaciones C/S puedan desempeñarse en el ambiente de producción tal como fueron diseñadas.

Interoperabilidad.

Todas las piezas de una aplicación deben interactuar y verse como una sola entidad aún viniendo de diferentes fabricantes. Esto se hace posible, como mencione antes, gracias al papel de los organismos de creación de estándares y sistemas abiertos.

1.5.13 Metodología de desarrollo de aplicaciones bajo el modelo cliente/servidor.

Existen cuatro componentes principales de la metodología:

1.- Análisis.

2.- Diseño.

3.- Implantación y pruebas.

4.- Mantenimiento.

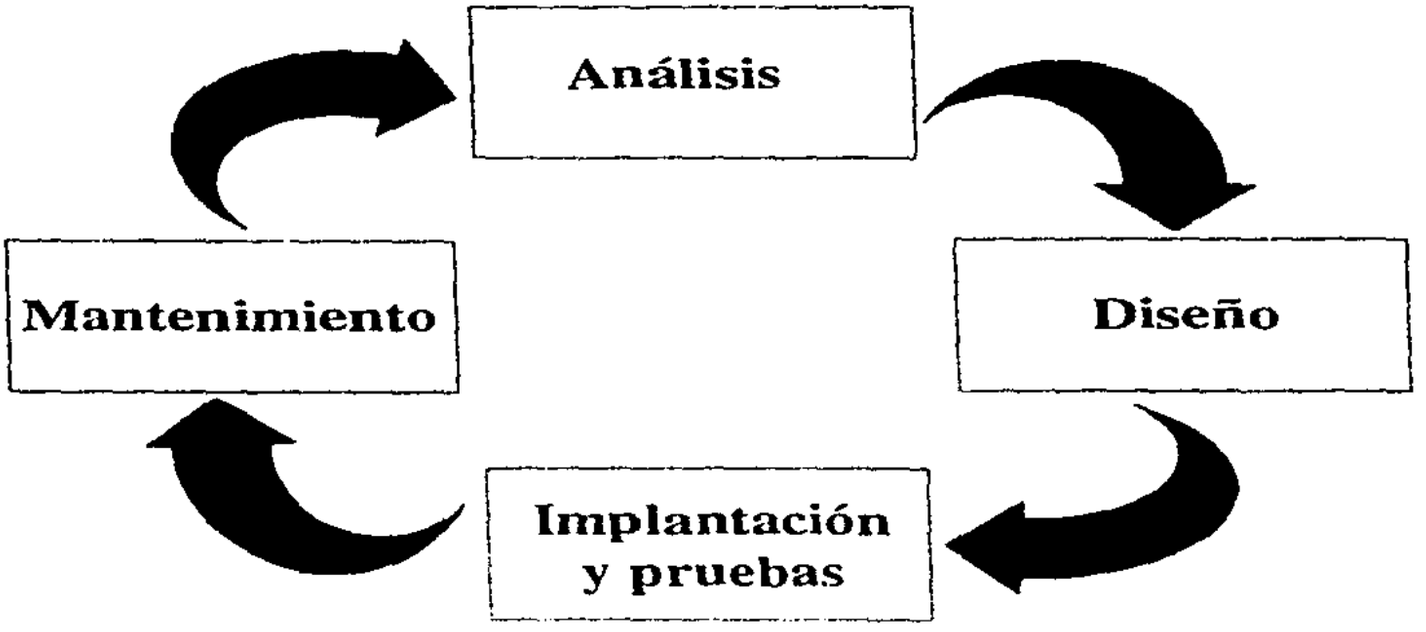


Fig. 1.5.13.1 Fases del proceso de desarrollo.

A la fase de análisis la precede una actividad conocida como "Requerimientos del proyecto", en la cual se obtiene información general acerca del problema, las necesidades a resolver y estimaciones de los beneficios y costos.

1.5.13.1 Análisis.

El objetivo de esta parte es proporcionar a la dirección encargada del proyecto, información suficiente que permita una toma de decisiones inteligente basada en la justificación económica y en la factibilidad técnica de desarrollar el proyecto en cuestión. Por otra parte, aquí se deben definir los requerimientos de funcionamiento, ambiente, desempeño y datos que el sistema a crear debe cubrir.

El análisis se divide en tres fases:

A) Investigación inicial.

B) Estudio de factibilidad.

C) Definición de requerimientos.

1.5.13.1.1 Investigación inicial.

Contiene los primeros informes del impacto técnico y económico de la solución buscada. La investigación inicial permite a la dirección estimar el potencial del sistema propuesto en términos de beneficios y costos. Provee un status de los problemas existentes, costos del proyecto y fechas para el desarrollo y operación del sistema propuesto, genera además un plan de trabajo para el estudio de factibilidad.

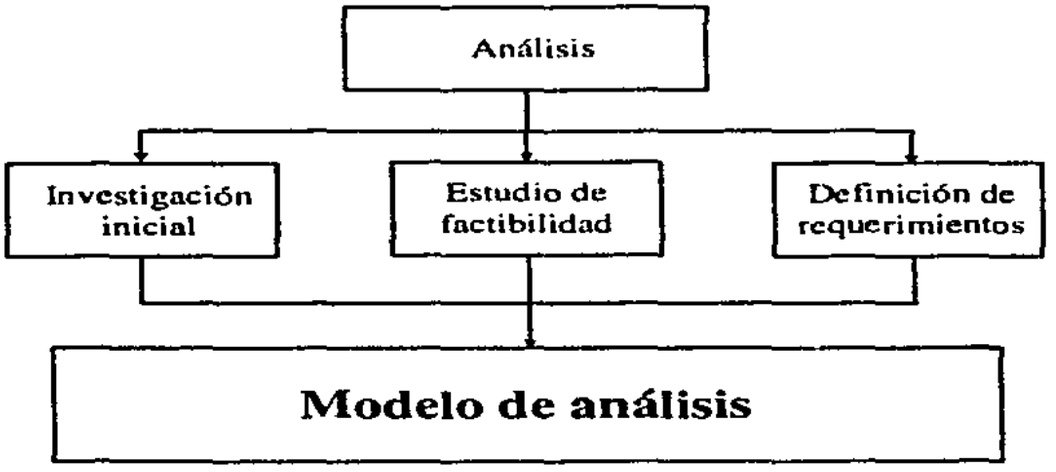


Fig. 1.5.13.1.1.1 El proceso de análisis.

Se divide en ocho funciones principales:

1.- Fija los requerimientos del proyecto. Determina el problema y las necesidades, y valida que los beneficios y costos indicados en el requerimiento del proyecto sean realizables.

2.- Establecer un repositorio. Se destina un lugar para alojar la documentación y los datos recolectados.

3.- Preparar la investigación inicial. La idea es organizar las entrevistas y preparar el material de apoyo para ellas.

4.- Realizar las entrevistas con las personas seleccionadas. El objetivo es conseguir la información de las personas involucradas para evaluar los beneficios y costos del proyecto.

5.- Analizar la información recopilada. El objetivo es analizar la información que se tiene para hacer determinaciones preliminares acerca del tiempo a invertir y el costo asociado.

6.- Preparar el plan de trabajo para realizar el estudio de factibilidad.

7.- Preparar el reporte de la investigación inicial. Se trata de preparar el informe para la revisión por parte de la dirección y su visto bueno para iniciar. El contenido del reporte es un resumen ejecutivo y el plan de trabajo para el estudio de factibilidad.

8.- Hacer la documentación de la investigación inicial.

1.5.13.1.2 Estudio de factibilidad.

La idea es proporcionar a la dirección información suficiente que posibilite una toma de decisiones inteligente basada en un análisis de la justificación económica y la factibilidad técnica. Para su realización se estudian las características de la solución o sistema actual y los requerimientos del sistema propuesto. Contempla ocho funciones.
1.- Revisión de los datos de la investigación inicial.

2.- Conducir entrevistas con los usuarios. El objetivo es conseguir información de los datos a utilizar y del proceso administrativo, para evaluar los beneficios y costos del proyecto.

3.- Revisión del sistema actual. La meta es entender lo que hace el sistema actual, en caso de existir, y cómo lo hace.

4.- Definir los objetivos del sistema propuesto. La idea es sintetizar los requerimientos preliminares del sistema propuesto.

5.- Identificar alternativas. Se trata de recopilar ideas que satisfagan los requerimientos del usuario.

6.- Evaluar beneficios y costos. La tarea es comparar beneficios versus costos para determinar la factibilidad económica de cada alternativa de solución.

7.- Preparar el reporte del estudio de factibilidad. En este punto se hace un informe a la dirección para comunicar los resultados del estudio de factibilidad.

8.- Poner la documentación del estudio de factibilidad en un repositorio.

1.5.13.1.3 Definición de requerimientos.

El objetivo de esta fase es definir los requerimientos físicos y funcionales del sistema propuesto. Se divide en diez tareas:

1.- Revisar la información del estudio de factibilidad.

2.- Conducir entrevistas de definición de requerimientos. La tarea es ampliar los datos recolectados durante el estudio de factibilidad.

3.- Definir los requerimientos del sistema orientados a objetos. Se trata de definir las funciones que se quiere realice el sistema.

4.- Definir objetos. El objetivo es identificar cada objeto especificado en el requerimiento.

5.- Definir requerimientos de cliente/servidor. La idea es definir los requerimientos para establecer el ambiente cliente/servidor que soportarán las demandas funcionales y de arquitectura del sistema. Entre los elementos a definir están:

a) Hardware del cliente.
b) Software del cliente.
c) Hardware del servidor.
d) Software del servidor.
e) Administración de datos del servidor.

6.- Definir los requerimientos de servicio de la red. Esta tarea implica definir el servicio que se le demandará a la red de cómputo en tres direcciones:

a) Tiempo de respuesta.
b) Capacidad de salida.
c) Confiabilidad y disponibilidad necesarias.

7.- Definir los requerimientos de implantación. Aspectos físicos que se tienen que satisfacer: desempeño, ambiente, organizacionales y de desarrollo.

8.- Ubicar alternativas. Proponer alternativas reales para diseñar el sistema.

a) Software, redes y hardware para desarrollo.
b) Contactar proveedores comerciales para conseguir apoyo o equipo en caso necesario.
c) Analizar el potencial de cada alternativa.

9.- Establecer criterios para evaluar alternativas. La meta es definir las reglas para seleccionar vendedores de cómputo.

10.- Hacer el modelo del requerimiento. El objetivo es construir un modelo que plasme los requerimientos establecidos durante la definición, considerando los aspectos orientados a objetos y cliente/servidor.

11.- Documentar los requerimientos en un repositorio.

1.5.13.2 Diseño.

El diseño se divide en dos fases: externo e interno.

1.5.13.2.1 Diseño externo.

Esta fase contempla la transición del modelo de requerimientos a un conjunto de modelos de diseño orientados al usuario. Las diez tareas que comprende, están enfocadas a involucrar al usuario para asegurar que se satisficieron las demandas.

1.- Revisión de los requerimientos de datos.

2.- Definir subsistemas. El objetivo es diseñar los subsistemas y asignar tareas y objetos a cada subsistema.

3.- Definir las funciones cliente/servidor y asignar las tareas a los procesadores.

a) Determinar el tipo de procesamiento.
b) Definir las funciones que desarrollarán el cliente.
c) Definir las funciones que desarrollará el servidor.

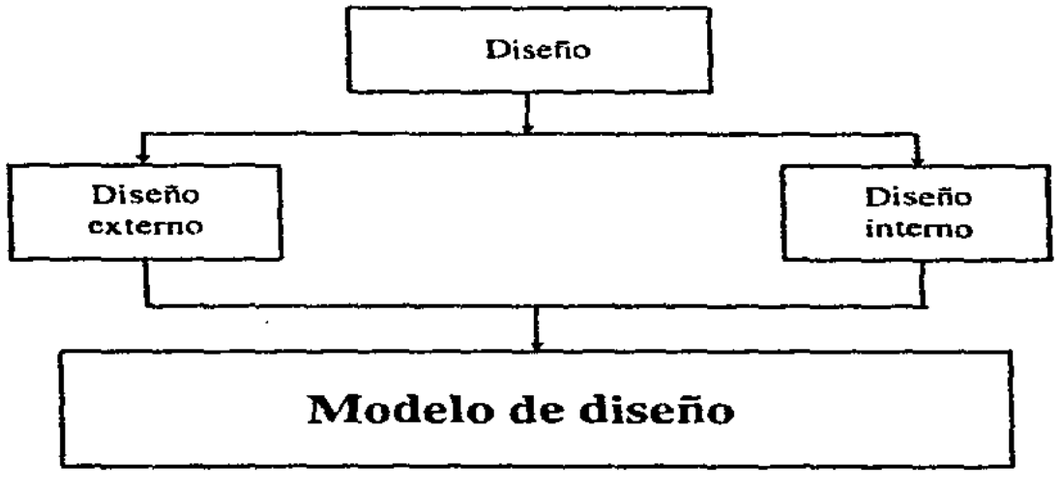


Fig. 1.5.13.2.1.1 El proceso de diseño.

4.- Definir las interfaces de los subsistemas. La tarea es ubicar el límite entre los sistemas independientes y los módulos que requerirán servicios de comunicación.

5.- Definir sistemas de control de seguridad.

6.- Definir las restricciones de diseño físicas, organizacionales y operacionales. Las limitaciones pueden ser de:

a) Interfaz.
b) Hardware.
c) Comunicaciones.

7.- Revisar iterativamente el modelo de requerimientos y de diseño. La función de esta parte es revisar cada elemento del requerimiento y del diseño para asegurar que todo esté contemplado dentro del modelo de sistema. La iteración se debe dar en tres direcciones:

a) Modelo de objetos.
b) Especificaciones de subsistemas.
c) Especificaciones de interfaz.

8.- Evaluar alternativas de diseño. Se debe comprobar si se satisfizo los requerimientos de:

a) Desempeño.
b) Datos.
c) Ambiente.
d) Organizacionales o de negocio.

9.- Concluir el modelo de diseño preliminar. Esta parte se concreta con la elaboración de un modelo de diseño preliminar, considerando los requerimientos orientados a objetos y de cliente/servidor. Los primeros muestran la distribución del comportamiento de los objetos y cómo se agrupan para formar

subsistemas. Los segundos ilustran cómo se almacenan las tareas de los subsistemas para clientes y servidores.

Ambos se fusionan para integrar el modelo de diseño preliminar para implantar el sistema cliente/servidor orientado a objetos. Las salidas de estas tareas son descripciones narrativas de alto nivel de todos los subsistemas y esquemas que muestran la distribución de las funciones orientadas a objetos y cliente/servidor.

10.- Documentación del diseño externo.

1.5.13.2.2 Diseño interno.

Establece la organización para la implantación y pruebas del sistema. En esta fase se crean documentos que especifican precisamente cómo los elementos funcionales definidos durante el diseño externo se transformarán en una arquitectura física para un sistema cliente/servidor orientado a objetos. Es necesario crear todas las especificaciones necesarias para codificar los programas. Esta fase se compone de diez tareas.

1.- Revisar los datos del diseño externo.

2.- Diseñar la arquitectura del sistema. El objetivo es diseñar los elementos técnicos que se tienen que integrar en el diseño de la arquitectura del sistema:

a) Sistema orientado a objetos.
b) Cliente/servidor.
c) Compartición de recursos.
d) Bases de datos relacionales.
e) Procesamiento distribuido.

3.- Concluir el diseño de objetos en el ambiente de desarrollo:

a) Definir estructuras de clases.
b) Especificar algoritmos para operaciones de las clases.
c) Optimizar las especificaciones de los objetos para mejorar el desempeño.
d) Iterar especificaciones de las clases para herencia.
e) Especificar procedimientos para implantar relaciones.
f) Preparar especificaciones para instancias.

4.- Especificar estructuras de datos. Aquí se deben concluir las especificaciones de datos para proveer al sistema de entradas y salidas.

5.- Diseñar la base de datos. Especificar la base de datos que se requerirá para desarrollar el diseño del sistema transformando a diseño de base de datos:

a) Objetos simples.
b) Objetos compuestos.
c) Asociaciones.
d) Objetos híbridos.

6.- Especificar consideraciones de diseño especiales.

7.- Concretar los procedimientos de control manual:

a) Entrada/salida.
b) Usuario.
c) Acceso.
d) Mantenimiento de archivos.
e) Recuperación de archivos.
f) Prueba.

8.- Definir planes de implantación y pruebas para:

a) Extraer componentes de objetos reutilizables.
b) Crear nuevos componentes de objetos.
c) Implantar el sistema cliente/servidor.
d) Instalar el DBMS.
e) Crear bases de datos.
f) Instalar las aplicaciones.
g) Probar la red, aplicaciones y el DBMS.

9.- Construir el modelo de diseño detallado. Considera los requerimientos de cómputo orientados a objetos y cliente/servidor. En la primera parte debe especificar los procedimientos para crear objetos y subsistemas. En la segunda debe contener los procedimientos para almacenar las funciones de los subsistemas en clientes y servidores. Con la unión de estas partes se llega al modelo de diseño detallado.

10.- Documentar el diseño detallado.

1.5.13.3 Implantación.

En esta fase los elementos especificados en el modelo de diseño se transforman en una arquitectura física para un sistema cliente/servidor orientado a objetos. Esta parte se subdivide en diez funciones:

1.- Revisar los datos del diseño interno.

2.- Instalar el sistema cliente/servidor:

a) Instalar el hardware y software del servidor de red.

b) Crear el ambiente de trabajo de la red.

3.- Instalar el sistema de administración de base de datos.

4.- Crear las bases de datos.

5.- Realizar los programas principales.

6.- Extraer los componentes de objetos reutilizables.

7.- Programar los nuevos componentes.

8.- Hacer las pruebas unitarias e integrales.

9.- Instalar la aplicación:

a) Instalar archivos en el cliente.
b) Instalar archivos en el servidor.

10.- Hacer pruebas de aceptación. El objetivo es comprobar que la aplicación cubre los requerimientos del sistema.

11.- Hacer el manual de usuario.

12.- Crear la documentación técnica.

1.5.14 Herramientas de desarrollo.

1.5.14.1 Servidores de bases de datos.

Tres de los manejadores de bases de datos más representativos y robustos del mercado son: Oracle, Informix y Sybase. Es por ello que son los que se incluyen en el presente análisis.

Oracle.

Esta base de datos corre sobre más hardware, software y redes que cualquier otra base de datos relacional. Esta particularidad lo identifica como un producto muy portable. Oracle es soportado por más del 70% de front-ends que hay en el mercado.

Informix.

Informix Online opera bajo sistemas operativos UNIX, proporciona alto performance y alta disponibilidad de información. Incluye características como disco espejo y mecanismos ágiles de recuperación. Es por ello que con frecuencia se utiliza en aplicaciones de misión crítica.

Sybase.

Microsoft se asoció con SYBASE en 1992, el SQL Server de Microsoft y el de SYBASE contenían las mismas características, con las pertinentes adecuaciones para OS/DOS y para UNIX respectivamente. Posteriormente se dividieron y actualmente cada firma posee su propia versión.

Algunas de las principales bondades de SYBASE son:

- Debido a que SYBASE es un DBMS robusto frecuentemente usado en grandes sistemas los cuales requieren gran velocidad de procesamiento.
- SQL Server es una arquitectura multihilos con su propio kernel que tiene integrado un monitor de control de SQL.
- Parte de SYBASE, Transact SQL es un conjunto de extensiones para ANSI-estándar SQL que es usado para escribir stored procedures. Adicionalmente, es posible construir triggers que se utilizan, entre otras cosas, para garantizar la importante propiedad de base de datos denominada integridad referencial.
- El producto integra funciones de seguridad que permiten a los administradores de sistemas, controlar el personal que tendrá acceso a los datos y el nivel de autoridad con que lo hará.
- Utiliza un Log de transacciones al cual se escribe cualquier modificación que sea objeto la base de datos.
- Como parte de su sistema de seguridad, SYBASE también proporciona la opción de generar discos espejos.
- El API de SQL Server permite a los desarrolladores de software construir aplicaciones cliente que trabajen con SQL Server y con una interfaz transparente. DB-Library, es un conjunto de funciones creadas en C que permiten realizar comandos de SQL para recuperar y actualizar datos. Las rutinas de DB-Library pueden ser usadas con C o Cobol. DB-Library es una biblioteca con ligas estáticas bajo DOS y ligas dinámicas bajo Windows y OS/2.
- SYBASE SQL Server tiene más herramientas de terceros que cualquier otra base de datos, por ello goza de popularidad y es considerada como una de las líderes.

Pruebas de desempeño.

A continuación tengo el resultado de las pruebas aplicadas a Oracle, Informix y Sybase.

Para evaluar estas bases de datos se utilizó el benchmark AS3AP (ANSI-SQL Standard Scalable and Portable) para sistemas de bases de datos relacionales. [BUT94]

Las pruebas que se realizaron son:

Transacción aleatoria de escritura mezclada (Random Write Transaction Mix)

Accede seis tablas con transacciones de lectura y escritura, a su vez se llevan a cabo operaciones de borrado, inserción, selección y actualización al servidor de BD.

Los cinco tipos de ejecuciones son:

a) Se actualiza un campo entero en una tabla con 7 millones de registros usando un operador between mediante la llave primaria.

b) Operación join entre dos tablas con 1 millón de registros.

c) Actualización de un campo entero en una tabla de 1000 renglones además del almacenamiento en una tabla vacía.

d) Actualización de una tabla con 2 millones de registros con valores definidos.

e) Mueve 5 millones de registros uno por uno a una tabla vacía.

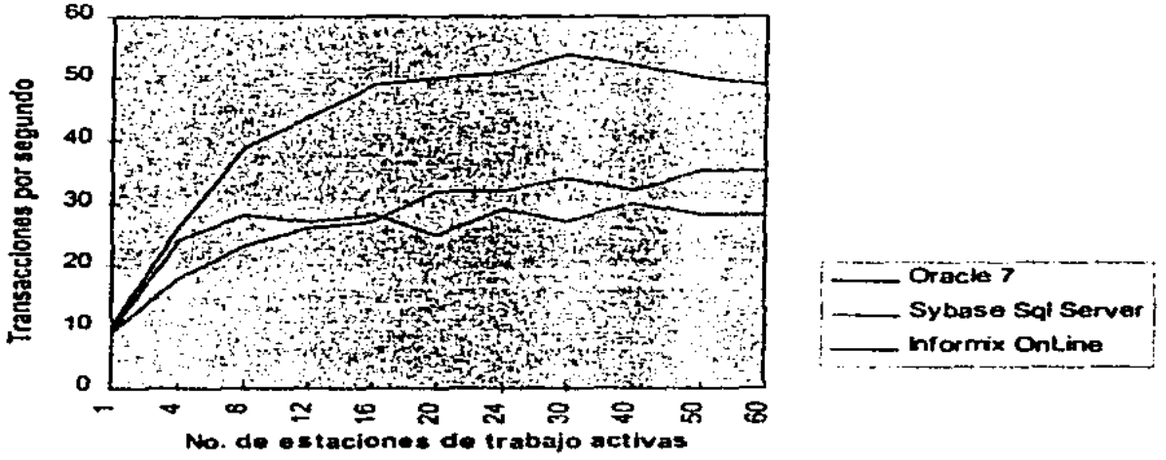


Figura 1.5.14.1.1 Transacción aleatoria de escritura mezclada.

La base de datos ganadora fue Oracle, después Sybase y al final Informix. Resaltando que Sybase usó stored procedures accediendo mediante llamadas a RPC's. Informix por su parte utilizó bloqueos a nivel registro.

Lectura aleatoria única (Single Random Read).

Se refiere a la lectura sencilla de un registro mediante la llave primaria, mostrando el número máximo de recuperaciones concurrentes que los sistemas puedan manipular. Se debe seleccionar un renglón aleatoriamente de una tabla única, repitiendo el proceso a la mayor velocidad posible de la BD.

El resultado más favorable le correspondió a Sybase SQL Server que usó indices cluster y stored procedures. Oracle en segundo lugar, operó manteniendo un cursor el cual se puede compartir a través de múltiples clientes.

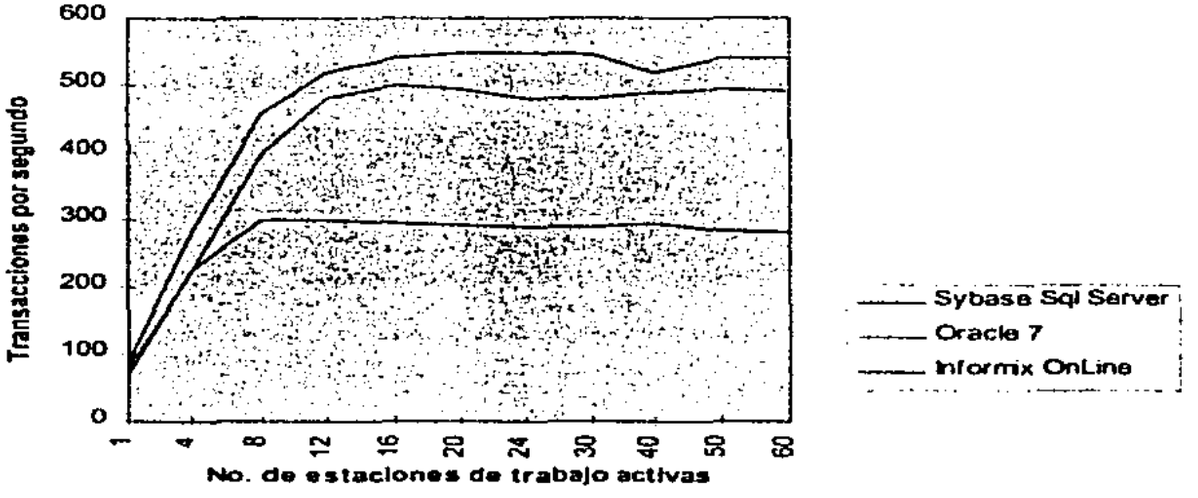


Figura 1.5.14.1.2 Lectura aleatoria única.

Transacción aleatoria de lectura mezclada (Random Read Transaction Mix).

En esta fase se acceden 5 tablas de la base de datos, para simular una carga de trabajo mixta de consultas de sólo lectura. El objetivo es dificultar la capacidad de recuperación de datos. Cada estación aleatoriamente seleccionó y ejecutó una serie de queries de un conjunto de 5 tipos.

Las características de las consultas son: La primera es una lectura de registro mediante la llave primaria. Después es un join sobre la llave primaria entre dos tablas con un millón de registros. La tercera es una selección (select) de una tabla con 7 millones de registros usando valores

definidos. En cuarta posición tenemos un join entre una tabla con un millón de registros y otra con 2 millones usando un operador between como restricción y un join sobre un campo carácter. El quinto query es otro join entre dos tablas con uno y cinco millones de registros respectivamente.

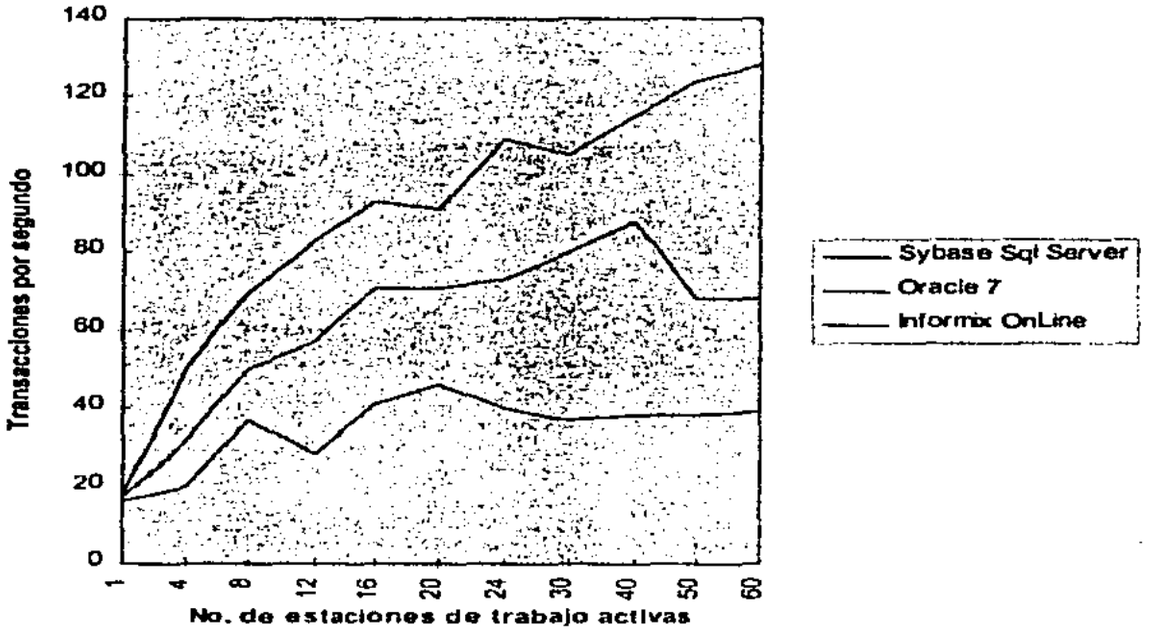


Figura 1.5.14.1.3 Transacción aleatoria de lectura mezclada.

Sybase SQL Server se colocó a la cabeza usando nuevamente índices cluster y procedimientos almacenados por medio de RPCs, Oracle e Informix llegaron en ese orden.

Recuperación de BLOBs (Binary Large Objects).

Un BLOB es una estructura utilizada para almacenar imágenes. La prueba mide qué tan rápido el cliente puede recuperar grandes estructuras y la forma en que la base de datos explota la red.

Los manejadores en cuestión ofrecen métodos para la obtención de las imágenes como BLOBs en una única llamada a la red. En este caso se empleó una tabla con 5000 imágenes diferentes en formato GIF entre 20 y 150 Kb de tamaño por cada una.

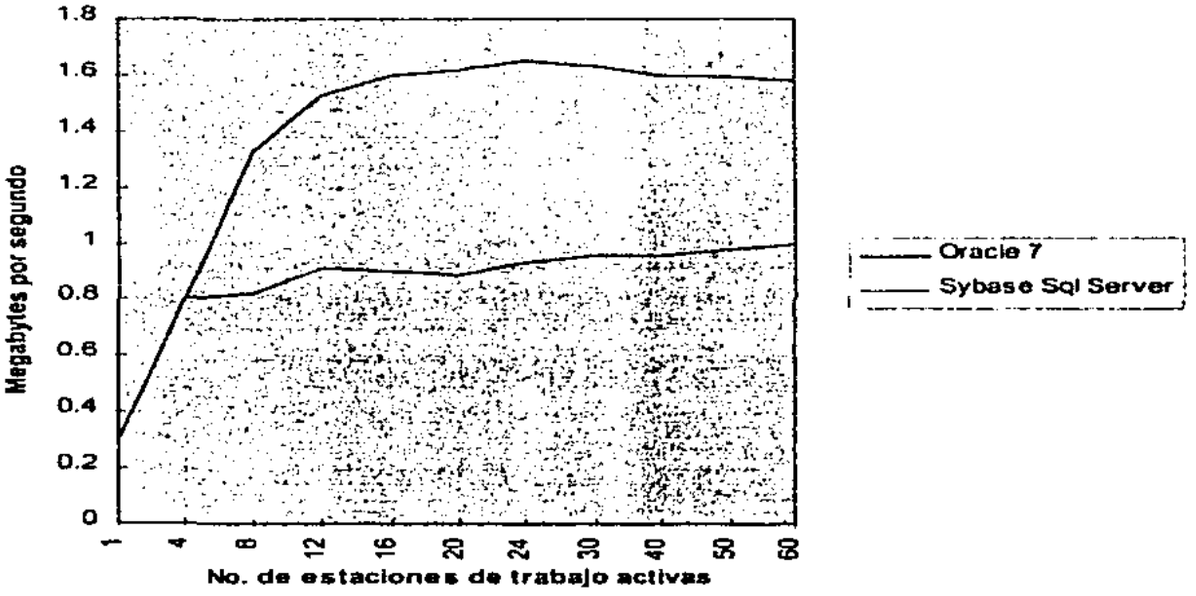


Figura 1.5.14.1.4 Recuperación de BLOBs.

Los promedios de transferencia máxima fueron:

Oracle, 1.61 Mbps.

Sybase SQL Server, 1 Mbps.

Informix, sin resultados.

Consultas ad hoc.

Consiste de 34 queries que utilizan funciones como: selects, joins, proyecciones, adiciones, clasificaciones y subqueries. El objetivo es ubicar al producto en el ámbito del apoyo a la toma de decisiones. Al final: Oracle a la delantera, Sybase en segundo e Informix en tercero.

Cargar e indexar.

Se cargan tablas consecutivamente para importar 18.11 millones de registros y crear 33 índices.

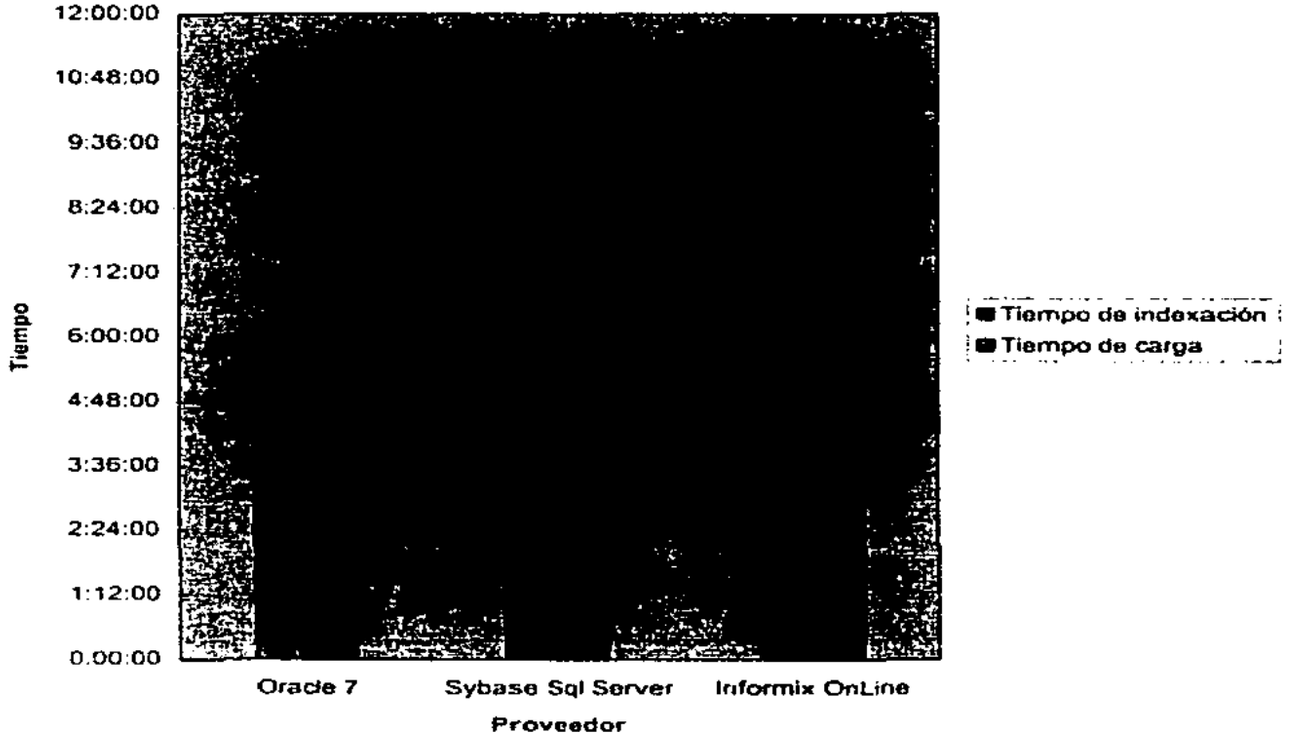


Figura 1.5.14.1.5 Cargar e indexar.

Como se aprecia en la gráfica: Oracle, Sybase e Informix, fueron las posiciones.

Exportar.

La prueba mide qué tan rápido una base de datos puede exportar una tabla con un millón de registros a un formato texto y delimitando cada campo con una coma de ASCII a un disco local del servidor.

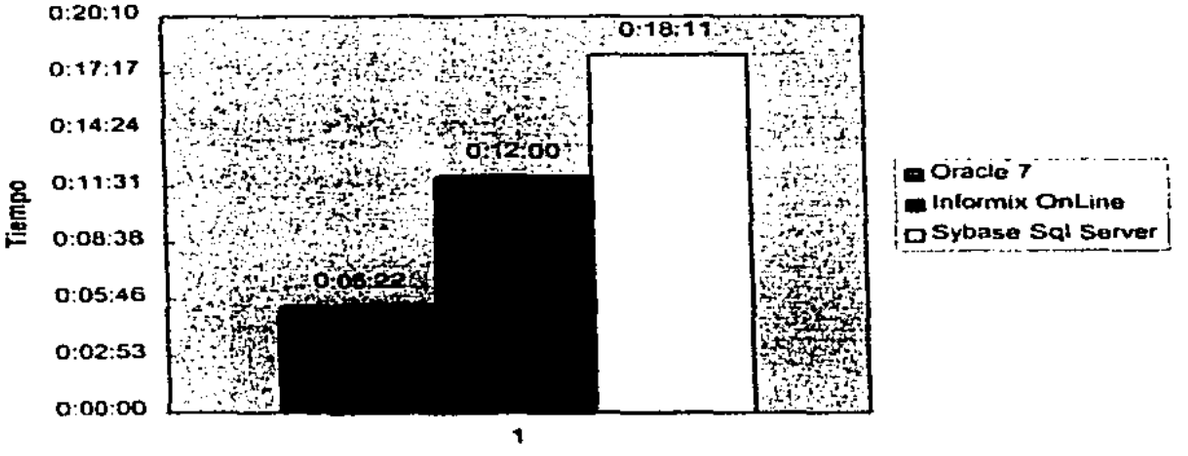


Figura 1.5.14.1.6 Exportar.

Primera ocasión que cambian las posiciones, Oracle ganador, Informix en segundo sitio y Sybase al final.

Conclusión final.

De acuerdo al estudio realizado el mejor manejador de base de datos relacional es Oracle. Sin embargo, Sybase que es con el que se cuenta en el centro de cómputo, obtuvo muy buenos resultados con una distancia pequeña del primero.

1.5.14.2 Herramientas de desarrollo de clientes o front-ends.

Los aspectos que se consideran para hacer la evaluación son:

. Sistema operativo.

. Interfaz utilizada.

. Experiencia de uso de la herramienta.

. Capacidad de creación de ejecutables.

. Soporta multiplataforma.

. Soporte técnico ofrecido en el país.

. Solvencia financiera de la empresa fabricante.

Delphi 2.0 de Borland.

Requerimientos del sistema:

Procesador 386, 8MB de RAM, 35MB de espacio en disco duro, unidad de CD-ROM para su instalación, Windows 95 o Windows NT.

Utiliza como lenguaje de programación Object Pascal, el cual es enseñado casi en cualquier universidad, lo que facilita su uso sin mucha experiencia. Delphi es la primera herramienta en ofrecer un alto desempeño en código nativo compilado con rapidez de ejecución y con la capacidad de accesar a bases de datos para cliente/servidor. Habilita una alta productividad, ya que permite reutilizar buena parte del código logrando un producto sumamente competitivo.

Desventajas.

Quizá la principal desventaja de Delphi es la problemática financiera que enfrenta Borland. En adición, el ambiente low end client lo que lo hace poco competitivo en el desarrollo de aplicaciones de gran tamaño.

Utilerías extras disponibles con el producto.

Un kit de desarrollo para un servidor de base de datos local, el reporteador Report Smith SQL, herramientas para desarrollo en equipo y un constructor de consultas visuales.

Servidor de DB SQL local.

Incluye Interbase, que es una versión individual del Engine SQL de Borland.

Reporteador.

Report Smith es un producto adquirido por Borland con el objeto de ofrecerlo junto a sus herramientas de desarrollo, Delphi no fue la excepción.

Interfaz con el programador CUA.

Obviamente se ajusta a la interfaz de Windows con controles tridimensionales, resultando en un ambiente de trabajo muy intuitivo.

Modo de programación.

Maneja el concepto de "lenguaje de dos vías", en el cual la programación visual va junto a la programación normal, de forma que existen 2 formas diferentes pero equivalentes de desarrollar una aplicación. Dependiendo de la parte de la aplicación que se esté trabajando se usa una u otra.

Lenguaje de programación.

Trabaja con Object Pascal que tiene un alto desempeño en el desarrollo visual. Incluye Exception Handling, información a tiempo de corrida, y métodos de tablas virtuales. Object Pascal es una variante de Pascal, que es altamente compatible con Borland Pascal 7. Incorpora las características de programación de un lenguaje 4GL y el desempeño y potencia de un lenguaje 3GL.

Tipo de código generado.

Genera ejecutables completamente autónomos de 16 ó 32 bits sin necesidad de intérpretes de DLL's o runtimes, lo que resulta en una gran velocidad.

Programación orientada a objetos.

Soporta los puntos básicos de la POO, además de aspectos avanzados como la excepción de errores, característica que lo hace uno de los más completos en el mercado.

Programación orientada a eventos.

Se tiene la facilidad de crear nuevos eventos en clases creadas por el programador, además de soporte a todos los eventos de Windows

Biblioteca o repositorio de componentes.

Cuenta con una Librería de Componentes Visuales de 32 bits (VCL32) incluyendo código fuente que ya tiene implementados aproximadamente 100 y se pueden ir añadiendo componentes creados por el desarrollador.

Distribución de aplicaciones. Crea ejecutables autónomos brindando amplia facilidad de distribución de aplicaciones.

Capacidades de interconexión.

La versión Client/Server cuenta con drivers nativos para los más conocidos DBMS del mercado, así como conectividad a través de IDAPI, el estándar de conectividad de Borland. Por otro lado es posible conectarse a un servidor por medio de ODBC.

Facilidad de migración.

El concepto de alias hace posible que no se necesite ni recompilar el código para migrar de una fuente local a una remota.

Desarrollo de aplicaciones con interconexión.

Soporte para OLE servidor y cliente, con soporte para automatización, DDE (Dynamic Data Exchange), DLLs (Dynamic Link Libraries), VBX (Visual Basic Controls) y OCX (OLE Custom Control).

Visual Basic 4.0 de Microsoft.

Requerimientos del sistema.

Procesador 486, 8MB de RAM (16MB de RAM para Windows NT), 20MB a 80MB de espacio en disco, unidad de CD-ROM para instalación y Windows 95 o Windows NT (para la versión de 32 bits).

Plataformas de desarrollo soportadas.

Su uso se reduce para plataforma de PC. El run-time de Visual Basic 3 consiste en un conjunto de DLL'S, los cuales van de 350Kb a 2Mb, dependiendo de lo que requiera el programa.

Ventajas.

Fue el producto que abrió brecha a la programación en ambiente Windows. Como consecuencia es el más popular en el mercado. Tiene una cantidad importante de add-ons y librerías, ya sea de Microsoft o de terceros.

El ser un producto Microsoft asegura una perfecta integración con Windows y solidez financiera para el futuro. El dialecto de Basic que utiliza es simple de aprender y de usar. El tamaño y requerimientos reducidos ayudan a programar rápidamente sin requerimientos de hardware altos. La mayor parte de los demás productos ofrecen, aunque sólo parcialmente, un camino de migración de Visual Basic a ellos. Los controles VBX, que se empezaron a usar con Visual Basic, son ahora el estándar del mercado.

Desventajas.

Está orientado para aprender programación en Windows, no para desarrollo de aplicaciones cliente/servidor. Es por ello que sólo tiene facilidades básicas de debug y distribución de aplicaciones, y carece por completo de facilidades de control de versiones, programación en grupo y otras características avanzadas de desarrollo.

Visual Basic tiene un desempeño bajo y características inferiores a cualquier producto posterior nativo de PC. Esto incluye los controles VBX o el uso de muchas DLL's diferentes como run-time. Basic, no es un lenguaje poderoso para creación de aplicaciones industriales.

Utilerías extra disponibles con el producto.

Viene con el engine de Access 1.1 con muchos problemas, ODBC, un ejemplo de una aplicación de distribución, soporte para Pen for Windows, Mapi y telecomunicaciones. Incluye además una versión crippleware pequeña de Crystal Reports for Visual Basic con problemas de operación con servidores de bases de datos remotos. Tiene información y ejemplos sobre cómo crear controles VBX con Microsoft Quick C.

Servidor de BD SQL local.

Access responde a comandos de consulta de SQL, pero sin el desempeño de éstas.

Reporteador.

El Crystal Reports para Visual Basic es un reporteador simple que permite el desarrollo de reportes básicos para bases de datos Access, Paradox, Xbase y Btrieve. Permite el acceso de otros tipos de manejadores o de bases de datos remotas por medio de ODBC, con un rendimiento pobre.

Interfaz con el programador.

CUA.

Se ajusta a la interfaces de Windows, con controles 3-D como agregados en la versión profesional, siendo la base de las interfaces de la mayoría de los productos actuales.

Modo de programación.

Visual Basic es el primer lenguaje de programación visual, en el cual se le da una forma visible a cada una de las partes de la aplicación, y en donde el código está limitado a estos componentes. La distribución y apariencia de los componentes toman precedencia a la creación de un código orientado a eventos delimitado a ellos.

Tipo de lenguaje de programación.

Se considera al lenguaje un dialecto de Basic.

Tipo de código generado.

Es un código compactado y verificado por sintaxis, el cual es interpretado a tiempo de corrida, lo que produce menor eficiencia, velocidad y detección de errores, aunque hace más sencillo el desarrollo.

Programación orientada a objetos.

No es un producto muy fértil para explotar las bondades de la programaión orientada a objetos. Sólo utiliza la sintaxis de objeto componente.

Programación orientada a eventos.

Se reduce a los eventos y parámetros predefinidos en Visual Basic, y carece de respuesta a eventos dentro de métodos.

Biblioteca o repositorio de componentes. Incluye una librería controles VBX.

Distribución de aplicaciones.

Se complica por el gran número de DLL's que deben ser incluidos y la carencia de control de versiones.

Método de conexión a servidor SQL.

La edición profesional incluye ODBC versión uno con drivers para Microsoft SQL Server y Oracle 6.

Facilidad de migración.

Es necesario modificar el código y recompilar. Así mismo, la sintaxis del SQL de Access presenta diferencias con el SQL estándar.

Desarrollo de aplicaciones con interconexión.

Soporte para OLE 2.0, a nivel servidor y cliente, con soporte para automatización, DDE, DLL's, y VBX's . Existen herramientas para operar con el software de oficina de Microsoft.

Programación en grupo. No soportada.

Aplicaciones de terceros. Existe un número importante de ellas.

Ingeniería de software asistida por computadora. No existen para Visual Basic.

Escalabilidad. No existe.

Manejo de gráficos. Se realizan usando un control VBX incluido, y con herramientas de terceros.

Powerbuilder 5.0 de PowerSoft.

Requerimientos del sistema.

Procesador 486 o superior, MS-DOS versión 3.3 o mayor, 12 MB de RAM, Microsoft Windows 3.x, Windows NT o Windows 95; 32 MB de espacio en disco duro y unidad de CD-ROM para instalación.

Plataformas de desarrollo soportadas.

PowerBuilder es uno de los mejores front-end que existe en la actualidad, ya que permite el desarrollo de aplicaciones robustas, bajo ambientes multi plataforma, cuenta con un optimizador de código, posibilidad de distribución de objetos en un ambiente de red y drives nativos para diferentes bases de datos.

PowerBuilder posee un soporte completo para ambientes Windows de 16 y 32 bits en plataformas Intel, incluyendo Microsoft Windows 3.x, Windows NT, Windows95, Win OS/2, Mac y UNIX (Motif y Open Look).

Ventajas.

Sybase compró hace poco a PowerSoft, por lo que la versión 5.0 provee de herramientas y drivers mejorados para este producto. Además, permite el fácil desarrollo y distribución de aplicaciones en varias plataformas a nivel corporativo.

Desventajas.

Tiene un ambiente de programación diferente del normal y causa incertidumbre su futuro al haber cambiado de dueño.

Utilerías extra disponibles con el producto.

Posee una familia de herramientas de desarrollo escalable que incrementan la productividad de las aplicaciones. La serie incluye PowerBuilder Enterprise, PowerBuilder Team/PDBS, PowerBuilder Desktop, Infomaker y PowerBuilder Library for Lotus Notes.

Servidor de BD SQL local.

Viene con el engine más popular en el mercado, Watcom SQL con la limitante de base de datos de 5 Mb.

3GL. Al incluir Watcom C++ permite hacer módulos externos y ligarlos a la aplicación.

Reporteador.

Su reporteador Infomaker, goza de versatilidad permitiendo múltiples tipos de reportes sin demasiada programación. Las consultas se realizan a través de un constructor gráfico y un quickselect multitabla. Sólo hay que salvar los consultas como objetos y entonces usarlos como fuentes de datos para una gran variedad de reportes.

CUA. No se ajusta demasiado a los estándares de Windows.

Modo de programación.

Es capaz de definir, compilar y corregir una clase integradas de C++ basadas en el compilador Watcom C/C++ de alto rendimiento, aparte del 4GL nativo que incluye.

Lenguaje de programación.

Se ofrece un extenso lenguaje orientado a objetos con miles de funciones. Los desarrolladores pueden escribir sus propias funciones o utilizar las ya existentes escritas en C o en otros lenguajes. Han sido incluidos un compilador manejador y un debugger con muchas particularidades positivas.

Tipo. Lenguaje 4GL similar a C++.

Tipo de Código Generado. P

Programación orientada a objetos.

Incluye toda la gama de programación orientada a objetos, pero en Watcom C++, y en 3GL. PowerBuilder soporta la definición de clases para modelados visuales y objetos no visuales. Además, también provee soporte para otras características de la POO, incluyendo herencia, encapsulamiento de datos y procesos lógicos, y polimorfismo. Estas capacidades aseguran consistencia en aplicaciones, incrementando la productividad y minimizando costos. Es posible usar ventanas de PowerBuilder, menús y objetos creados por los usuarios para definir objetos ancestro con atributos de encapsulamiento, eventos y funciones, pudiendo heredar las características.

Programación orientada a eventos.

Se basa en los eventos de Windows en su totalidad.

Biblioteca o repositorio de componentes.

Incluye una biblioteca de objetos centralizada y un administrador de código fuente, además, una aplicación de administración de configuración e interfaces para los más populares programas de administración de versiones de terceros.

Distribución de aplicaciones. La desventaja es que hay que incluir runtimes.

Programación en grupo. Soporta desarrollo en grupo.

Método de conexión a servidor SQL.

El Watcom SQL ODBC driver que soporta conexiones con bases de datos Watcom SQL o de PowerBuilder. Este driver es multi-tier, el cual procesa funciones de ODBC pero manda las instrucciones SQL dependiendo de la base de datos usada para que se procesen. De esta forma el código está separado del lenguaje de transacción, optimando la programación y el aprendizaje de la herramienta

Facilidad de migración.

Si se utiliza en un principio Watcom SQL como servidor, es fácil migrar a otro servidor SQL, ya sea de Watcom u otros.

Control del back-end. Se pueden crear bases de datos desde PowerBuilder.

Desarrollo de aplicaciones con interconexión.

Soporte para OLE 2.0, a nivel servidor y cliente, con soporte para automatización, DDE, DLLs, VBXs y COX.

Aplicaciones de terceros.

Ingeniería de software asistida por computadora.

Existe un CASE denominado ERWIN que tiene una versión especial diseñada para PowerBuilder.

3GLs.

Cuenta con un lenguaje de programación incluido, el Watcom C++.

Escalabilidad.

El CODE (Client/Server Open Development Enviroment) expande la tecnología de los productos de PowerSoft que cubren varios aspectos como son llamadas remotas a procedimientos y procesos de transacciones de modelo de datos y pruebas automatizadas.

Manejo de gráficos.

Maneja gráficos de dos y tres dimensiones, de pastel, de barras, columnas, líneas, scatter y gráficas de área.

Capítulo II

Planteamiento del problema

2.1

Investigación inicial

2.1.1 Planteamiento del problema.

La Facultad de Derecho de la UNAM requiere realizar un proceso cada semestre conocido como inscripciones. El objetivo de este proceso es asignar materias a todos los alumnos que las soliciten, de acuerdo a las reglas establecidas por la propia institución asentadas en el Reglamento General de Inscripciones y en las disposiciones del H. Consejo Técnico de la Facultad de Derecho.

2.1.2 Objetivo del proyecto.

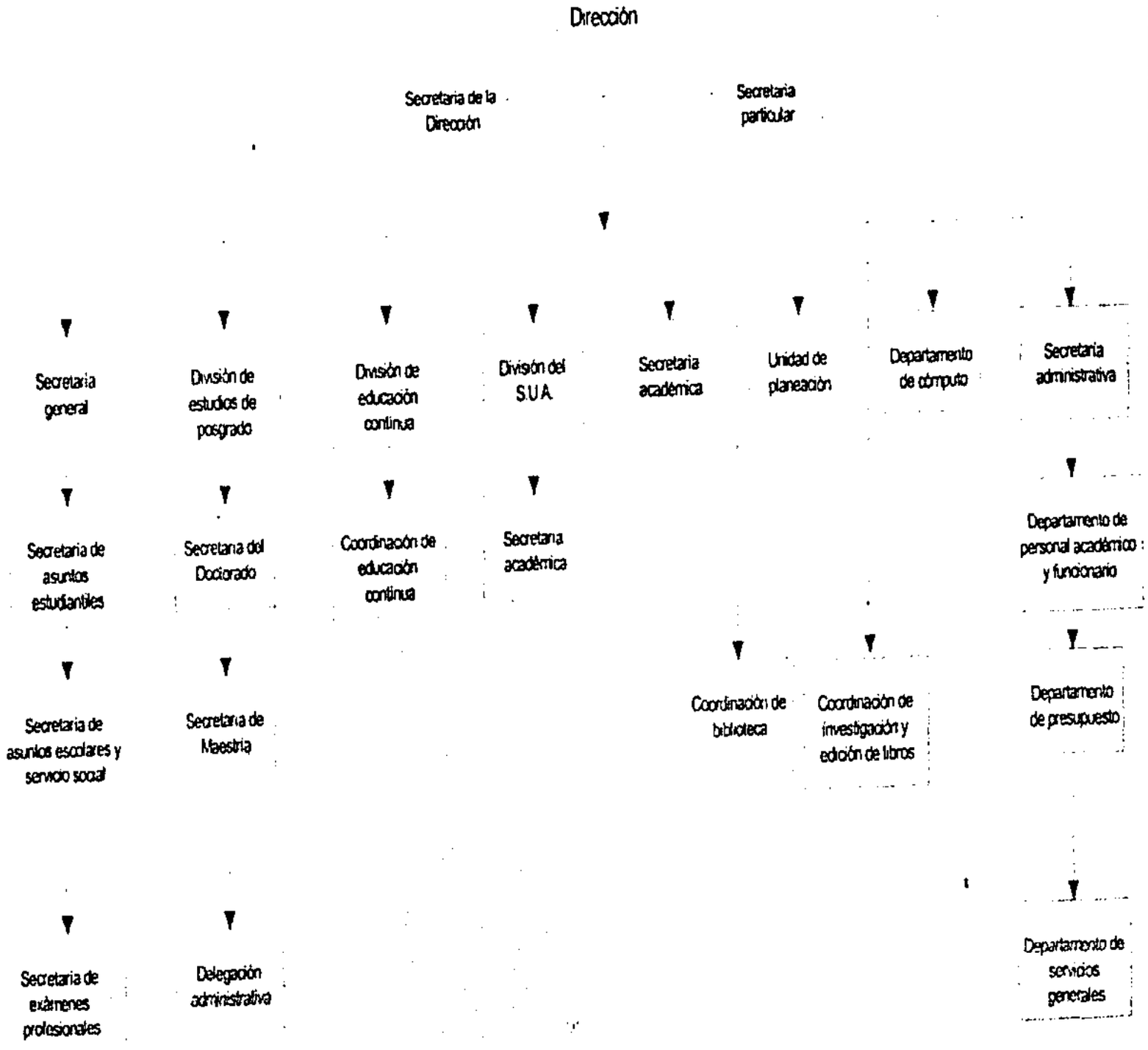
Analizar, diseñar e implantar un sistema de inscripciones para la Facultad de Derecho de la UNAM, de acuerdo a los lineamientos que se deriven de un análisis y mejora del proceso actual y de las características que demanden los clientes de la propia institución.

2.2

Organigrama de la Facultad de Derecho de la UNAM

El organigrama de la Facultad nos permite entender cómo está organizada la institución donde residirá la solución. En este caso, el desarrollo se realiza en el centro de cómputo inmerso en el departamento del mismo nombre. Los clientes de la solución son la Secretaría General y la Secretaría de Asuntos Estudiantiles.

Organigrama de la Facultad de Derecho de la UNAM



2.3

Estudio de factibilidad

2.3.1 Recopilación de expectativas.

El objetivo es superar las expectativas de los clientes y el primer paso para ello es conocerlas a fondo y con objetividad. Es por ello que esta parte del desarrollo es el punto de partida fundamental, y de lo que aqui se descubra dependen los lineamientos del proyecto mismo.

2.3.1.1 Preparación de las entrevistas.

El público de la Facultad de Derecho que se considera adecuado para obtener información acerca de las necesidades a resolver, los lineamientos que debe seguir la solución y los beneficios que traerá consigo son:

a) El Secretario de asuntos escolares de la facultad, quien es el responsable del proceso.
b) El jefe del centro de cómputo de la facultad, el cual es el responsable técnico de las inscripciones.
c) Alumnos de la propia facultad, que constituyen los clientes de las inscripciones.
d) El personal que opera el sistema actual.
e) Personal técnico del centro de cómputo que le da soporte al sistema.

Contenido del cuestionario.

1.- ¿ Cuál es la principal problemática que representa actualmente el proceso de inscripciones en la facultad ?

2.- ¿ Qué aspectos conservaría del proceso y del sistema de inscripciones actual ?

3.- ¿ Qué características debiera tener el proceso y el sistema de inscripciones en la facultad ?

4.- ¿ Qué características relevantes introduciría en el proceso de inscripciones de la facultad que posean en otras dependencias ?

5.- ¿ Qué beneficios obtendría usted directamente de un nuevo sistema para el proceso de inscripciones en la facultad ?

6.- ¿ Que características debiera contemplar un sistema de inscripciones para soportar el proceso en el mediano y largo plazo ?

(Sólo para alumnos)

7.- ¿ Cuál es el tiempo promedio de espera en filas ?

8.- ¿ Qué porcentaje de sus solicitudes se satisface ?

2.3.1.2 Aplicación de las entrevistas.

Los alumnos que elegí para las entrevistas fueron tomados totalmente al azar, de modo que evitara desviaciones en las respuestas a consecuencia de consultas a grupos específicos. El resto de los entrevistados no son anónimos.

El contenido de cada entrevista se encuentra en el Apéndice 1.

2.3.1.3 Análisis de la información recopilada.

2.3.1.3.1 Problemas.

1.- El hecho de que el periodo de inscripciones sea de 5 días afecta la academia en la medida que los días de clase se ven disminuidos.

2.- No hay suficiente oferta de profesores para los grupos que se necesitan.

3.- Del total de profesores que imparten clase, el 20 % no entregan actas a tiempo.

4.- No existe información actualizada con la frecuencia necesaria, de la movilidad que van teniendo los cupos de los grupos conforme se van dando las inscripciones durante el día, lo que ocasiona que los alumnos formen de nuevo sus grupos en la misma ventanilla.

5.- El tiempo que los alumnos tienen que estar formados para inscribirse oscila entre 2 y 3 horas, y el 60 % logran inscribirse en su primera opción.

6.- La tira de materias se pierde con frecuencia al momento de mandarla imprimir.

7.- El sistema actual se inhabilita de 1 a 10 minutos por día.

8.- Existen dos periodos de extraordinarios al final del semestre, época poco apropiada para dedicarle tiempo a estudiar. Como consecuencia, el 90 % de los alumnos no aprueban en examen extraordinario.

2.3.1.3.2 Conclusiones.

1.- Es bueno conservar el proceso de inscripción en línea asignando un horario de atención objetivo basado en el promedio y avance en créditos de cada alumno.

2.- Se debe procurar que el proceso de inscripciones tome los menos días posibles ganando más días de clase, con la intención de favorecer a la academia.

3.- Sería deseable contar con mecanismos para procesamiento local de las actas de calificaciones como lectores ópticos, que evitaran depender de DGAE, aunque posteriormente existiera un mecanismo de validación.

4.- Es necesario contar con mecanismos de información de la movilidad de grupos con una frecuencia alta, v.gr. cada 10 minutos. Con esto se lograría tener informados a los alumnos para que revisaran su selección de asignaturas y grupos, minimizando los "cuellos de botella" derivados del tiempo que consumen los alumnos armando de nuevo los horarios en la misma ventanilla.

5.- El sistema debe seguir vigilando que se cumplan los reglamentos de inscripción de la UNAM y de la facultad en particular.

6.- Se debe idear un proceso eficiente y confiable de impresión y entrega de tiras de materias.

7.- El nuevo sistema debe diseñarse de tal modo que tenga flexibilidad para adaptarse a los cambios al plan de estudios y al reglamento de inscripciones.

8.- Se debe soportar el sistema por una verdadera y sólida base de datos que permita tener un buen desempeño, ir creciendo con la matrícula de la facultad y adaptarse a los cambios del medio y tecnológicos.

9.- No se conoce otro proceso de inscripciones fuera de la facultad.

2.3.2 Análisis de la situación actual.

Antes de entrar de lleno al sistema de inscripciones actual, se detallan brevemente los procedimientos administrativos, que caracterizan las inscripciones en la Facultad de Derecho.

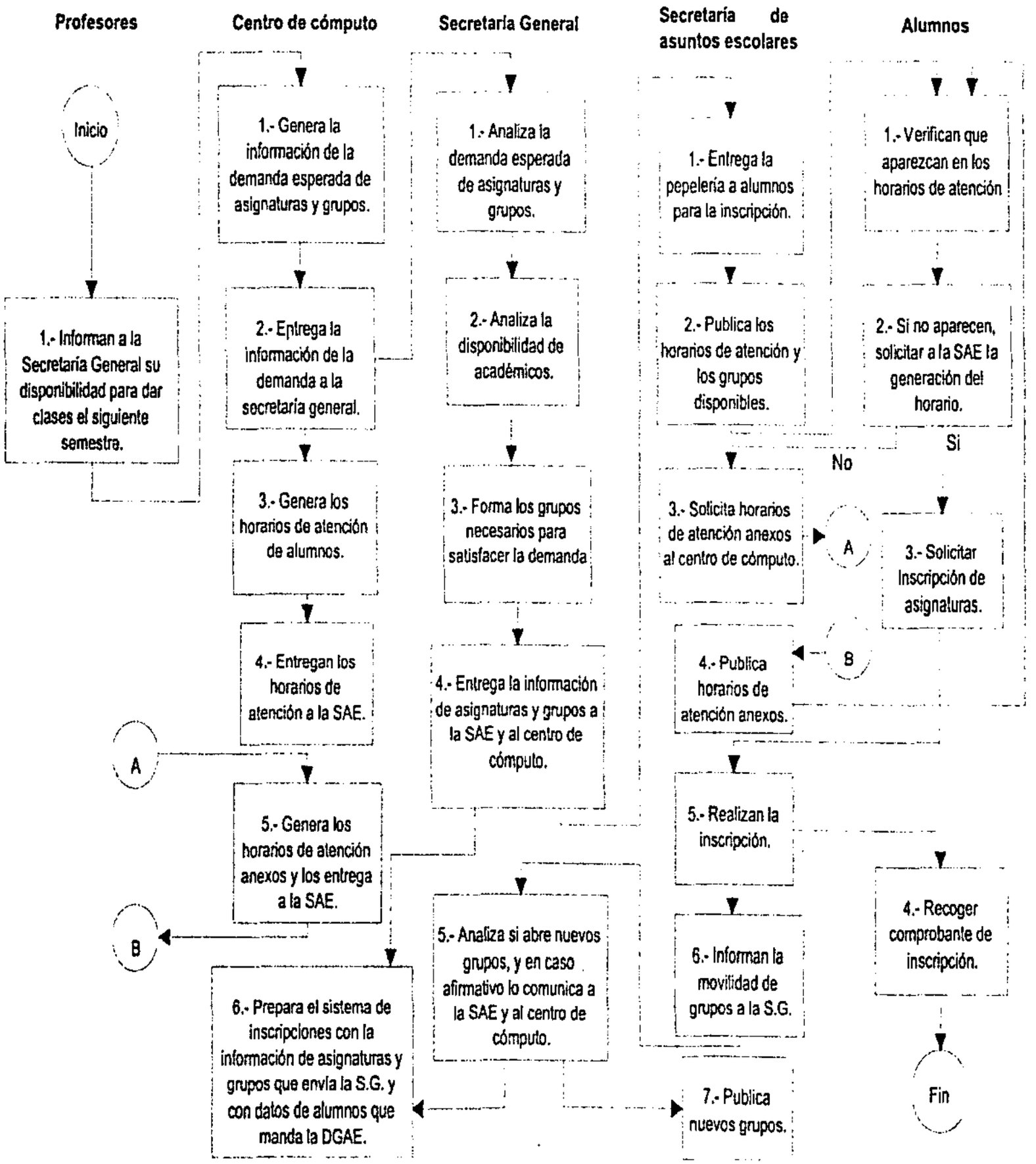
Conviven dos planes de estudios conocidos como "viejo" y "nuevo" plan. El segundo plan entró en vigor a partir del semestre 91-1. Los alumnos sólo pueden cursar materias del plan que les corresponda, pero no de ambos. La Secretaría General de la Facultad forma varios grupos con cupo de 90 alumnos para cada materia en los dos planes; mismos que se publican previamente al inicio de cada semestre para que los alumnos tengan conocimiento de las materias, profesores y horarios disponibles.

Para las materias de nuevo ingreso, se arman 16 grupos divididos 8 en el turno matutino y 8 en el vespertino. En estos casos el cupo es de 100 espacios, para permitir a los alumnos de tercer semestre recursar materias del primero.

La Secretaría de Servicios Escolares publica previamente al período de inscripciones, el horario de atención del alumno de acuerdo a su promedio y a su avance en créditos, y entrega la documentación necesaria para llevar a cabo la inscripción.

Deben acudir en el horario asignado, a las ventanillas de la facultad con los formatos de inscripción prellenados. En ese preciso momento se realiza la inscripción en línea en las materias que soliciten, validando que cumplan los requisitos que fijan los reglamentos. En la carrera de Derecho existe seriación para ambos planes.

Diagrama del proceso de inscripciones



Esquema tecnológico actual.

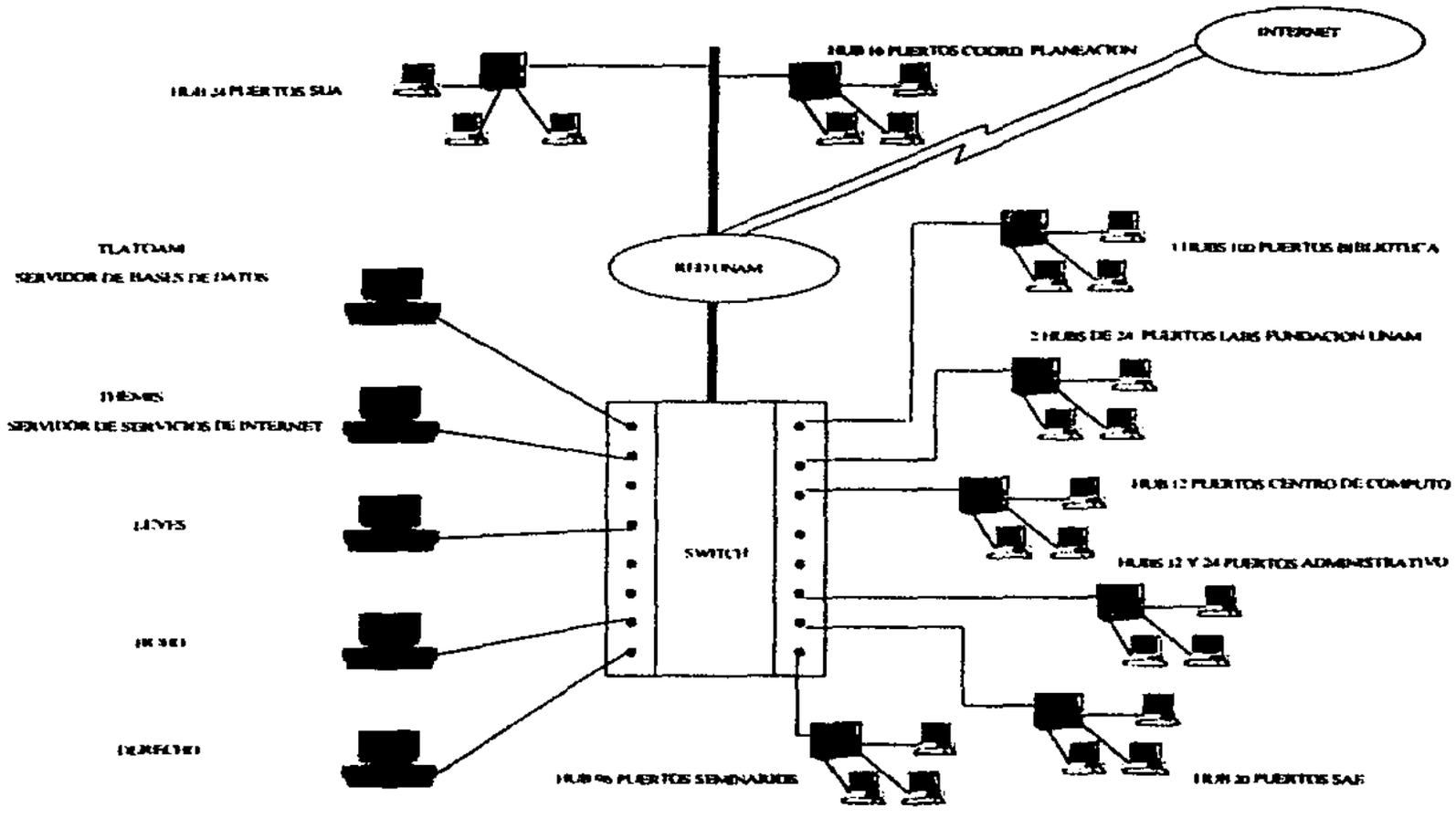


Figura 2.3.2.1 Esquema tecnológico actual.

Como se puede observar, se tiene un backbone principal de fibra óptica del cual se desprenden conexiones a los hubs y un switch general encargado de distribuir la carga, ambos conectados a la red UNAM. La idea es que las peticiones de la computadora se vayan al Hub respectivo y de ahi al switch, estando en este punto se puede hacer el acceso a cualquier servidor conectado.

Se tienen 8 redes de área local y dos servidores UNIX, con capacidad para conectar a 372 computadoras personales.

Los servidores UNIX (Tlatoani y Themis) utilizan el sistema operativo SOLARIS 2.5. Estos brindarán servicio de bases de datos y de Internet respectivamente. Del resto de los servidores, Derecho tiene el S.O. Netware 3.12 mientras que Leyes y Buho el Netware 3.11.

Gracias a que se tiene una topología en estrella, los accesos serán a mayor velocidad y la carga de trabajo está balanceada.

Capítulo III

Alternativas de solución

3.1

Descripción del método

Ahora, con la idea clara que se tiene del problema y las expectativas de nuestros clientes, se pueden plantear tres opciones viables que satisfacen sus demandas y escoger la que cumple en mejor medida sus necesidades.

El método de evaluación de opciones consiste en plantear varios aspectos clave para la elección de una solución y asignarles un peso en porcentaje, que depende de la importancia que tiene para la institución el punto en particular. Por otro lado, cada una de las soluciones se califica en esos aspectos en la escala bueno, regular y malo, correspondiéndoles una ponderación de 100, 50 y 0 respectivamente. Finalmente, se multiplican los factores obtenidos y se obtiene un subtotal por aspecto. La suma de todos los subtotales da la calificación final. La alternativa ganadora es aquella que sume mayor puntuación en la escala 0-100.

Los aspectos clave que se considerar en el análisis de las opciones y su peso en porcentaje son:

Aspecto	Peso en porcentaje (%)
Funcionalidad	20
Seguridad	20
Desempeño	20
Infraestructura de cómputo requerida vs. disponible.	10
Recursos humanos capacitados en la institución para llevar a cabo el desarrollo.	10
Capacidad de crecimiento.	5
Tiempo de desarrollo.	5
Costo de la solución.	5
Complejidad de mantenimientos.	5
Total	**100**

Desde esta perspectiva, a continuación se presentan las tres opciones viables encontradas y finalmente un cuadro comparativo entre ellas.

3.2

Generar un gran mantenimiento al sistema de inscripciones actual

3.2.1 Descripción.

Esta opción consiste en mejorar el sistema actual de inscripciones para que se adapte a las nuevas demandas de la institución y liberarlo de los múltiples errores que presenta en su operación.

En términos de los programas actuales, esta adaptación consiste en optimizar el código de los programas existentes para disminuir los altos tiempos de respuesta que se presentan; incluyendo además, las modificaciones necesarias para tener un buen funcionamiento del sistema en la red local, dado que constantemente se queda fuera de servicio con los alumnos esperando; y como tercer rubro, incorporar todas las validaciones del reglamento general de inscripciones y disposiciones internas de la facultad que le hacen falta.

3.2.2 Ventajas.

La principal ventaja es que no se requiere inversión alguna dado que se cuenta con el software que necesita esta solución para su desarrollo y para operar. Por la parte de hardware, demanda muy pocos recursos por lo que prácticamente cualquier computadora con las que se cuenta actualmente exceden los recursos necesarios.

3.2.3 Desventajas.

El lenguaje no cuenta con facilidades para hacer una aplicación muy funcional ni agradable al usuario. El desempeño de las aplicaciones hechas en Paradox con un volumen importante de datos, como es el caso, es muy lento. No posee un esquema de seguridad efectivo, ya que se reduce la protección a nivel archivos.

El tiempo de desarrollo sería alto puesto que el ambiente no permite con facilidad lo contrario y se necesita que el personal involucrado en el desarrollo se capacite en Paradox para DOS que hoy es obsoleto.

3.3

Actualizar la versión de Paradox a la 7.0 para Windows

3.3.1 Descripción.

Esta alternativa se refiere a hacer un nuevo sistema de inscripciones utilizando la versión 7.0 de Paradox que trabaja en el ambiente gráfico Windows.

Esta solución implica ciertas caracteristicas minimas para el equipo de cómputo de desarrollo y para el de ejecución. Se requerirían computadoras pentium con 30 Mb de espacio en disco duro disponible y 16 Mb de memoria RAM, y por la parte de software, Windows 95 para trabajar eficientemente. Adicionalmente, se requeriría comprar Paradox 7.0 con 2 licencias para desarrollo y una licencia de uso para 15 usuarios. Las máquinas que ejecutarán el sistema requieren las mismas dimensiones con la opción de 8 Mb en memoria.

3.3.2 Ventajas.

Se logra mayor seguridad para la información, al contar con claves para las tablas.

Se obtienen todos los beneficios mencionados en el Capitulo 1, de trabajar con interfaces gráficas, tanto en desarrollo como en uso.

3.3.3 Desventajas.

La seguridad sigue dejando mucho que desear, puesto que la protección es a nivel archivo. El tiempo de respuesta se vuelve muy alto para volúmenes grandes de datos. Esta combinación de Netware y Paradox ocasiona que el tráfico en la red sea muy alto dado que las peticiones son de archivos, no nada más de datos. Finalmente, existe desembolso para adquirir las licencias de desarrollo y uso de Paradox 7.0.

3.4

Creación de un nuevo sistema de inscripciones bajo un ambiente cliente/servidor

3.4.1 Descripción.

La tercer opción considera la generación de un sistema nuevo bajo un ambiente cliente/servidor soportado por un robusto manejador de base de datos, corriendo sobre un potente servidor, y con una interfaz cliente amigable en computadoras que permitan una interacción satisfactoria con los usuarios.

En este caso me refiero a tener la información centralizada administrada por Sybase SQL SERVER v. 11, como DBMS, corriendo en un servidor con amplia capacidad de procesamiento. Todo montado sobre el sistema operativo UNIX del cual se tienen referencias muy satisfactorias de seguridad y velocidad.

Los clientes tendrían una aplicación creada en PowerBuilder 5.0 con interfaz gráfica Windows que es la más común, y ejecutándose sobre los equipos que hoy se tienen, los cuales van desde los 486 con 4 Mb de RAM, hasta los pentium con 16 Mb de memoria.

Es recomendable usar los manejadores de red propietarios de Sybase conocidos como dblibrary, que brindan más velocidad.

3.4.2 Ventajas.

Se puede satisfacer prácticamente cualquier demanda de funcionalidad de los clientes al contar con un potente DBMS y un producto de desarrollo que pertenece a la misma compañía (PowerSoft), con la seguridad de compatibilidad que brinda esta pertenencia.

Se tiene todos los elementos para garantizar una alta seguridad a los datos y gracias al DBMS y al sistema operativo UNIX, se lograría un excelente desempeño con altos volúmenes de información, como los que la facultad maneja.

No habría inversión ni en software ni en hardware, puesto que el centro de cómputo de la facultad adquirió Sybase SQL Server v. 11 para 32 conexiones simultáneas que incluye los manejadores de red nativos que lo conectan con el cliente. La Dirección General de Administración Escolar donó una licencia de desarrollo de PowerBuilder Enterprise 5.0.

Por la parte de hardware, se tiene una estación de trabajo SUN modelo UltraSparc con procesador RISC y sistema operativo Solaris que es la versión UNIX de SUN. Corre a una velocidad de 143 Mhz, posee 6 gigabytes de disco duro y 64 megabytes de memoria RAM.

Las máquinas que ejecutarían la parte cliente existen y cumplen con las características necesarias para un buen desempeño.

Gozaríamos de todas las ventajas de un ambiente gráfico en la parte cliente.

El tráfico en la red disminuiría drásticamente, dado que el cliente sólo envía su petición al servidor, el cual la procesa y le devuelve la respuesta especifica.

Existen varias redes interconectadas en la facultad, por lo que la parte de comunicaciones estaría resuelta al tener acceso a una de esas redes.

Al estar soportado por un verdadero DBMS y un lenguaje potente para el cliente, tendríamos amplias posibilidades de hacer crecer en el mediano plazo las capacidades del sistema y los mantenimientos no serían engorrosos, siempre y cuando el diseño sea flexible y amplio en cuanto a funcionalidad.

Adicionalmente a lo anterior, existe en la facultad un CASE que auxiliaría en el diseño y construcción de la parte del servidor.

3.4.3 Desventajas.

La única desventaja es la inexperiencia en el desarrollo de una aplicación cliente/servidor por parte del centro de cómputo de la facultad. Esto se hace extensivo tanto al manejador de base de datos como al lenguaje de desarrollo para la parte cliente. De cualquier forma, existe la formación suficiente de nuestra parte y de los recursos asignados para poder incursionar con las herramientas y crear una potente aplicación.

3.5

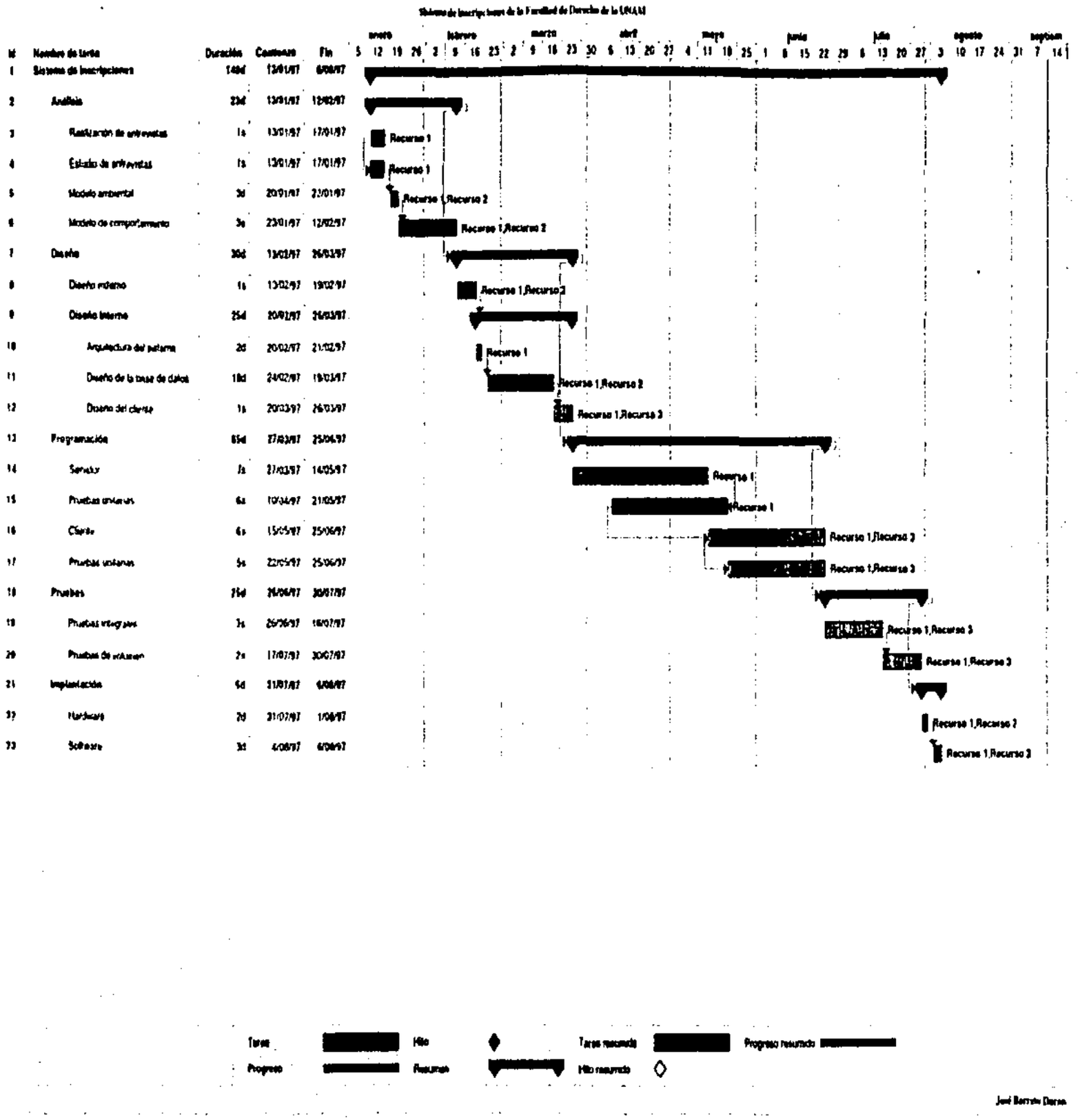
Elección de la mejor opción

3.5.1 Justificación.

A continuación muestro el cuadro comparativo entre propuestas, el cual fue construido de acuerdo a las consideraciones establecidas al inicio de este Capítulo.

Aspecto a evaluar	Generar un gran mantenimiento al sistema actual	Actualizar la versión de Paradox a la 7.0 para Windows	Creación de un nuevo sistema de inscripciones bajo un ambiente cliente / servidor
Funcionalidad (20)	50 (10)	50 (10)	100 (20)
Seguridad (20)	0 (0)	50 (10)	100 (20)
Desempeño (20)	0 (0)	0 (0)	100 (20)
Infraestructura de cómputo requerida vs. Disponible. (10)	100 (10)	50 (5)	100 (10)
Recursos humanos capacitados en la institución para llevar a cabo el desarrollo. (10)	50 (5)	50 (5)	50 (5)
Capacidad de crecimiento. (5)	0 (0)	50 (2.5)	100 (5)
Tiempo de desarrollo. (5)	0 (0)	100 (5)	50 (2.5)
Costo de la solución. (5)	100 (5)	50 (2.5)	50 (2.5)
Complejidad de mantenimientos. (5)	0 (0)	50 (2.5)	100 (5)
Total	**30**	**42.5**	**90**

Por un margen muy considerable y de acuerdo al análisis objetivo, es muy claro que la opción que más conviene es desarrollar un nuevo sistema de inscripciones bajo un ambiente cliente/servidor, con los lineamientos planteados.


Sistema de inscripciones de la Facultad de Derecho de la UNAM
enero
febrero
marzo
abril
mayo
junio
julio
agosto
septiembre
Id
Nombre de tarea
Duración
Comienzo
Fin
1 Sistema de inscripciones 148d 13/01/97 6/08/97
2 Análisis 23d 13/01/97 12/02/97
3 Realización de entrevistas 1s 13/01/97 17/01/97 Recurso 1
4 Estudio de entrevistas 1s 13/01/97 17/01/97 Recurso 1
5 Modelo ambiental 3d 20/01/97 23/01/97 Recurso 1,Recurso 2
6 Modelo de comportamiento 3s 23/01/97 12/02/97 Recurso 1,Recurso 2
7 Diseño 30d 13/02/97 26/03/97
8 Diseño externo 1s 13/02/97 19/02/97 Recurso 1,Recurso 2
9 Diseño interno 25d 20/02/97 26/03/97
10 Arquitectura del sistema 2d 20/02/97 21/02/97 Recurso 1
11 Diseño de la base de datos 18d 24/02/97 19/03/97 Recurso 1,Recurso 2
12 Diseño del cliente 1s 20/03/97 26/03/97 Recurso 1,Recurso 3
13 Programación 85d 27/03/97 25/06/97
14 Servidor 3s 27/03/97 14/05/97 Recurso 1
15 Pruebas unitarias 6s 10/04/97 21/05/97 Recurso 1
16 Cliente 6s 15/05/97 25/06/97 Recurso 1,Recurso 2
17 Pruebas unitarias 5s 22/05/97 25/06/97 Recurso 1,Recurso 3
18 Pruebas 75d 26/06/97 30/07/97
19 Pruebas integrales 3s 26/06/97 16/07/97 Recurso 1,Recurso 3
20 Pruebas de aceptación 2s 17/07/97 30/07/97 Recurso 1,Recurso 3
21 Implantación 6d 31/07/97 6/08/97
22 Hardware 2d 31/07/97 1/08/97 Recurso 1,Recurso 2
23 Software 3d 4/08/97 6/08/97 Recurso 1,Recurso 3
Tarea
Hito
Tarea resumida
Progreso resumido
Progreso
Resumen
Hito resumido
José Barrera Durán

3.5.3 Análisis costo/beneficio.

Para llevar a cabo este análisis lo primero que necesito es identificar los beneficios y costos que traerá consigo el proyecto. Una vez que esto se haya realizado, la ecuación para calcular la relación beneficio / costo es:

B/C = beneficios - costos

Si la relación anterior es mayor que cero indica que el proyecto es ventajoso en términos económicos.

Beneficios.

Los beneficios asociados al proyecto son:

1.- Se le dará un mejor servicio a los alumnos, al eliminarse las constantes caídas del sistema.

2.-Se tendrá un control centralizado de la información de inscripciones en una sola base de datos que es robusta y confiable.

3.- Se evitará la redundancia existente actualmente en la información de alumnos.

4.- La facultad tendrá la seguridad de que no se podrán hacer inscripciones de alumnos que hayan cometido alguna falta a los reglamentos de la universidad.

5.- Se tendrá un sistema más flexible que permitirá incorporar en poco tiempo los cambios que se presenten en las políticas de inscripción, internas o de la universidad en general.

6.- Se reducirán los tiempos invertidos actualmente por el personal técnico del centro de cómputo en preparar la información previa a las inscripciones de 5 días a 1 día.

Costos.

Los costos involucrados en el desarrollo y operación del sistema son:

1.- Costos de hardware:

a) Servidor: $0.00 (existente)
b) Elementos de red: $0.00 (existentes)
c) Computadoras clientes : $0.00 (existentes)

2.- Costos de software

a) Sistema operativo UNIX Solaris v. : $0.00 (existente)
b) Manejador de base de datos Sybase SQL Server v. : $0.00 (existente)
c) Lenguaje de programación Power Builder v. 5 : $0.00 (existente)
d) Ambiente gráfico Windows 95 para las máquinas cliente : $0.00 (existente)

3.- Costos de desarrollo

Para estimar este costo se multiplica el costo por mes-hombre de acuerdo al perfil por el tiempo involucrado en el proyecto. En este proyecto en particular, este costo resulta muy bajo puesto que dos de los tres recursos involucrados no tenemos sueldo asignado puesto que se trata de una persona que realiza su servicio social y del trabajo de tesis. Sólo hay el costo de un recurso que labora para el centro de cómputo con sueldo asignado. En este sentido el costo es:

Fase	No. de recursos	Tiempo en semanas	Costo por semana ($)	Costo por fase ($)
Realización de entrevistas.	1	1	0	0
Estudio de entrevistas	1	1	0	0
Modelo ambiental	1 1	0.6 0.6	0 1,000	0 600
Modelo de comportamiento	1 1	3 3	0 1,000	0 3,000
Diseño externo	1 1	1 1	0 1,000	0 1,000
Diseño interno	1 1	5 5	0 1,000	0 5,000
Programación	1 1	13 6	0 0	0 0
Pruebas	1 1	4 4	0 0	0 0
Total general				**$9,600.00**

Con la intención de colocar el costo en una perspectiva real, si se hubiese desarrollado el proyecto con personal ajeno a la universidad y con tarifas comerciales el costo sería de:

Fase	No. de recursos	Tiempo en semanas	Costo por semana ($)	Costo por fase ($)
Realización de entrevistas.	1	1	6,250	6,250
Estudio de entrevistas	1	1	6,250	6,250
Modelo ambiental	1	0.6	6,250	3,750
	1	0.6	6,250	3,750
Modelo de comportamiento	1	3	6,250	18,750
	1	3	6,250	18,750
Diseño externo	1	1	6,250	6,250
	1	1	6,250	6,250
Diseño interno	1	5	6,250	31,250
	1	5	6,250	31,250
Programación	1	13	5,000	65,000
	1	6	5,000	30,000
Pruebas	1	4	4,000	16,000
	1	4	4,000	16,000
Total general				**$259,500.00**

4.- Costos de operación

Para el uso del sistema se contrata por el tiempo que duren las inscripciones a alumnos de la propia facultad, conocidos como "voluntarios". Es por ello que este costo es para fines de la facultad cero pesos, dado que se piensa utilizar el mismo esquema.

Técnicamente dos personas del centro de cómputo se asegurarán de que el sistema no presente ningún problema durante los periodos de uso en el año. En este sentido el costo de operación se

estima, dividiendo su sueldo mensual entre 4 y multiplicándolo por el número de semanas que dure en operación, que es de 3. Así, este costo se traduce en:

Participación en semanas	Costo por semana en $	Costo total
3	1,000	3,000

De esta forma, tenemos que los beneficios están expresados en términos cualitativos y por consiguiente no se les puede asignar un valor directamente, pero es claro para todas las partes que la relación beneficio/costo del proyecto es ampliamente positiva.

Capítulo IV

Análisis y Diseño

4.1

Análisis

4.1.1 Modelo ambiental.

4.1.1.1 Descripción breve del propósito del sistema.

" El propósito del sistema de inscripciones es asignar materias a los alumnos que las soliciten en el periodo de inscripción, de acuerdo a las reglas establecidas por la universidad y por la propia facultad, asentadas en el Reglamento General de Inscripciones de la UNAM y en las disposiciones del H. Consejo Técnico de la Facultad de Derecho."

Responsabilidades.

Las responsabilidades fundamentales del sistema de inscripciones son:

a) Manejar las inscripciones y calificaciones históricas de los alumnos.
b) Permitir el manejo de calificaciones extraoficiales, con el propósito de acelerar el proceso de inscripciones.
c) Soportar la existencia de varias carreras, y de varios planes de estudio dentro de cada una de ellas.

d) Dividir el proceso de inscripciones en varios periodos, cada uno con sus propias características.
e) Controlar el adelanto de materias.

Exclusiones.

El sistema de inscripciones obedece a todas las exclusiones del sistema "Integra", mismo que describo a continuación.

El sistema de inscripciones de la Facultad de Derecho forma parte de un concepto más amplio denominado "Integra", cuyo objetivo es satisfacer de manera flexible todas las necesidades de la administración escolar de la Facultad de Derecho, en cuanto a explotación de información se refiere.

Responsabilidades.

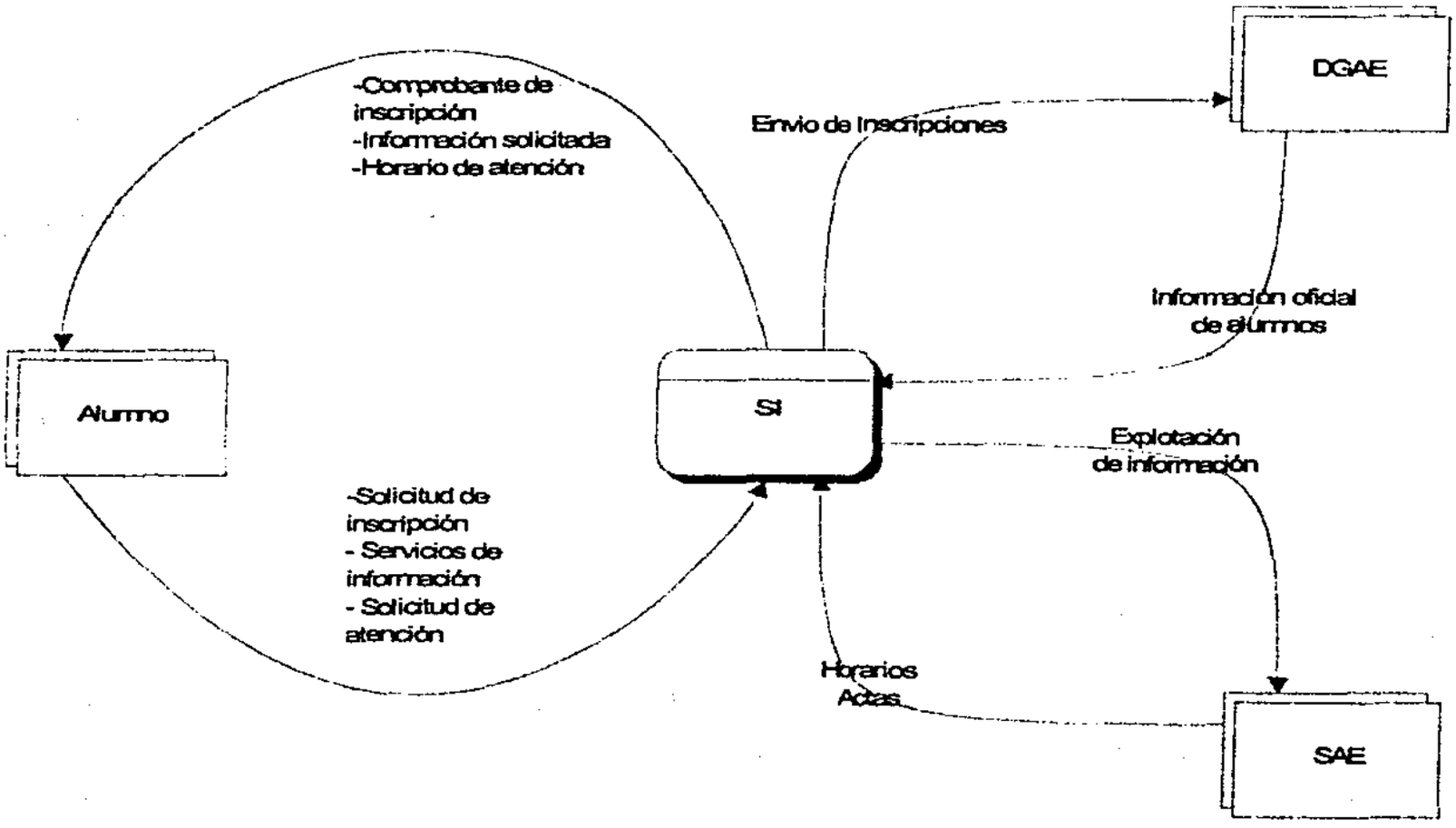
Las principales responsabilidades que se le han encomendado a Integra son:

a) Brindar un servicio que satisfaga a los alumnos.
b) Facilitar las operaciones de la Secretaria de Asuntos Escolares.
c) Controlar la información de la biblioteca y los laboratorios.

Exclusiones.

a) No trata de sustituir las funciones ejecutivas de la Secretaria de Asuntos Escolares, Coordinación de Biblioteca y/o Laboratorios, pero si lograr un alto grado de automatización de sus procesos.
b) No contempla la asignación de aulas y profesores en grupos.

4.1.1.2 Diagrama de contexto.



4.1.1.3 Lista de acontecimientos.

No.	Acontecimiento	Entidad externa
1	Solicitud de inscripción.	Alumno
2	Solicitud de horario de atención.	Alumno
3	Solicitud de información.	Alumno
4	Generación de bloques	Secretaría de Asuntos Escolares.
5	Solicitud de información diversa.	Secretaría de Asuntos Escolares.
6	Entrada de datos extraoficiales.	Secretaría de Asuntos Escolares.
7	Entrada de datos oficiales.	DGAE
8	Salida de datos oficiales.	DGAE

4.1.2 Modelo de comportamiento.

Como se comentó en el Capítulo 1, el modelo de comportamiento describe lo que se demanda del sistema para interactuar con el entorno. Lo conforman los elementos que se detallan a continuación.

4.1.2.1 Diagramas de flujo de datos.

A continuación se presentan los diagramas de flujo de datos del sistema en los diferentes niveles necesarios. Sin duda, es una herramienta de análisis muy poderosa que permite ubicar muy bien al sistema en el contexto en que funcionará, e incorporar desde este momento a todas las entidades necesarias.

Diagrama de flujo de datos general nivel 1

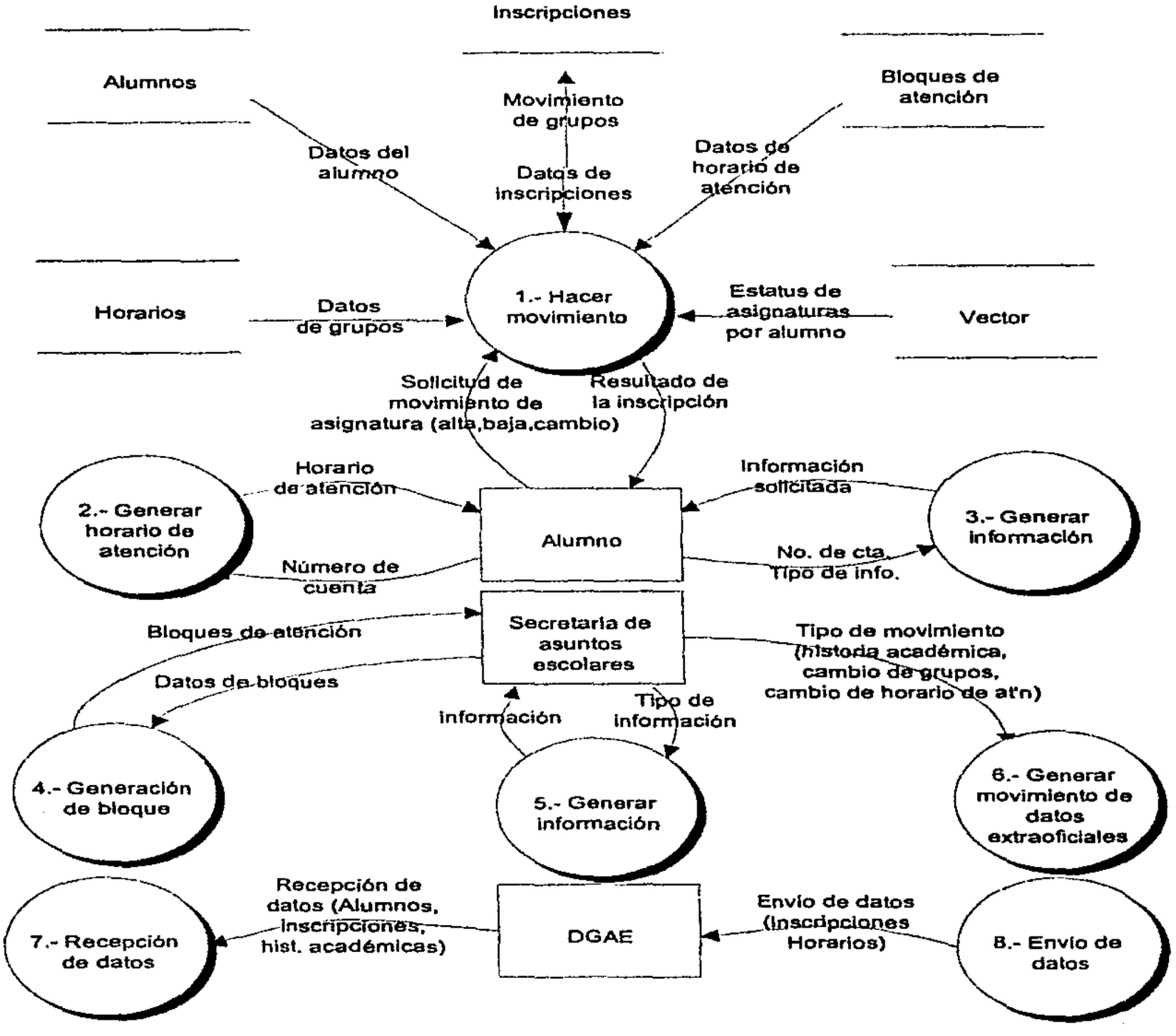


Diagrama de flujo de datos nivel 2 del proceso 1 "Hacer movimiento".

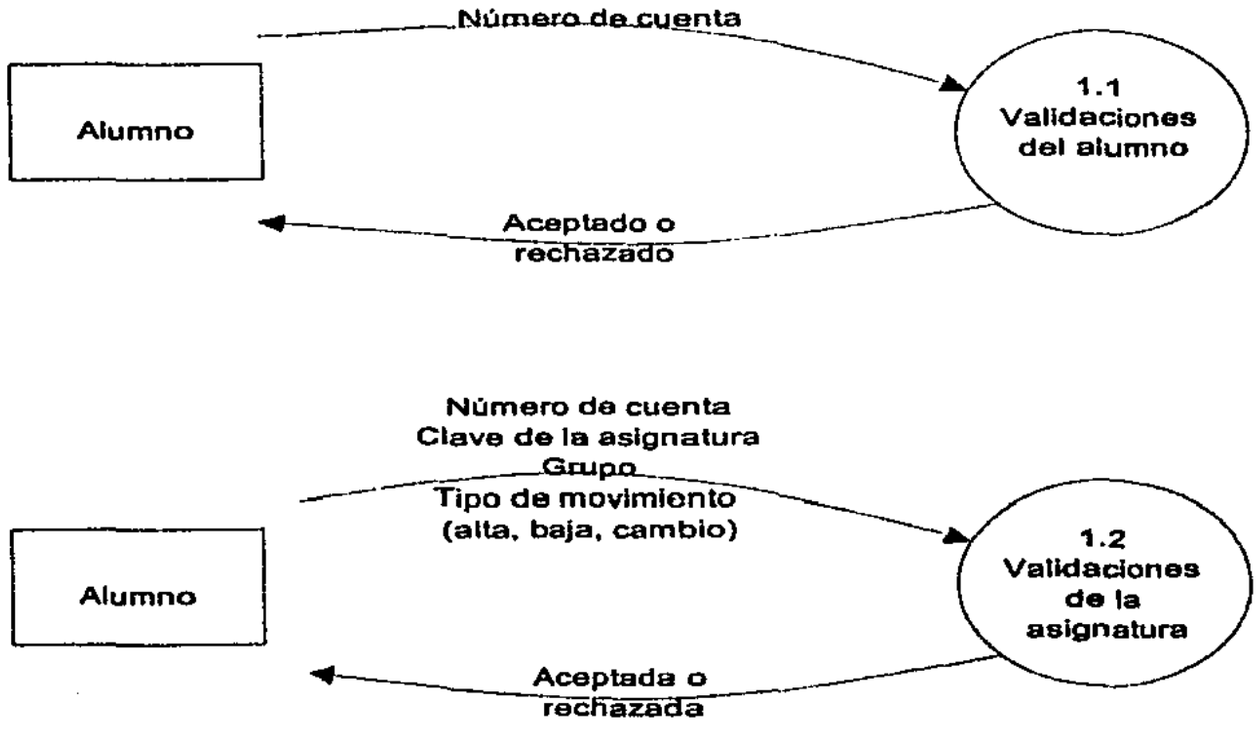


Diagrama de flujo de datos nivel 2 del proceso 2 "Generar horario de atención".

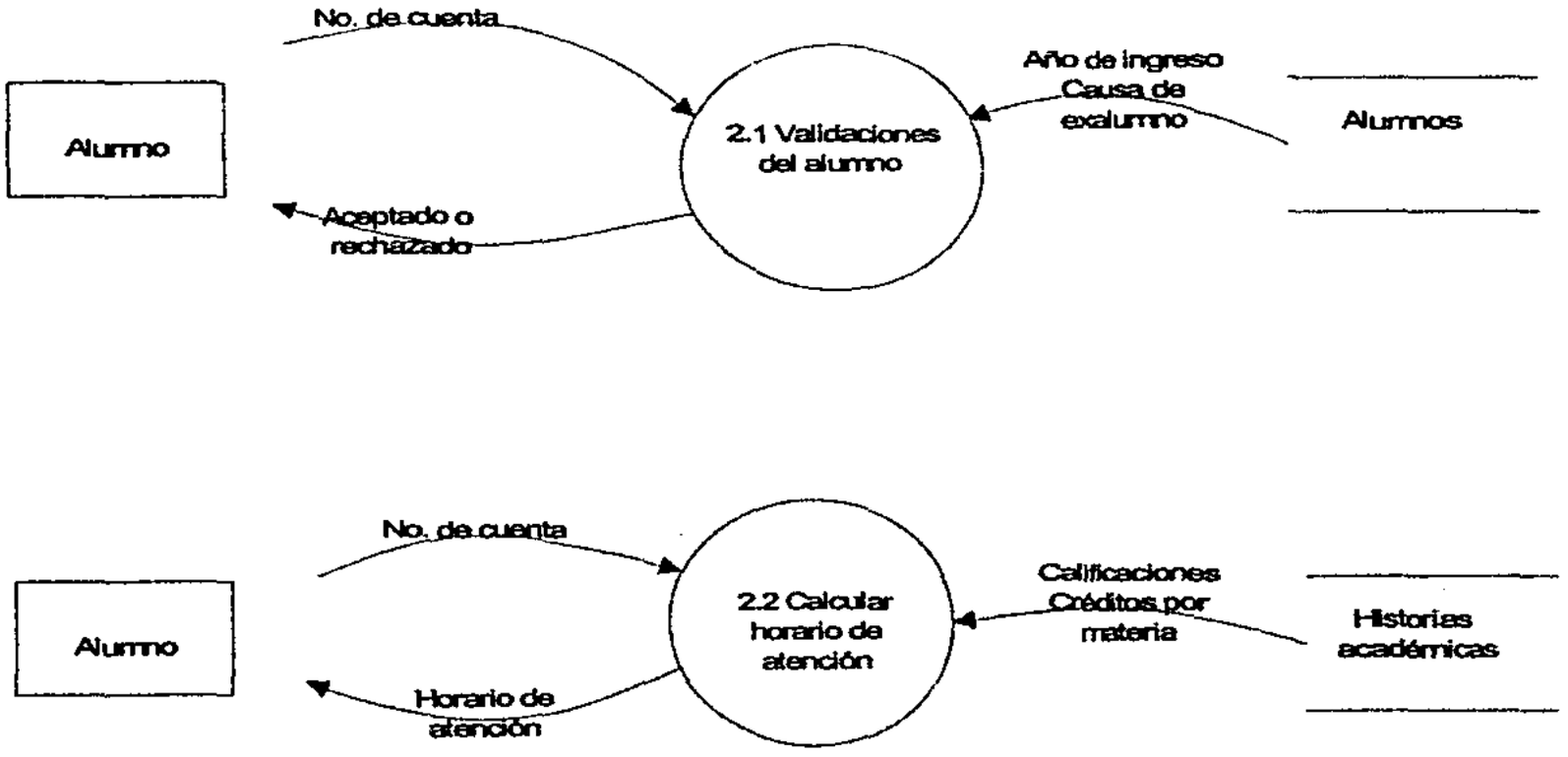


Diagrama de flujo de datos nivel 2 del proceso 3 "Generar información de alumno".

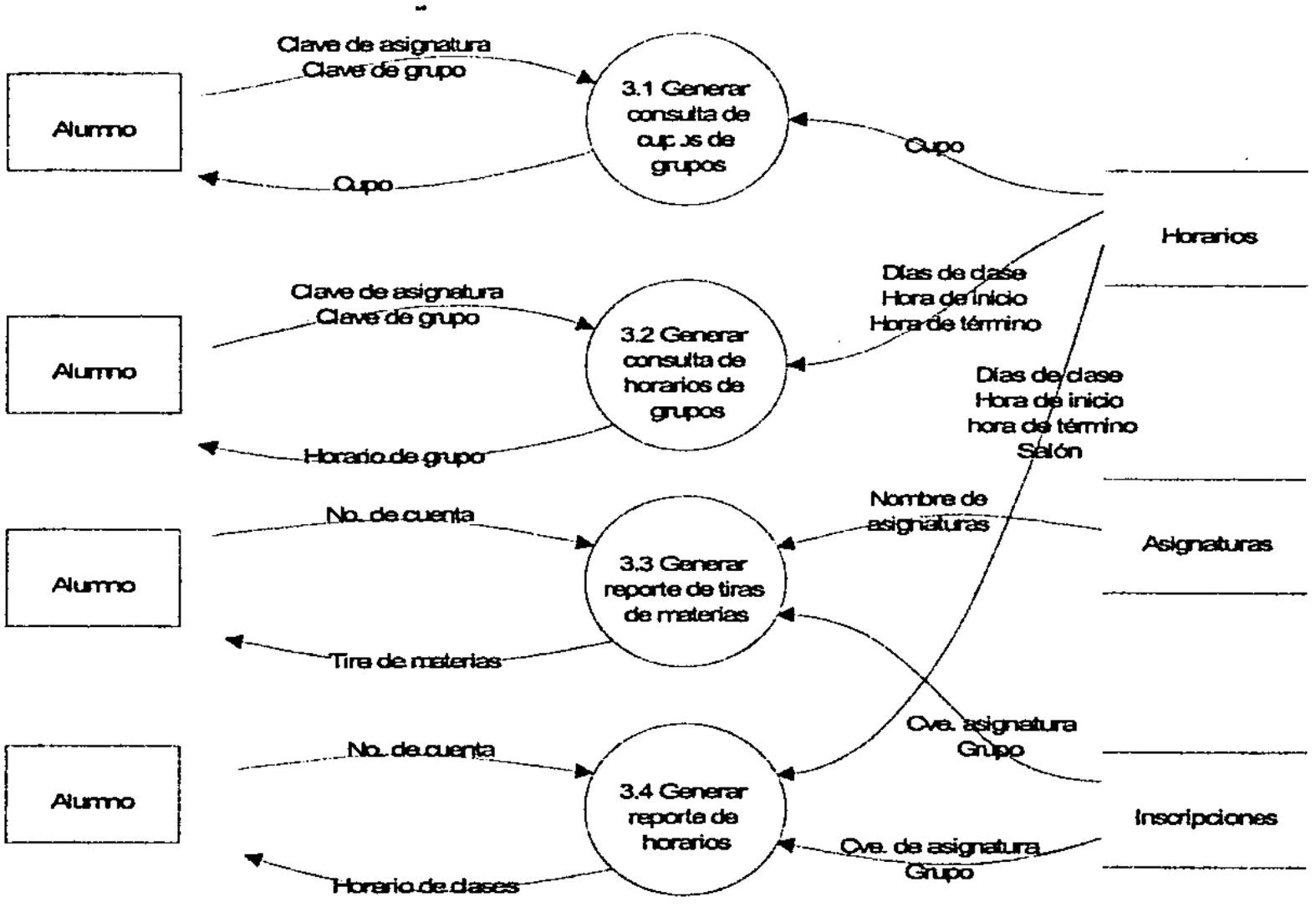


Diagrama de flujo de datos nivel 2 del proceso 4 "Generar bloque".

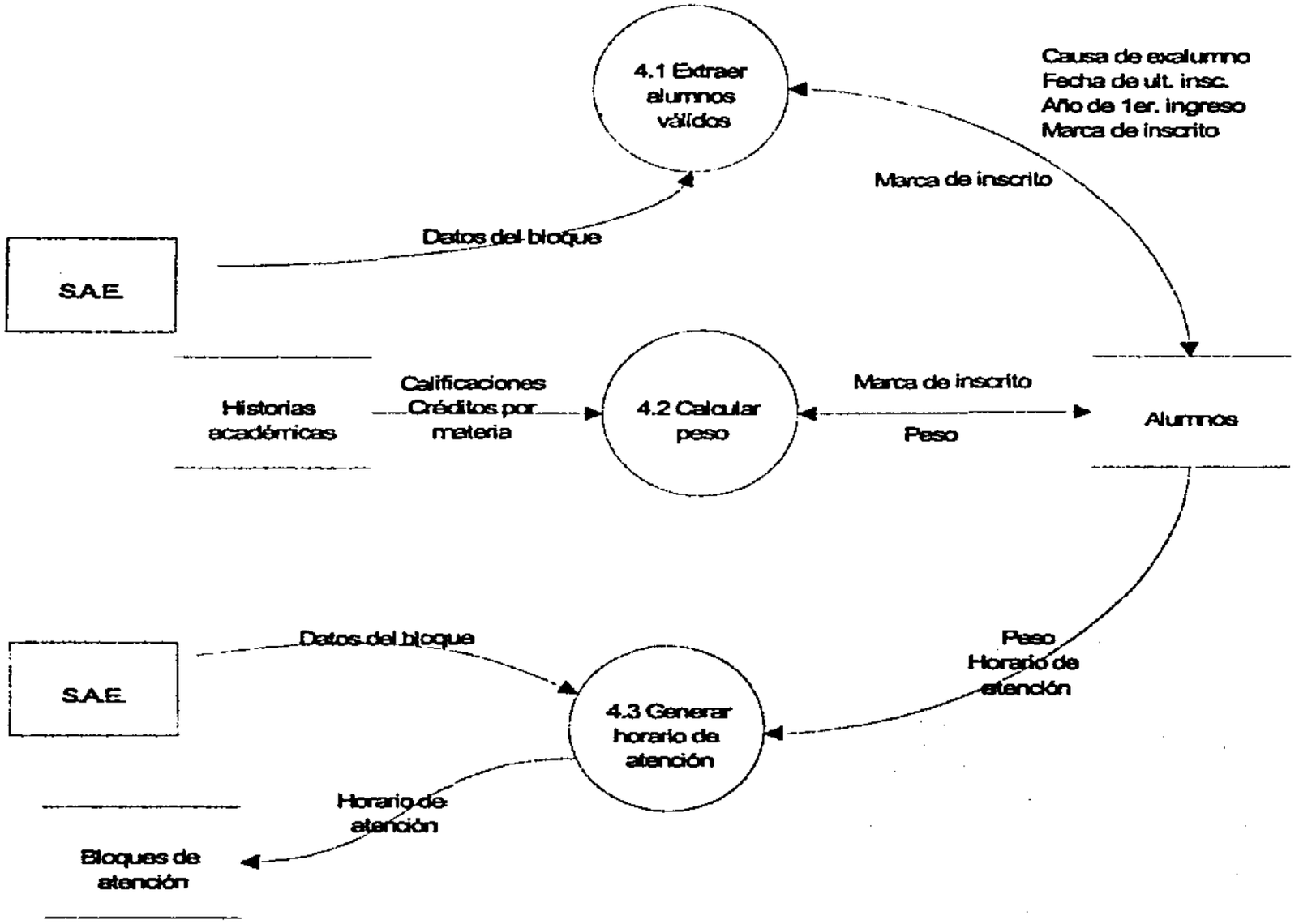


Diagrama de flujo de datos nivel 2 del proceso 5 "Generar información de SAE".

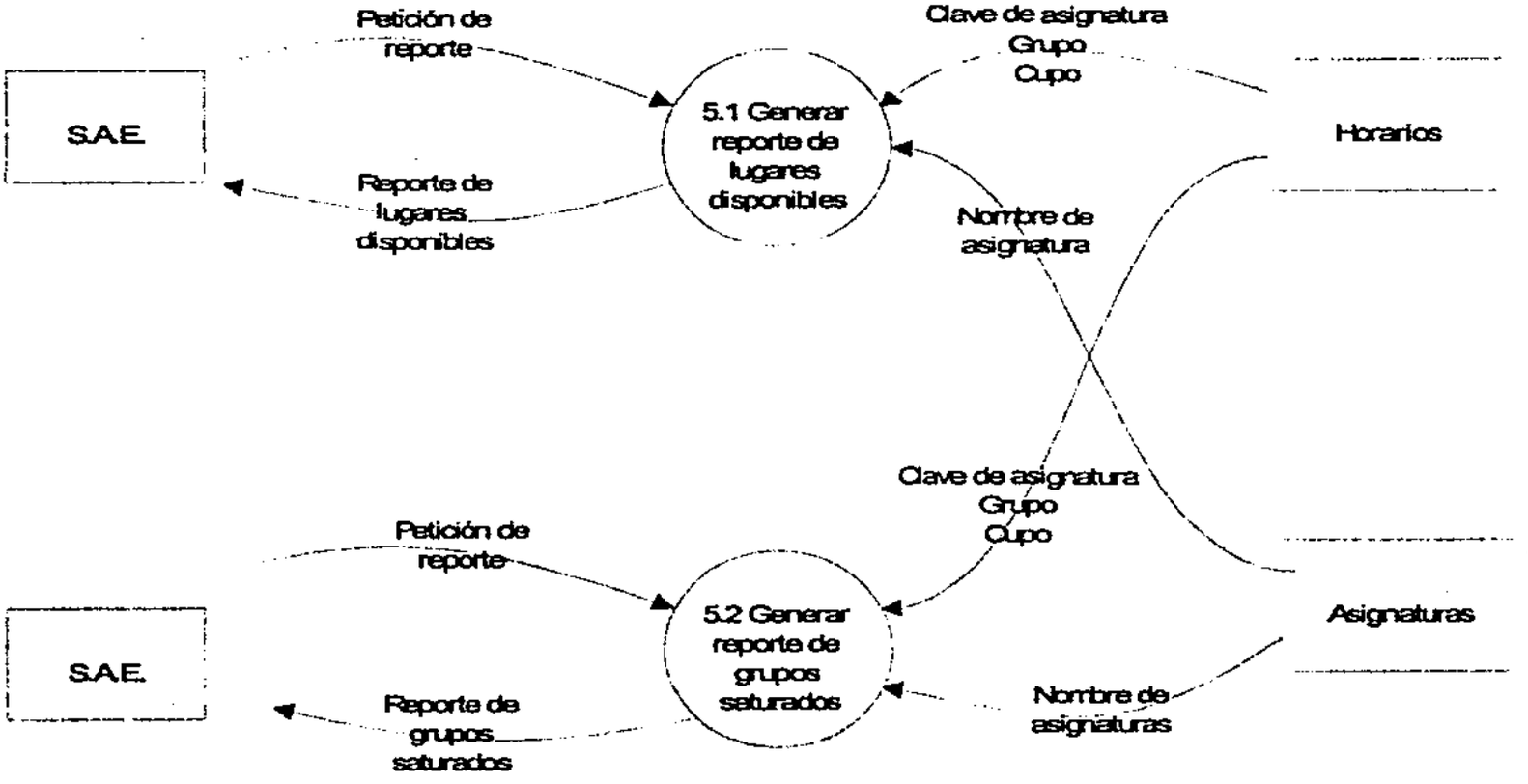


Diagrama de flujo de datos nivel 2 del proceso 6 "Generar movimiento de datos extraoficiales".

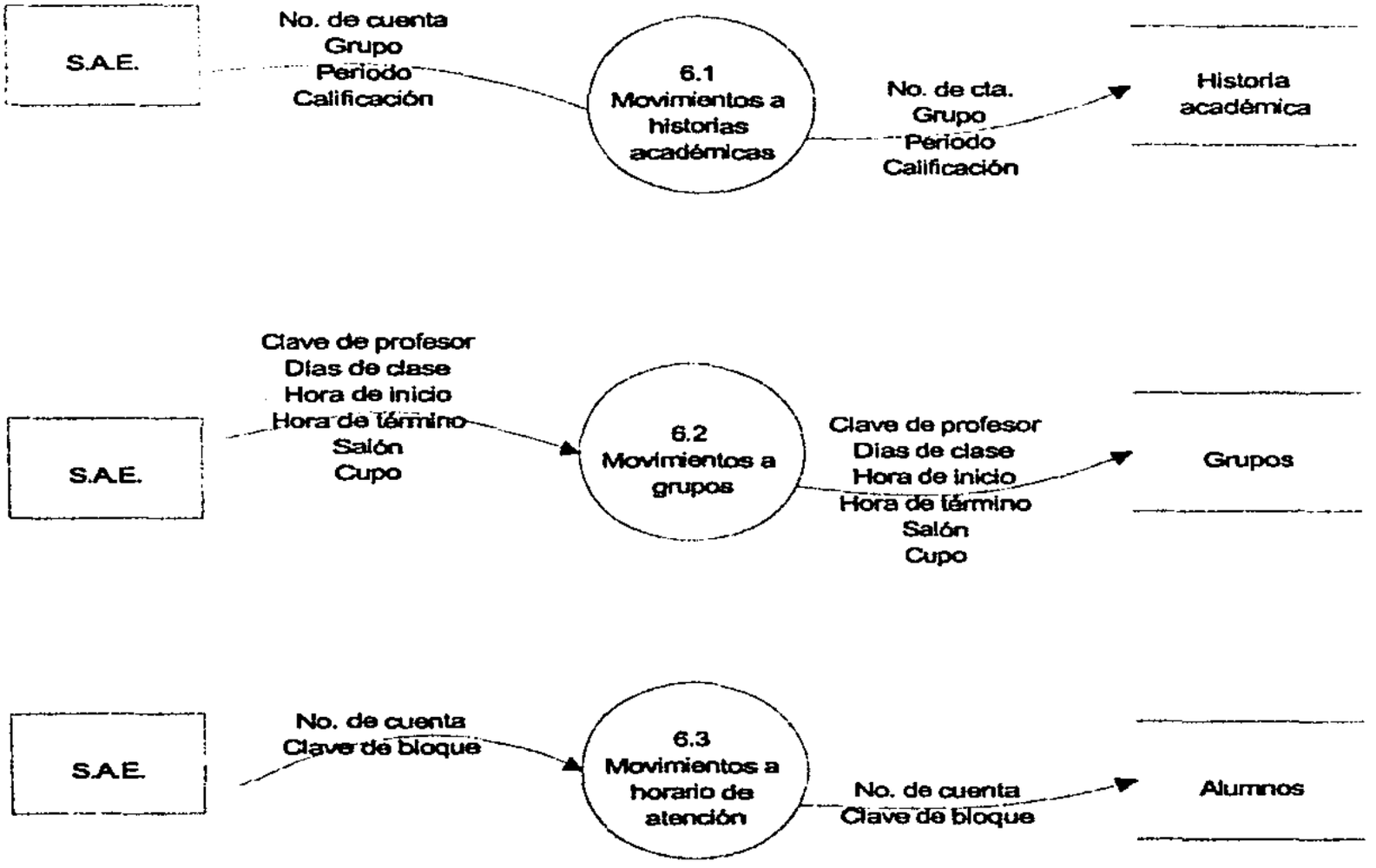


Diagrama de flujo de datos nivel 2 del proceso 7 "Recepción de datos de DGAE".

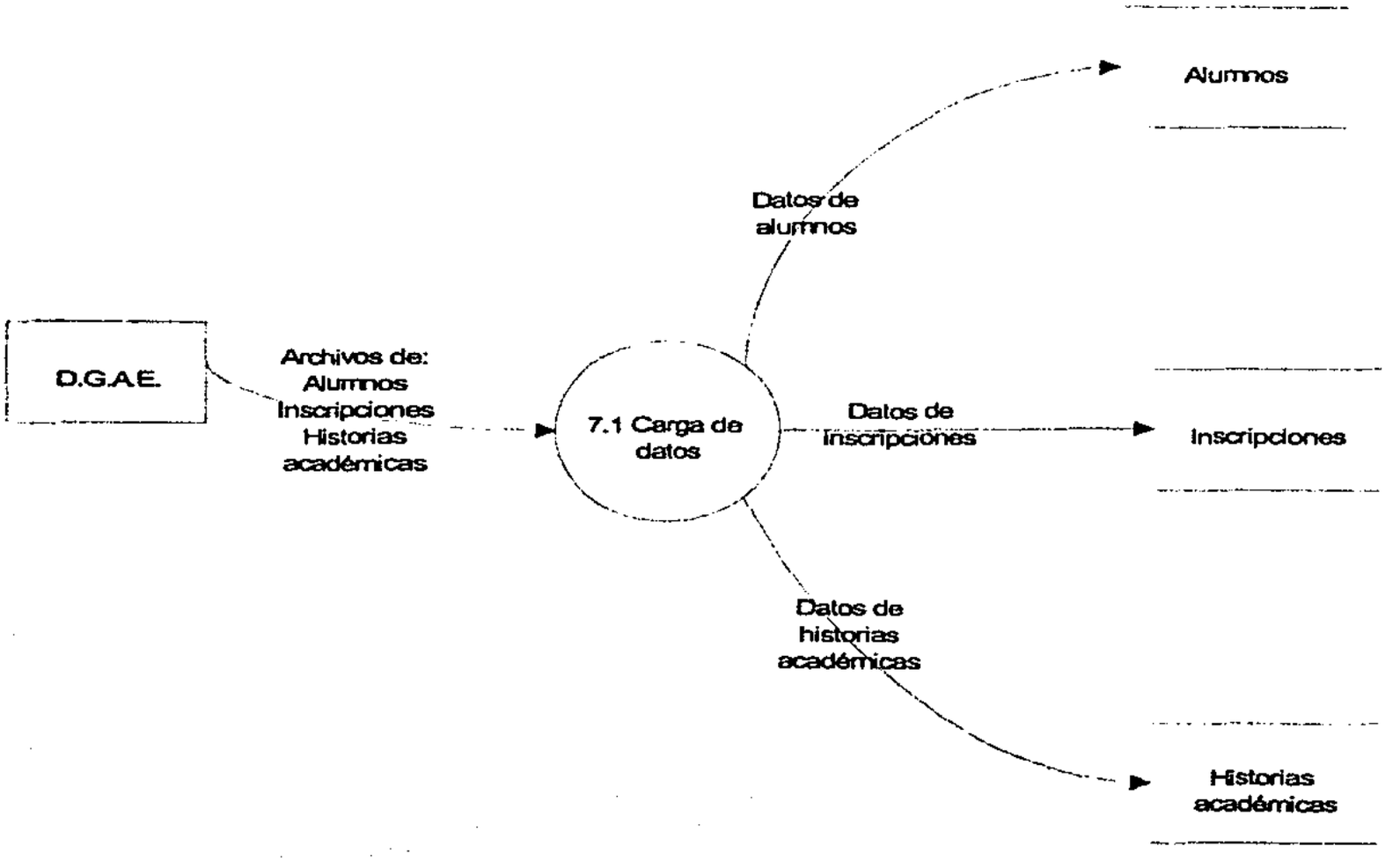
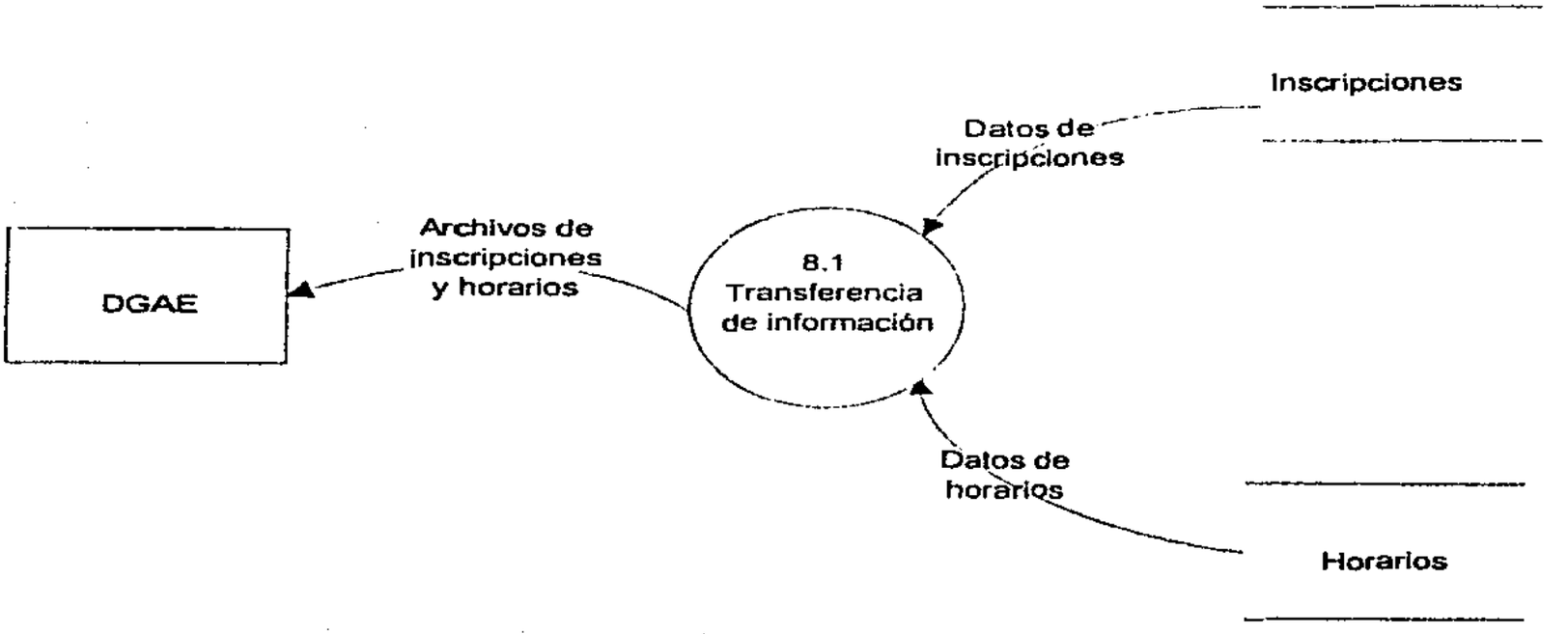


Diagrama de flujo de datos nivel 2 del proceso 8 "Envío de datos a DGAE".



4.1.2.2 Diccionario de datos.

Deben quedar descritos terminadores, almacenes y flujos.

Concepto	Descripción
Alumno.	=* almacén con información de las personas registradas en la Facultad de Derecho con posibilidades de cursar asignaturas. * @No. de cuenta + @clave de estado civil + @clave de bachillerato + @clave de municipio + apellido paterno + apellido materno + nombres + sexo + fecha de nacimiento + calle y número + colonia + código Postal + teléfono + extranjero + bachillerato en el extranjero + NIP + correo electrónico + promedio de bachillerato + folio + activo.
ALUMNOS.	={alumno}
Asignatura.	=* almacén que define las características de las materias que se imparten en la facultad. * @clave de asignatura + nombre + descripción + temario + activo.
Bloque de atención.	=*almacén con los datos de rango de horas de atención para la inscripción de asignaturas de una modalidad dada. * @clave de bloque de atención + @clave de modalidad de inscripción + @clave de periodo + @clave de turno + hora de inicio + hora de término + tiempo máximo de inscripción + número máximo de movimientos + número máximo de atenciones + altas + bajas + cambios + consultas + activo.

Concepto	Descripción
Bloques de atención.	={bloque de atención}.
Cupo de grupos.	=* número máximo de alumnos que es posible inscribir en una asignatura *
DGAE.	=* organismo encargado de administrar la actividad escolar y la información necesaria para este fin. (Dirección General de Administración Escolar). *
Envío de información a DGAE.	=* procedimiento de traslado de información producto de las inscripciones, el organismo rector. *
Grupo.	=*almacén con los datos de profesores asignados a un conjunto de alumnos para darles clases de una asignatura. * @clave de grupo + @clave de asignatura-plan + @clave de periodo + @clave de modalidad de inscripción + @clave de estado de grupo + @clave de turno + cupo + lugares ocupados.
Historia académica.	=* almacén que contiene todos los resultados académicos del alumno. * @no. de cuenta + folio del acta + @clave de asignatura + @clave de tipo de examen + @clave de grupo + calificación + número de exámenes.
Horarios de atención.	=* almacén con el rango de horas de atención para la inscripción de asignaturas de una modalidad dada de un alumno específico. * @clave de horario de atención + @clave de modalidad de inscripción + @clave de periodo + clave de turno + hora de inicio + hora de termino + tiempo máximo de inscripción + número máximo de movimientos + número máximo de atenciones + altas + bajas + cambios + consultas + activo

Concepto	Descripción
Horarios de grupos.	=* almacén con los rangos de horas y días en los que se imparte una asignatura. * @clave de horario de grupo + @clave de aula + lunes + martes + miércoles + jueves + viernes + sábado + hora de inicio + hora de término.
Inscripción.	=* almacén que controla las asignaturas que un alumno ha cursado y las que está actualmente cursando. * @clave de plan + número de cuenta + clave de grupo + clave de estado de inscripción + clave de calificación + fecha de inscripción.
Movimientos a grupos.	=* cambios a las definiciones de hora, profesor y aula de los grupos. *
Movimientos a historias académicas.	=* correcciones a los datos contenidos en la historia académica. *
Movimientos a horarios de atención.	=* modificaciones al horario de atención asignado a un alumno. *
Reporte de grupos saturados.	=* listado de los grupos cuyo cupo es cero. *
Reporte de horario de clase.	=* listado del horario de clase que le corresponde a un alumno una vez realizada la inscripción. *
Reporte de lugares disponibles.	=* listado de los lugares disponibles por asignatura. *
Recepción de información de DGAE.	=* proceso de aceptación y carga de la información que envía DGAE y sirve de base para todo el proceso de inscripciones. *
Reporte de lugares disponibles.	=* listado de los lugares disponibles por asignatura. *

Concepto	Descripción
Recepción de información de DGAE.	=* proceso de aceptación y carga de la información que envía DGAE y sirve de base para todo el proceso de inscripciones. *
SAE.	=*organismo de la Facultad de Derecho responsable del proceso de inscripciones. (Secretaría de Asuntos Escolares).
Solicitud de información de cupo de grupos.	=* Petición de informe sobre el cupo de los grupos en cuestión. *
Solicitud de generación de bloques.	=* Señal de arranque del proceso de generación de bloques de atención. *
Solicitud de horario de atención.	=* Petición para generar el horario de atención. *
Solicitud de información de horarios de grupos.	=* Consulta del horario de cierto grupo. *
Solicitud de movimiento de asignatura (alta, baja, cambio).	=* Proceso de inscripción en sí, mediante una petición de alta, baja o cambio en una asignatura en particular. * número de cuenta + clave de asignatura + clave de grupo.
Solicitud de reporte de grupos saturados.	=* Petición para generar el reporte de grupos sin cupo. *
Solicitud de reporte de horario de clase.	=* Petición para generar el reporte de horario de clase asignado al alumno. *
Solicitud de reporte de lugares disponibles.	=* Petición para generar el reporte de lugares disponibles en los grupos. *
Solicitud de reporte de tiras de materias.	=* Petición de impresión de la tira de materias. *
Tira de materias.	=* Comprobante oficial de la inscripción de asignaturas. *
Vector.	=* Almacén con información del estado del total de las asignaturas por alumno. * @no. de cuenta + @clave de asignatura + estado.

4.1.2.3 Diagrama entidad/relación

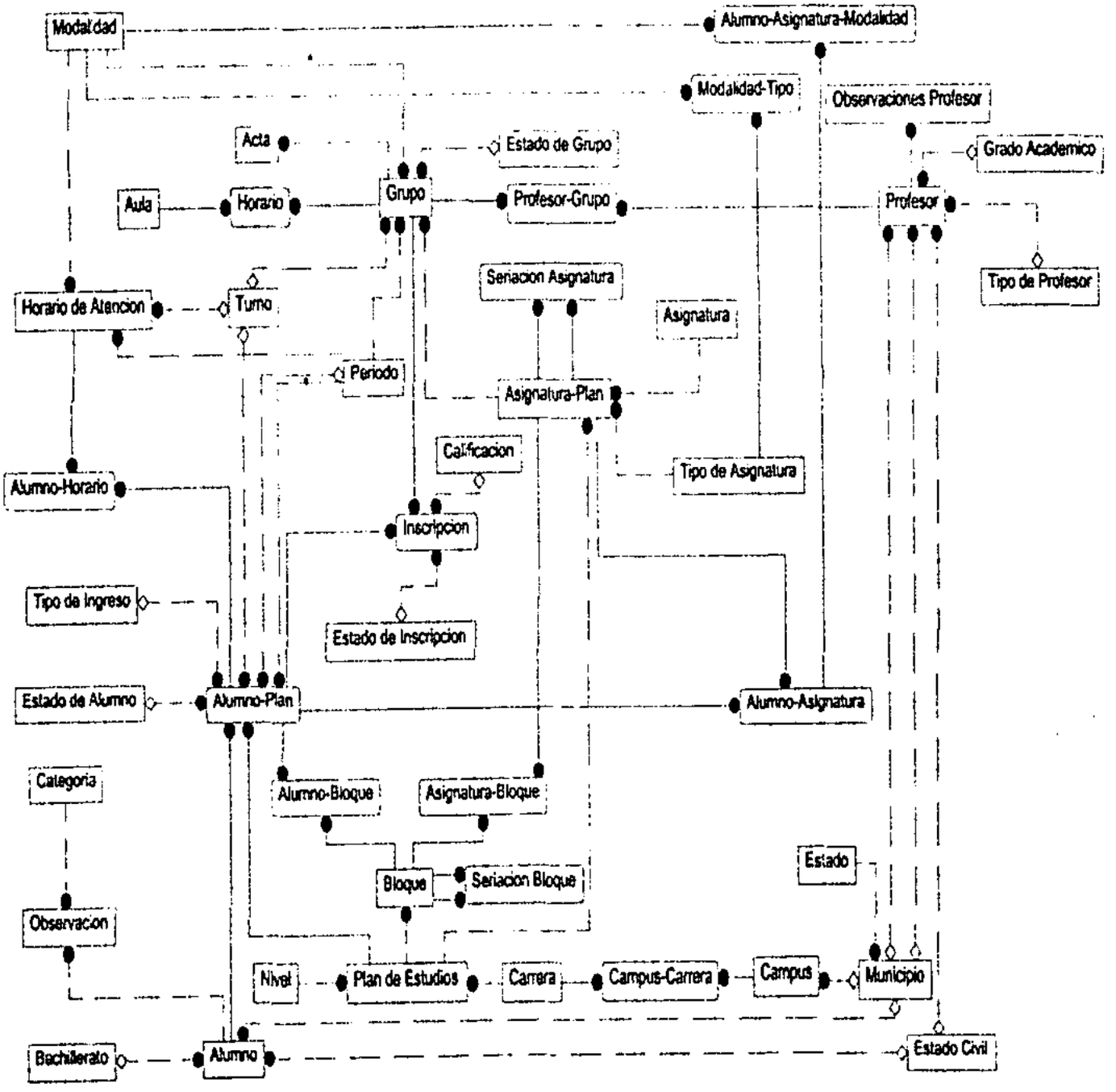
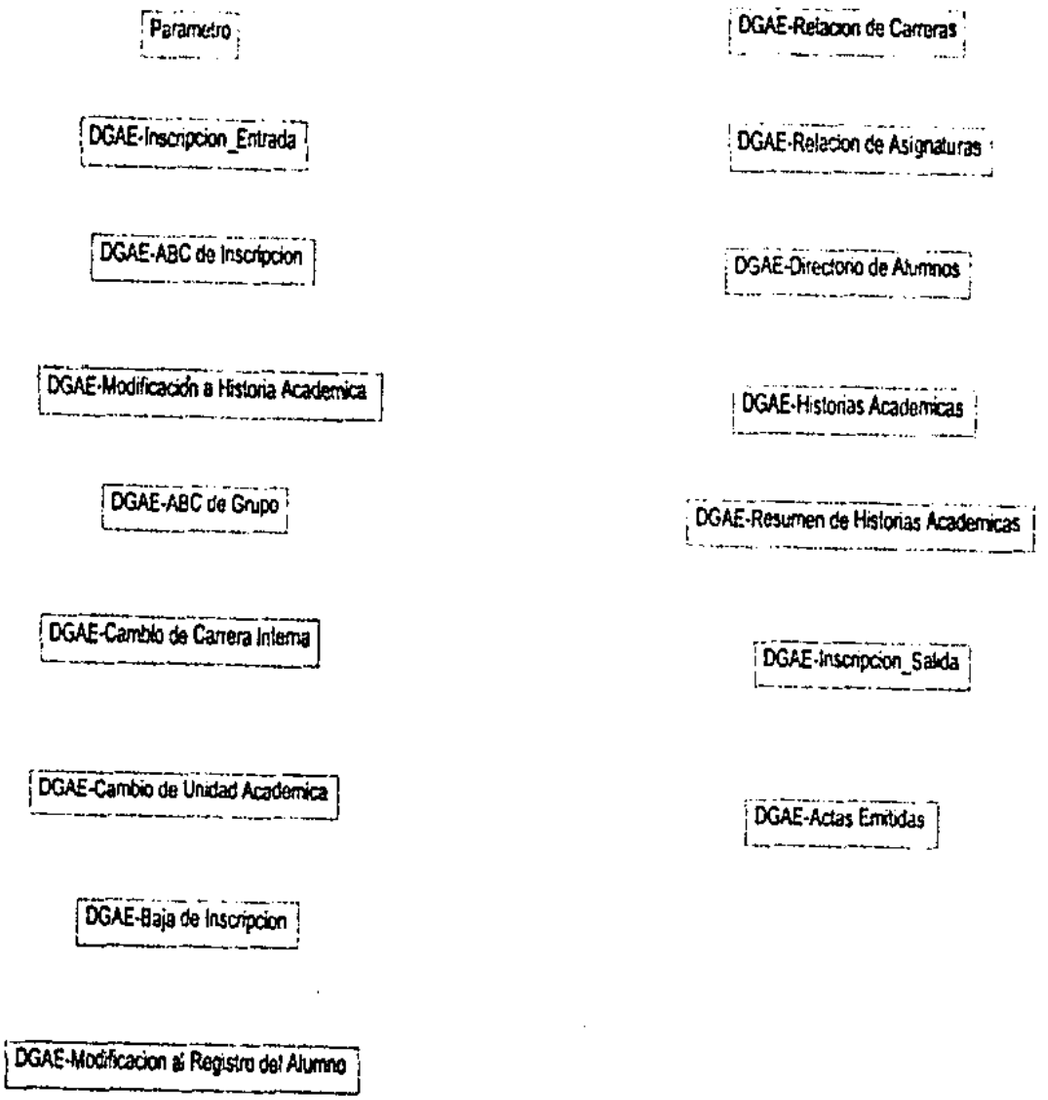


Diagrama entidad/relación



4.1.2.4 Diagrama de transición de estados.

Diagrama de transición de estados del proceso 1 "Hacer movimiento".

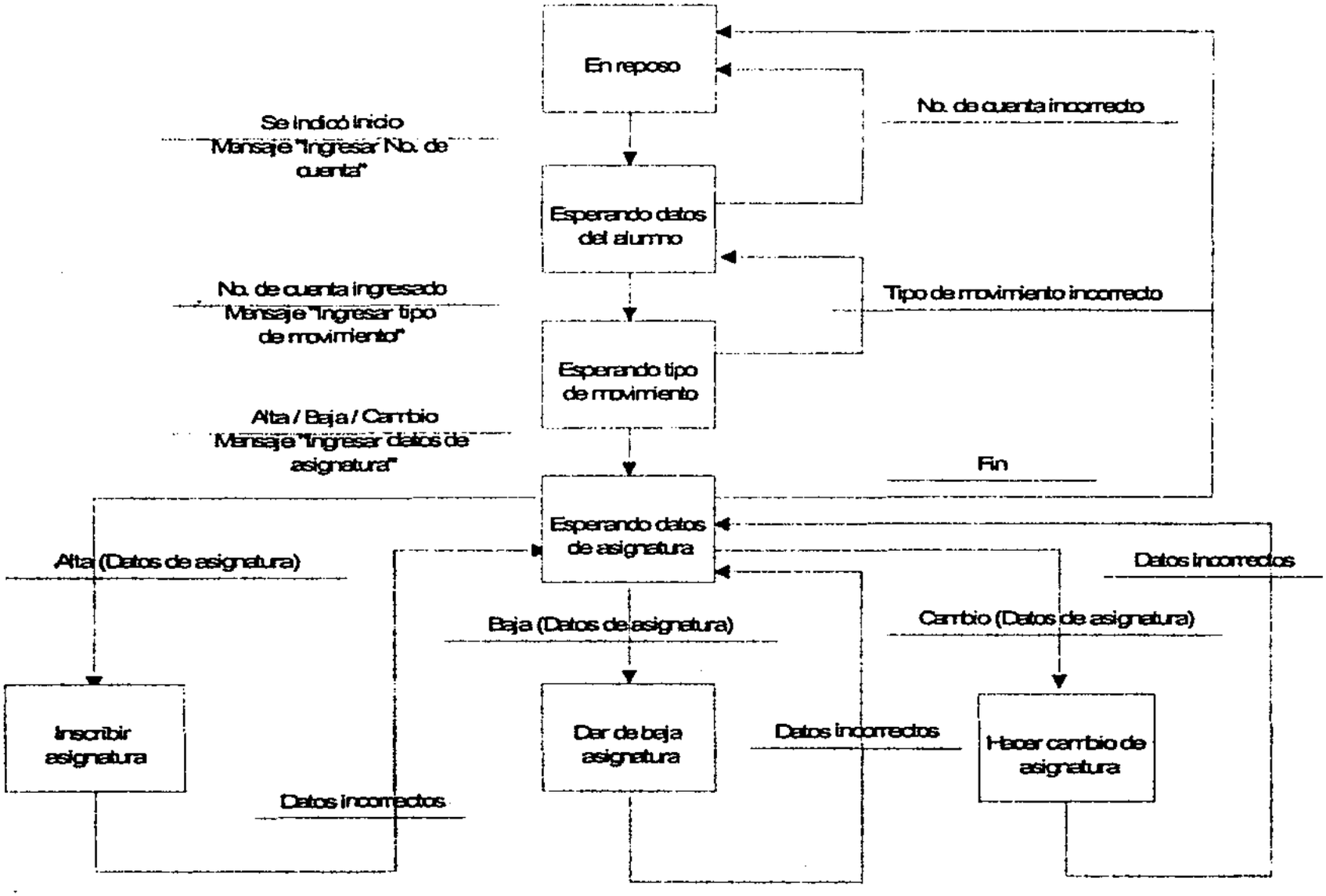


Diagrama de transición de estados del proceso 2 "Generar horario de atención".

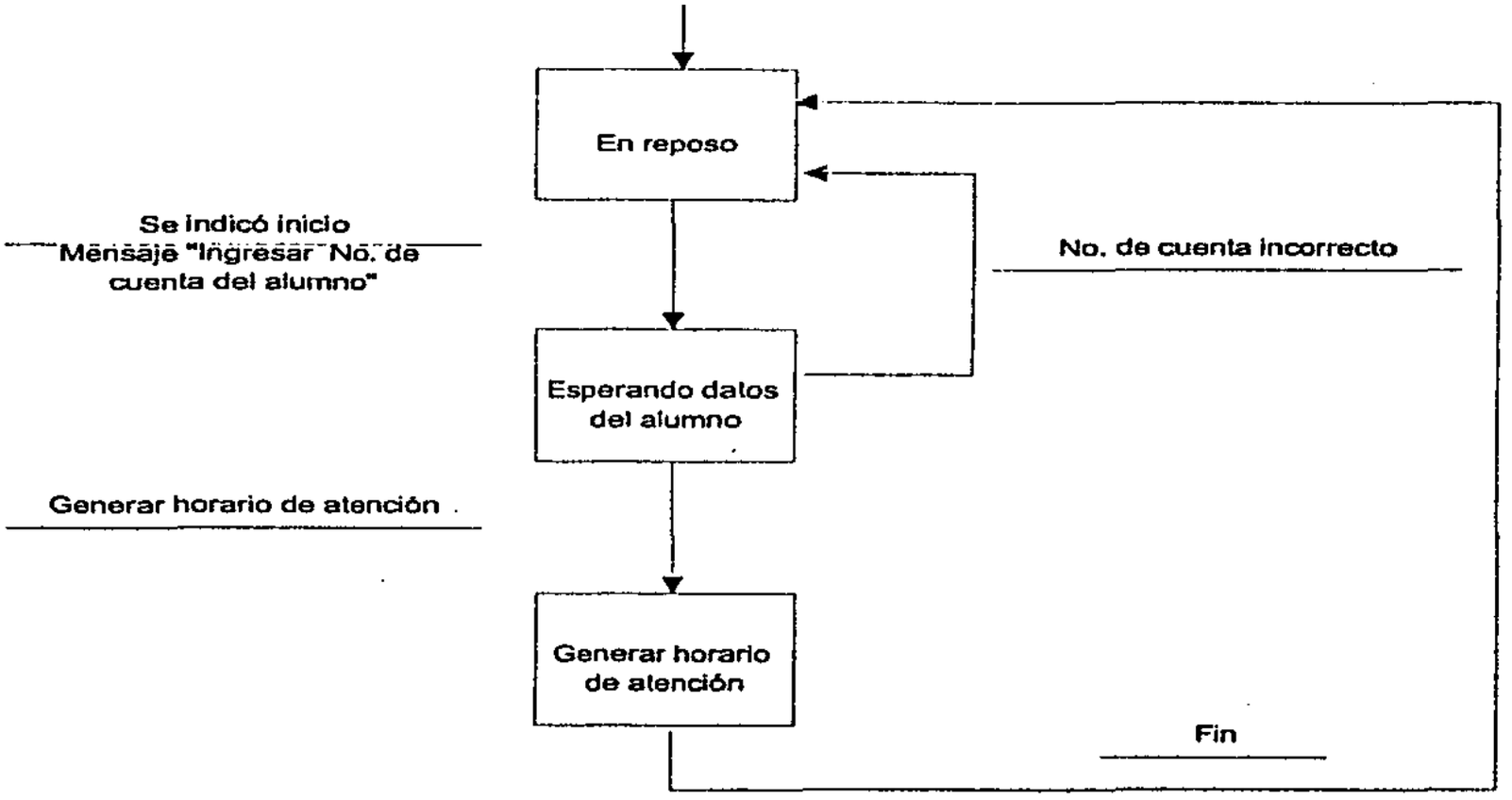


Diagrama de transición de estados del proceso 3 "Generar información".

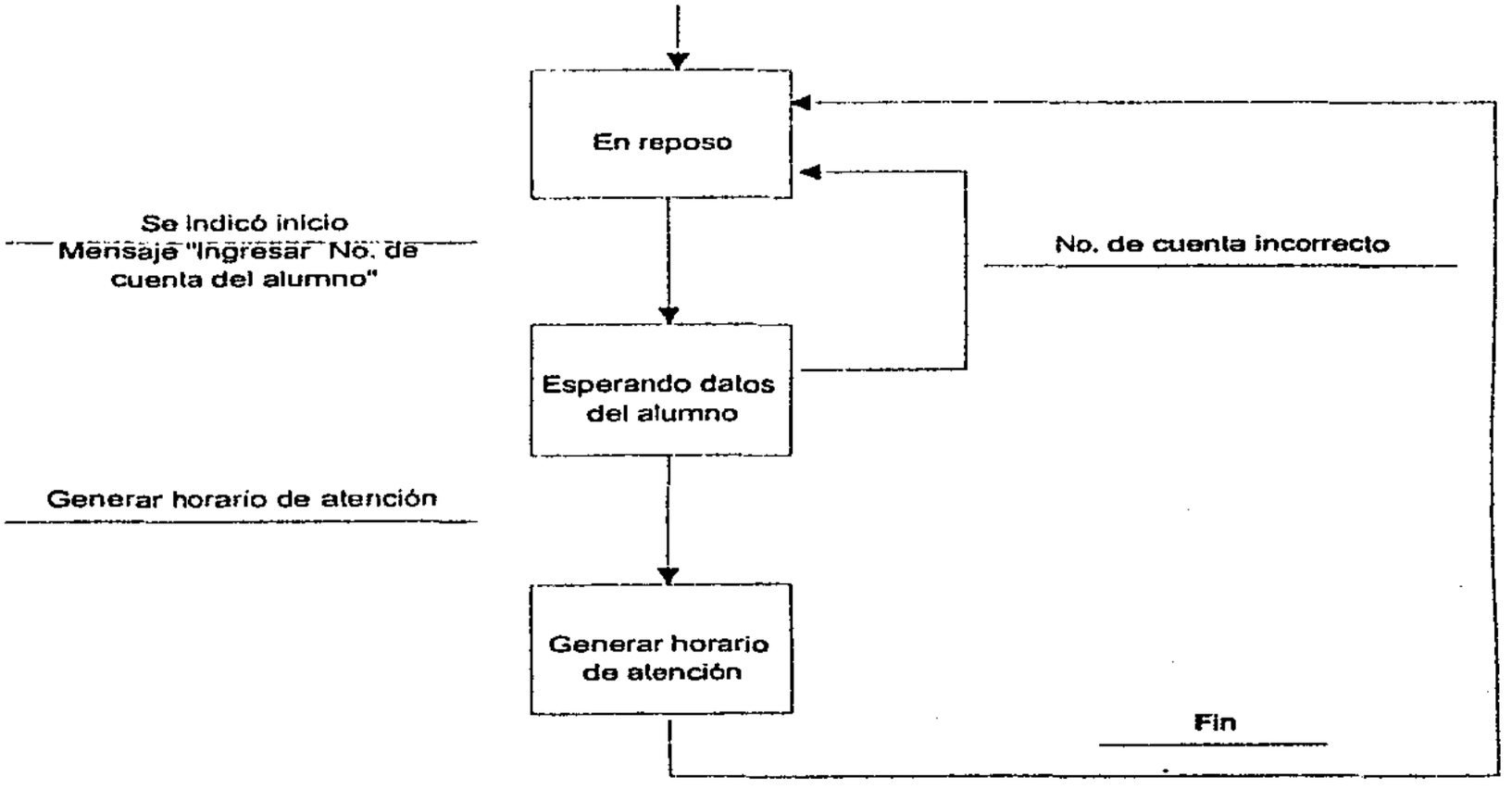


Diagrama de transición de estados del proceso 3, subproceso "Generar consulta de cupos de grupos".

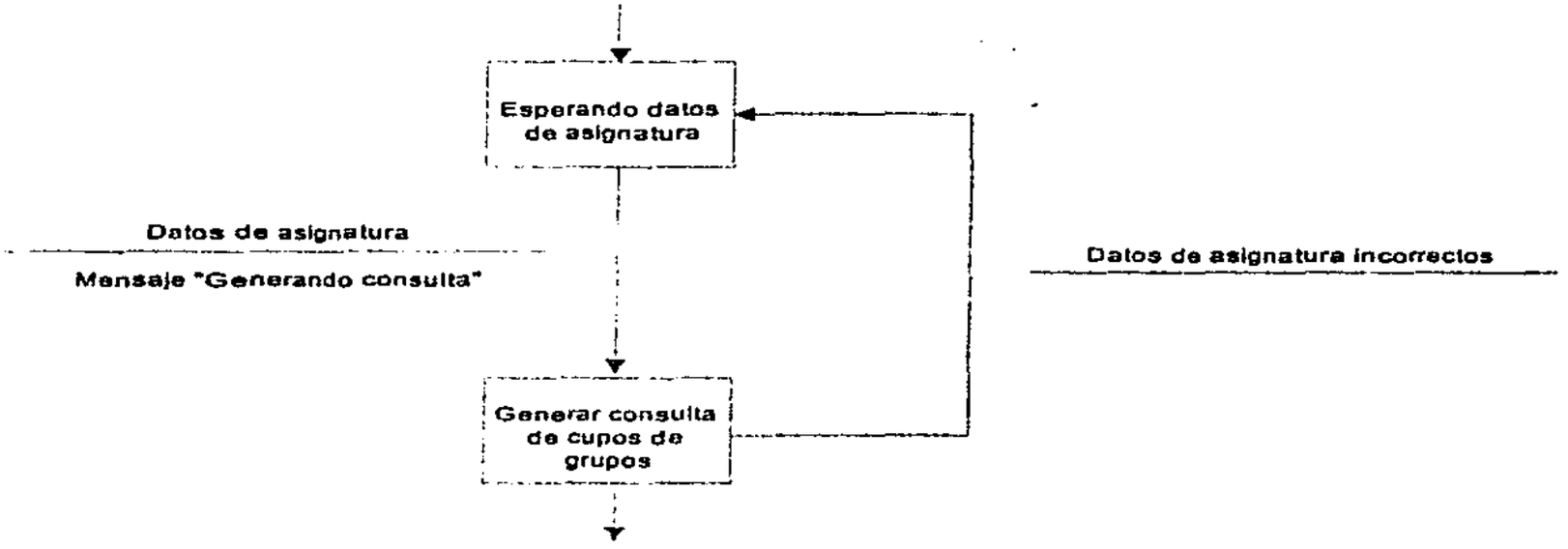


Diagrama de transición de estados del proceso 3, subproceso "Generar consulta de horarios de grupos".

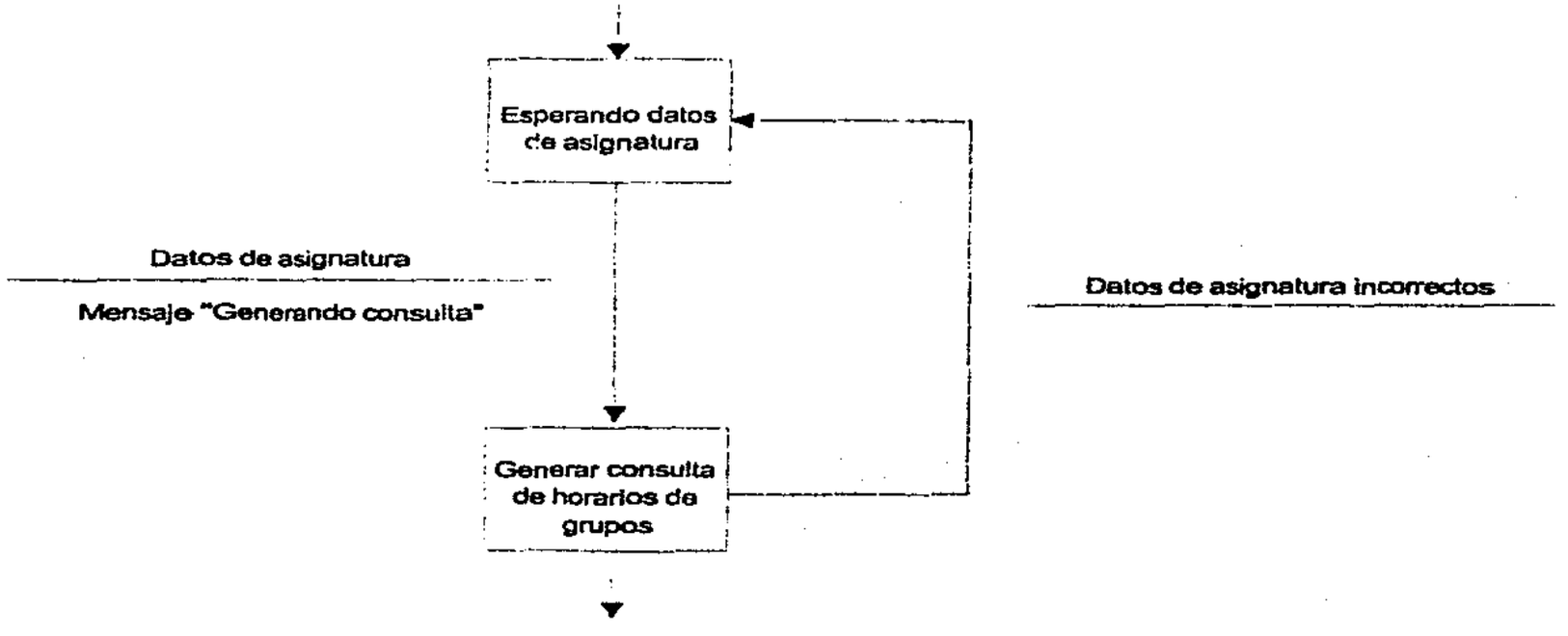


Diagrama de transición de estados del proceso 3, subproceso "Reporte de tiras de materias"

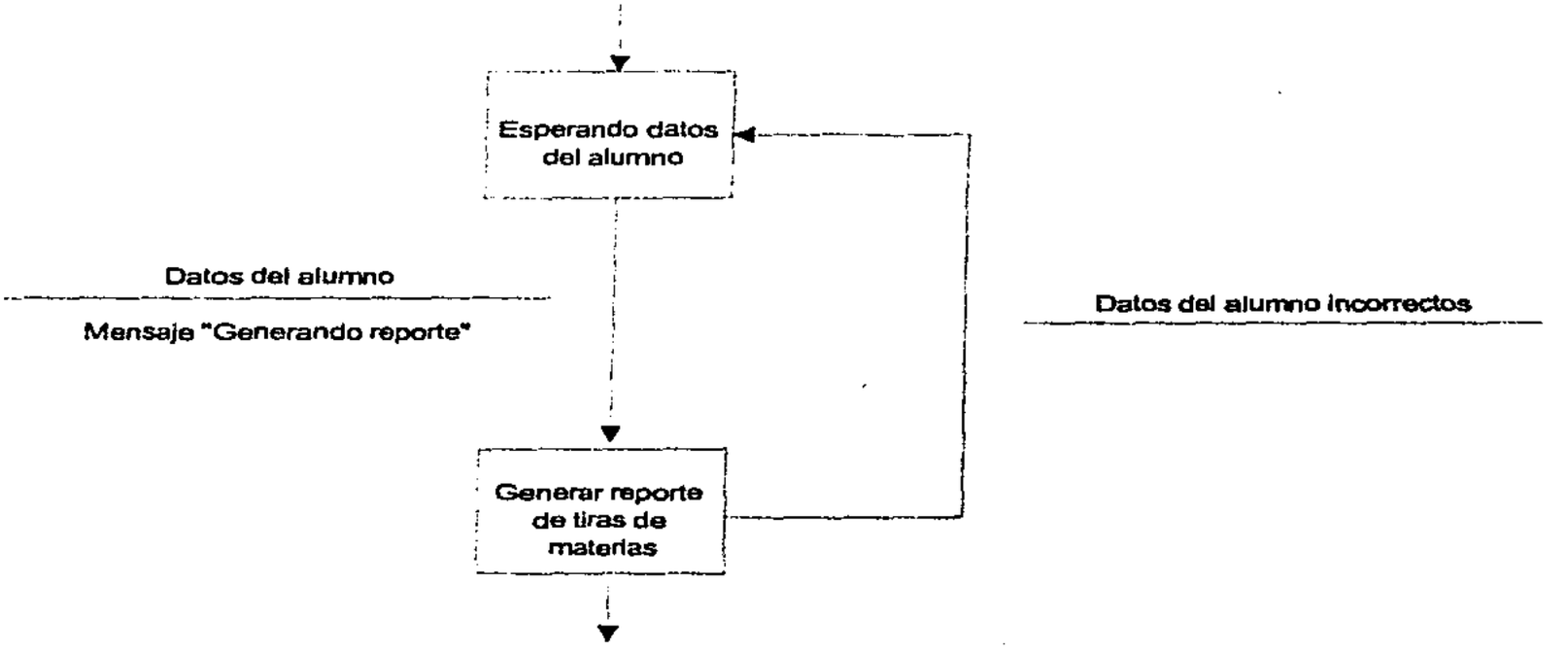


Diagrama de transición de estados del proceso 3, subproceso "Reporte de horarios".

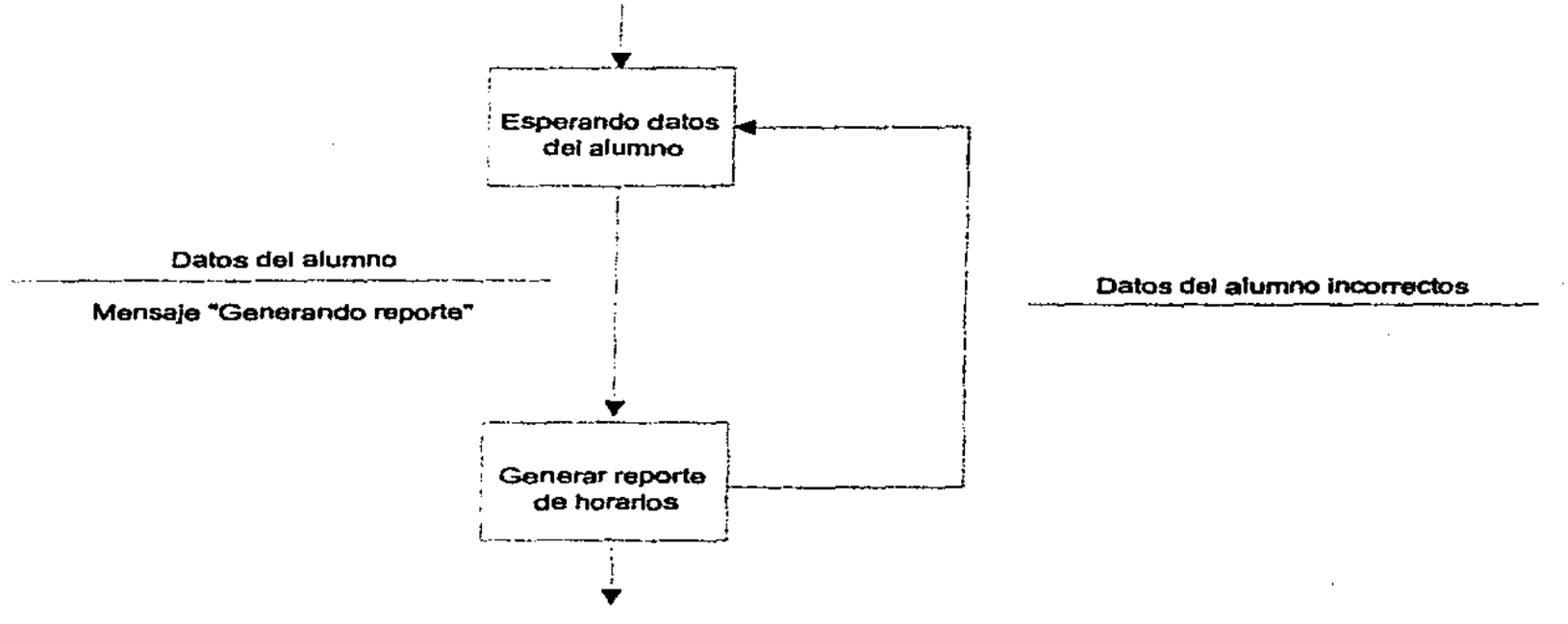
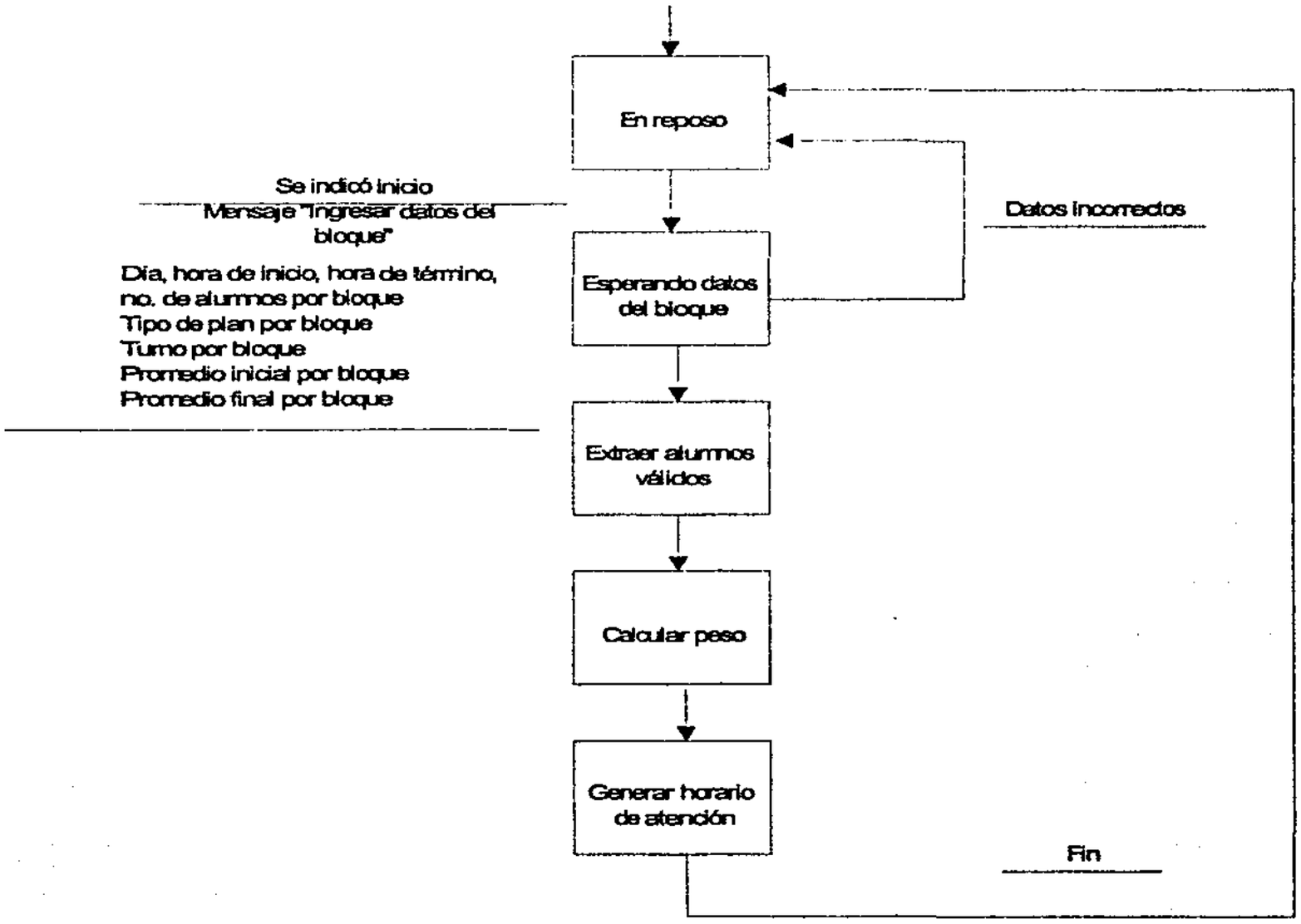


Diagrama de transición de estados del proceso 4 "Generar bloque".



4.2

Diseño

4.2.1 Diseño externo.

4.2.1.1 Definir subsistemas.

Como criterio de diseño se estableció que debido a que las funciones que debe cubrir el sistema de inscripciones, no se requieren usar simultáneamente, y puesto que es un factor determinante lograr tiempos de respuesta pequeños, es conveniente dividir el sistema en módulos de propósito específico de tal suerte que se utilicen cuando sea preciso y por tener un tamaño pequeño consuman pocos recursos del sistema y gocen de gran velocidad.

Desde esta perspectiva, los módulos con los que contará el sistema son:

Nombre del módulo	Descripción
1.-Inscripción	Módulo principal para uso en ventanillas de atención, su función es asignar materias a los alumnos, de acuerdo a las reglas establecidas.
2.- Alumno	Se encargará de administrar la información relacionada con alumnos.
3.- Profesor	Controla la información de académicos en su relación con los alumnos y con las asignaturas.
4.- Programa	Administra los planes de estudio, bloques de asignaturas y seriación de las mismas.
5.- Campus – Carrera	Controla las diferentes carreras y su relación con los planes de estudio y campus donde se imparten.
6.- Grupo	Su función es controlar los grupos existentes y su relación con profesores y aulas.
7.- Horario de atención	Controla el horario de atención de alumnos durante la época de inscripciones.
8.- Estado de alumno	Administra la relación entre alumnos y las asignaturas.
9.- DGAE	Administra la información que se recibe y envía a la Dirección General de Administración Escolar.
10.- Catálogos	La tarea asignada es controlar los datos de catálogos de uso general, usados por el resto de los módulos.

4.2.1.2 Definir las funciones por módulo del sistema.

A continuación se tiene una descripción general de los módulos con que contará el sistema y las tareas que cada uno debe realizar.

Inscripción.

El objetivo central es hacer altas, bajas y cambios de asignaturas para un alumno, además de imprimir su comprobante de inscripción.

Para este efecto, se proporciona el número de cuenta, la clave de la asignatura y el grupo, y con estos datos el servidor ejecuta una serie de procedimientos almacenados para asegurarse de que un movimiento es válido. Estos procedimientos se detallan en el diseño interno.

Alumnos.

En este módulo se administrará la información de alumnos como: datos personales, domicilio, escuela de procedencia, plan de estudios y observaciones en general. Aunque la información oficial de alumnos la proporciona DGAE, en esta parte del sistema se debe poder hacer altas, bajas y consultas a la información proporcionada.

Profesor.

Esta parte del sistema controlará la siguiente información de profesores: datos personales, domicilio particular y de trabajo e información laboral de profesores. De la misma forma que alumnos, estos datos los envía DGAE pero es necesario dar de alta, baja, hacer cambios y consultar información de académicos.

Programa.

Aquí se manejarán los planes de estudio, bloques, asignaturas, tipos de asignatura, modalidad, y las relaciones asignatura - plan, asignatura – bloque y tipo de asignatura – modalidad. Los grupos de datos necesarios para administrar esta información son: generales del plan, créditos, duración del plan, generales del bloque, seriación de bloques, generales de la asignatura, temario de

asignaturas y modalidad de inscripción. Nuevamente, será posible insertar, eliminar, cambiar y consultar la información antes mencionada.

Campus – Carrera.

Este subsistema tendrá el control de los datos generales de carreras, campus y las relaciones entre ambos. Poseerá la capacidad de agregar, eliminar, cambiar y consultar información de: descripción general de una carrera, datos generales del campus y campus en que se imparten las carreras.

Grupo.

Aquí se relaciona la información de grupos, horarios de grupos, aulas donde se imparten, profesores asignados, actas de calificaciones, periodo semestral y turno. Esto significa que se pueden dar de alta, baja, hacer cambios y consultas de información de grupos, la cual como se mencionó, se relaciona con las entes citadas.

Horario de atención.

La intención de esta parte es controlar los horarios de atención de alumnos durante el periodo de inscripciones. Esto implica ejecutar un proceso que, considerando la información que DGAE suministre, arme los bloques de atención de acuerdo a su promedio y avance en créditos. Cuando este proceso concluya será posible consultar, dar de alta, baja o hacer cambios a los horarios de atención.

Estado de alumno.

Esta parte permite administrar la relación de los alumnos con las asignaturas existentes y las modalidades de inscripción con que se cuente. Esto significa que, por medio de procedimientos

almacenados, se llenará las tablas que este subsistema considera. Desde la parte cliente, se podrán hacer consultas de la relación alumno – asignatura – modalidad.

DGAE.

La función de este módulo es incorporar al esquema de bases de datos del sistema de inscripciones la información que la DGAE envía electrónicamente al servidor, y una vez realizado el proceso de inscripción, formatear los datos que genera el sistema de inscripciones y enviarlos a la DGAE de acuerdo al estándar que esta dependencia establezca.

Catálogos.

La idea al crear este módulo es tener un subsistema que administre catálogos de uso general en el resto de los módulos y que se tenga una alta flexibilidad para dar de alta, baja, realizar cambios y consultas a dichos catálogos, esta característica es necesaria si consideramos que los datos que ahí residirán salen del control de la facultad por ser de uso general. La lista completa de los catálogos se especifica en el diseño interno.

4.2.1.3 Definir esquema de seguridad.

El sistema de inscripciones requiere los siguientes niveles de seguridad:

Nivel	Descripción
1.-Administrador del sistema	Es el usuario con mayor capacidad de acción, puede operar todas los módulos del sistema y administrar al resto de los usuarios.
2.-Personal técnico del centro de cómputo	Tiene acceso a todos los módulos, su limitante es que no tiene capacidad para crear nuevos usuarios.
3.- Funcionario de la SAE	Opera todos los módulos excepto DGAE y catálogos.
4.-Usuario de ventanillas de atención	Sólo puede operar el módulo de inscripción.

Vale la pena destacar que el sistema está diseñado de tal modo que los niveles de usuarios pueden variar conforme cambien las condiciones del entorno, debido a que el nivel de seguridad se administra apoyándose en Sybase para fines de las bases de datos y de los procedimientos almacenados.

Esto significa que si bien un usuario de ventanillas, por ejemplo, pretende modificar las políticas de un plan de estudio, aunque ingresa al módulo, al momento de que se quiere hacer cualquier movimiento, entran en funcionamiento las propiedades de modificación definidas a nivel tabla, columna, registro y procedimiento almacenado.

Este criterio de diseño es preferible, que controlar el acceso a los módulos desde el cliente, puesto que la movilidad que va teniendo el personal que usa el sistema no afecta en lo absoluto la parte cliente, considerando que las variantes surgidas se implementan en el manejador de base de datos, sin tener que modificar código o recompilar programas.

4.2.2 Diseño interno.

4.2.2.1 Diseño de la arquitectura del sistema.

Como se aprecia en la figura de la arquitectura del sistema, se trata de un ambiente cliente/servidor donde tenemos las siguientes especificaciones técnicas:

Computadoras cliente.

Procesador	Memoria RAM	Tamaño del disco duro	Software
Pentium a 120 Mhz	16 Mb	120 Mb	. Windows 95 . Cliente de Power Builder v. 5.0

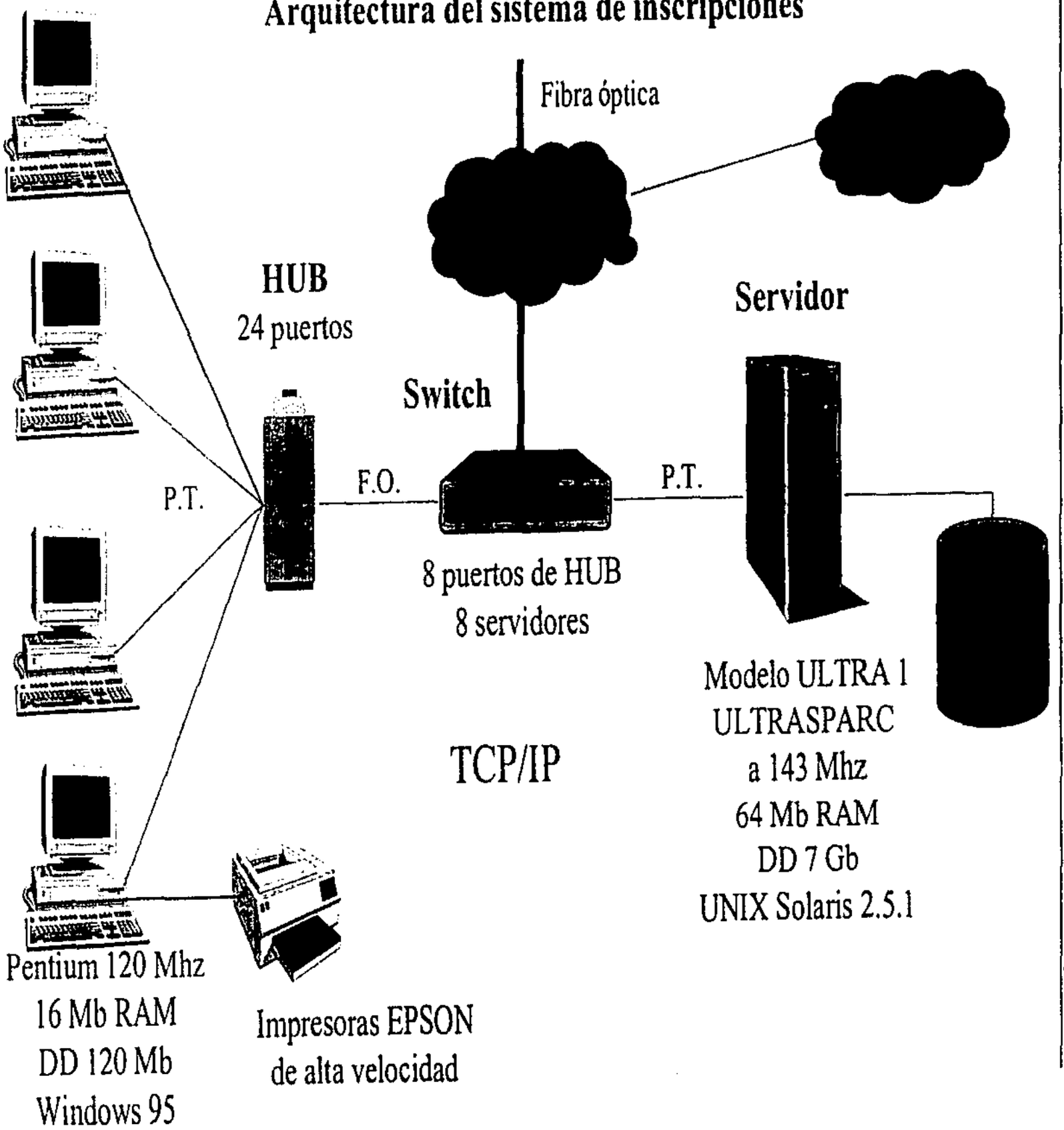
Servidor.

Procesador	Memoria RAM	Tamaño del disco duro	Software
Modelo ULTRA 1 con procesador ULTRASPARC a 143 Mhz	64 Mb	7 Gb	. Sistema operativo Solaris 2.5.1 (UNIX de SUN) . DBMS: Sybase v. 11

Red.

Protocolo de transmisión	Protocolo de control de acceso	Topología	Medio de transmisión físico
TCP/IP	CSMA/CD (Ethernet)	Estrella	Fibra óptica y par trenzado

Arquitectura del sistema de inscripciones



4.2.2.2 Diseño de la base de datos.

Para apoyar el desarrollo del proyecto se utilizó la herramienta de ingeniería de software asistida por computadora (CASE) ERwin para PowerBuilder 2.5.01. Por lo que todas las especificaciones de la base de datos están dentro de esta herramienta.

4.2.2.2.1 Entidades, atributos y relaciones.

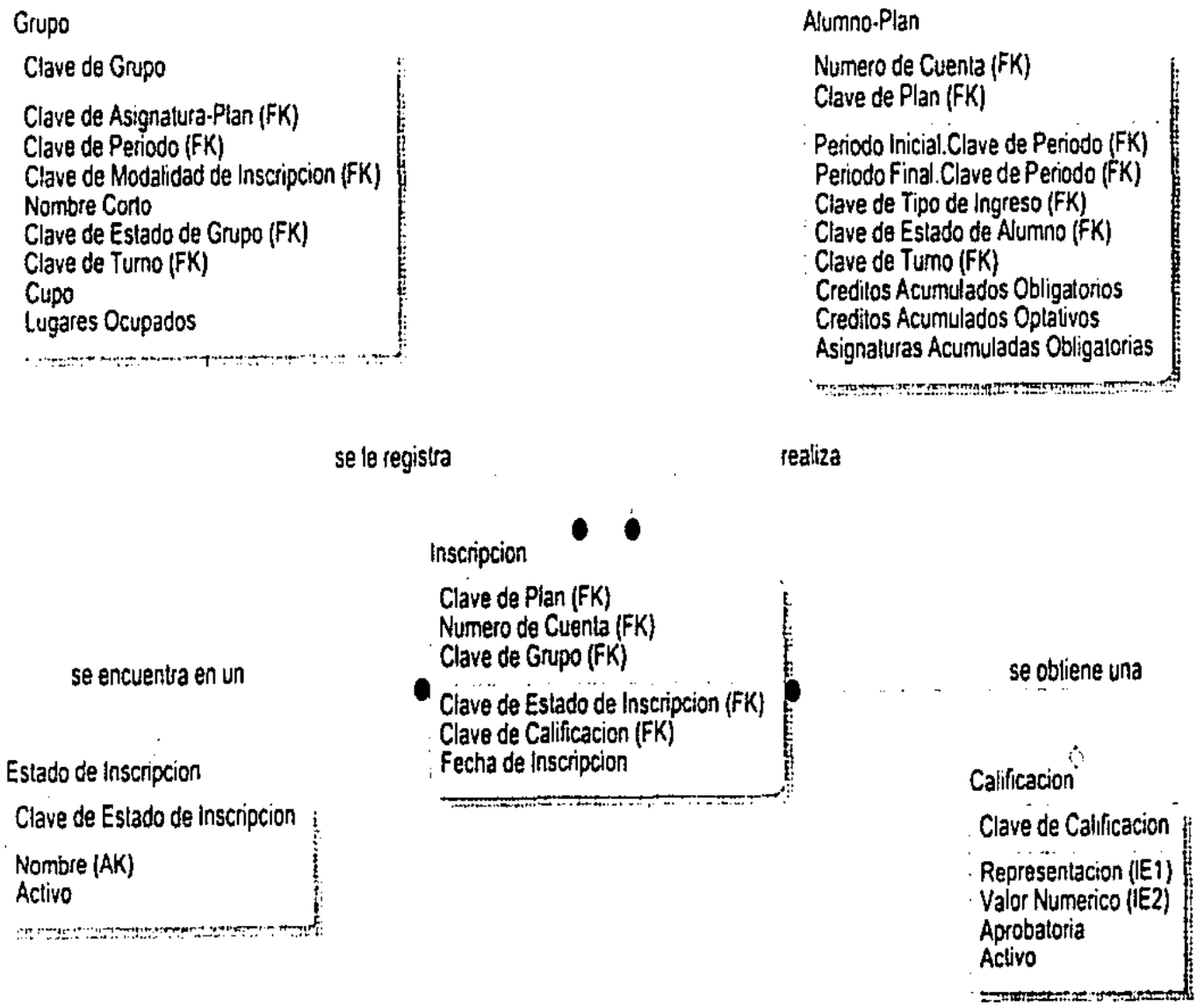
Con el objeto de que pueda apreciarse mejor, se tiene la definición de entidades, atributos y relaciones por módulo, aunque evidentemente el análisis es para todo el entorno.

4.2.2.2.2 Normalización de la base de datos.

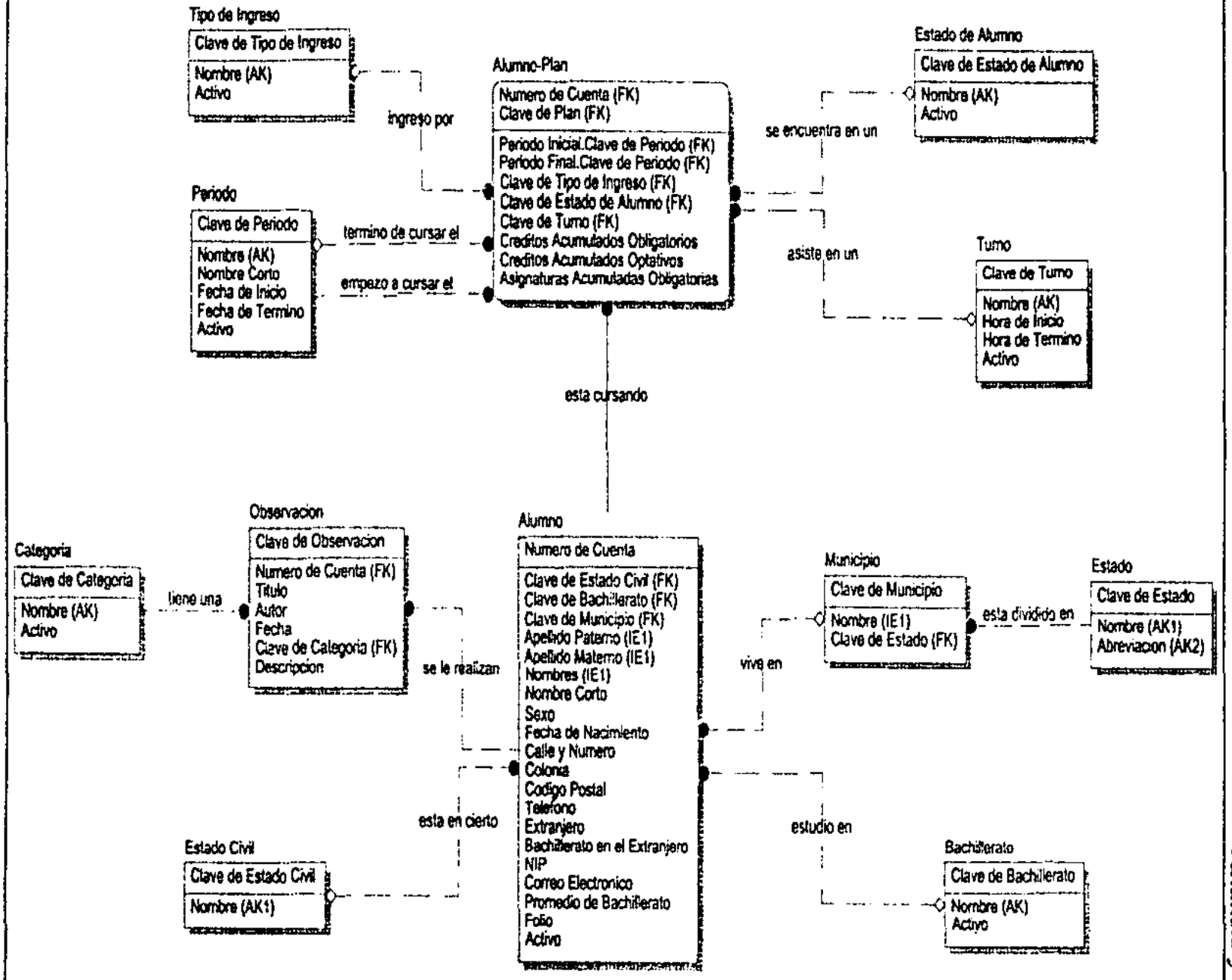
El modelo de base de datos final se encuentra en tercera forma normal, y para llegar a ésta fue necesario una serie de análisis iterativo que condujo a la versión usada en la programación. Se dejó en esta forma porque no era necesario seguir normalizando dadas las características del sistema.

En este punto se cumple que los movimientos de inserción, actualización y eliminación, se realizan de forma eficiente.

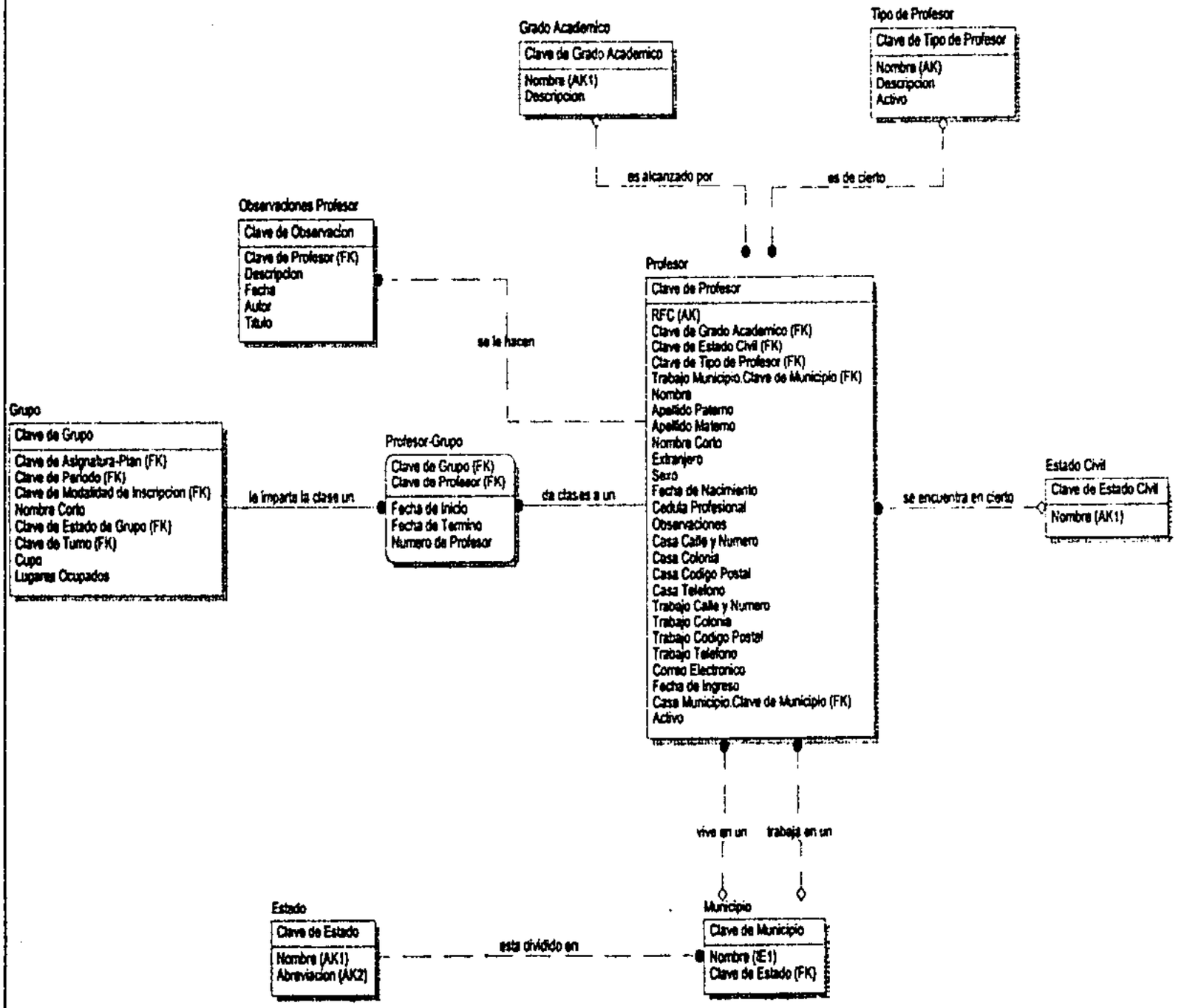
Módulo de inscripción



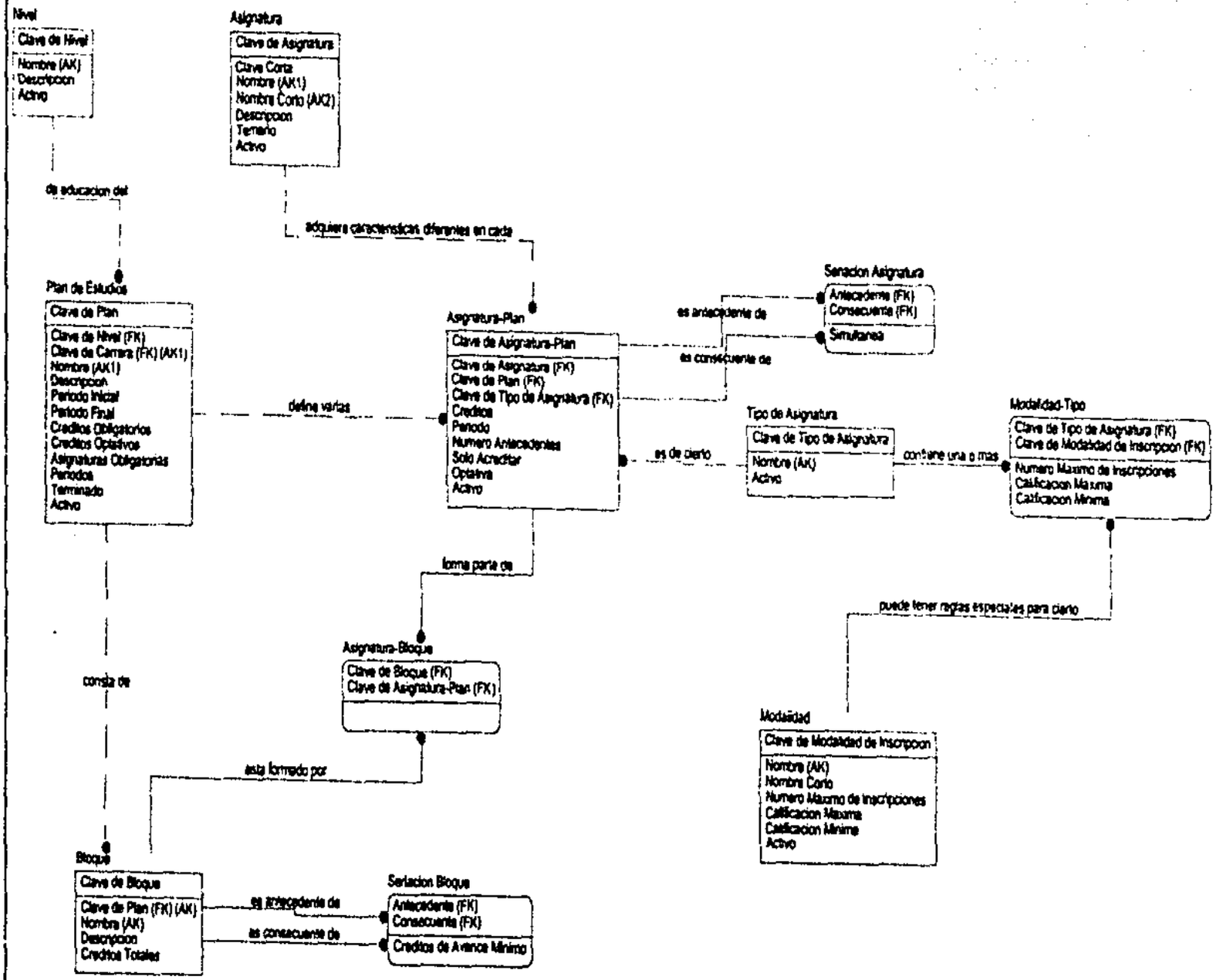
Módulo de alumnos



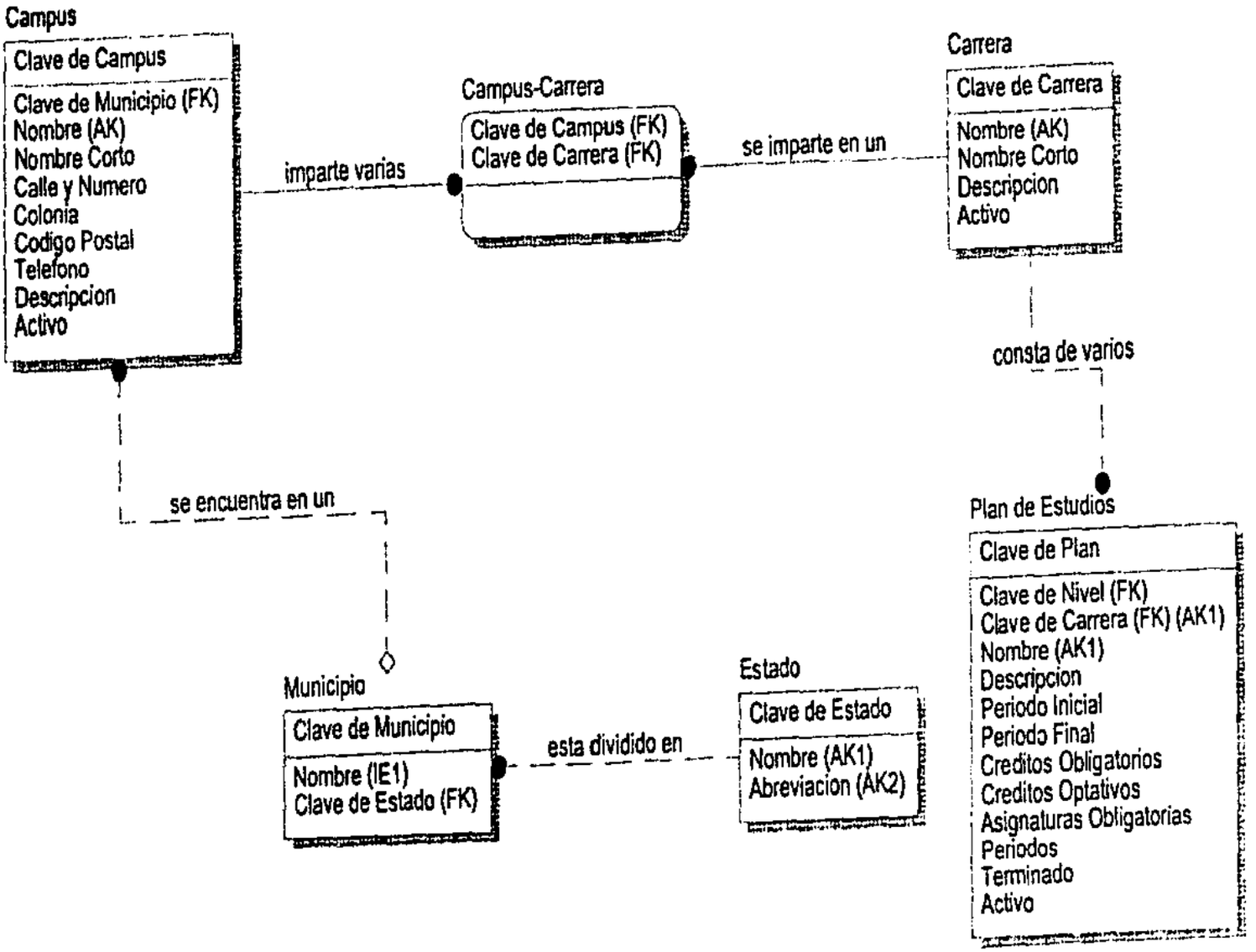
Módulo de profesor



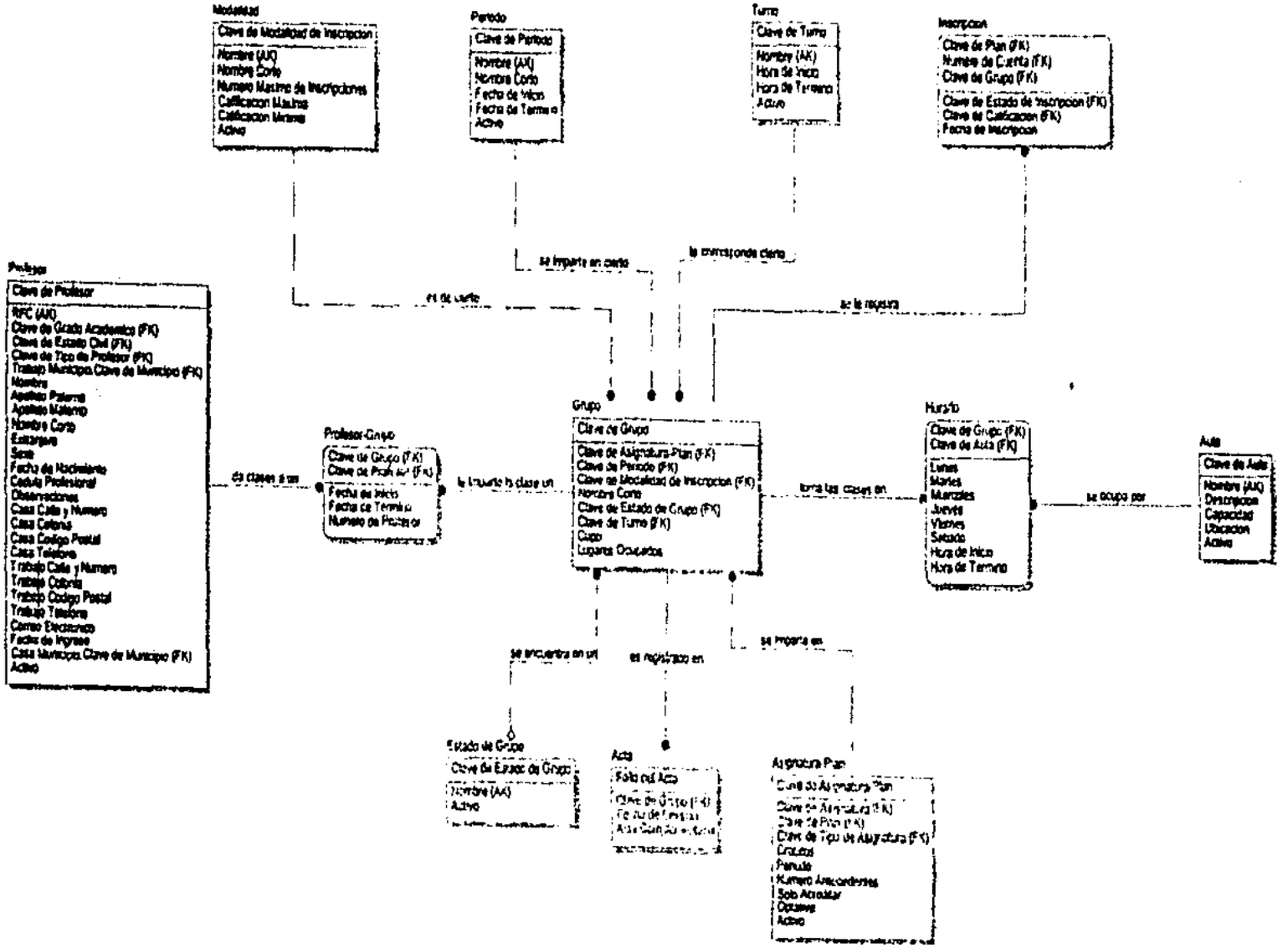
Módulo de programa (plan de estudios)



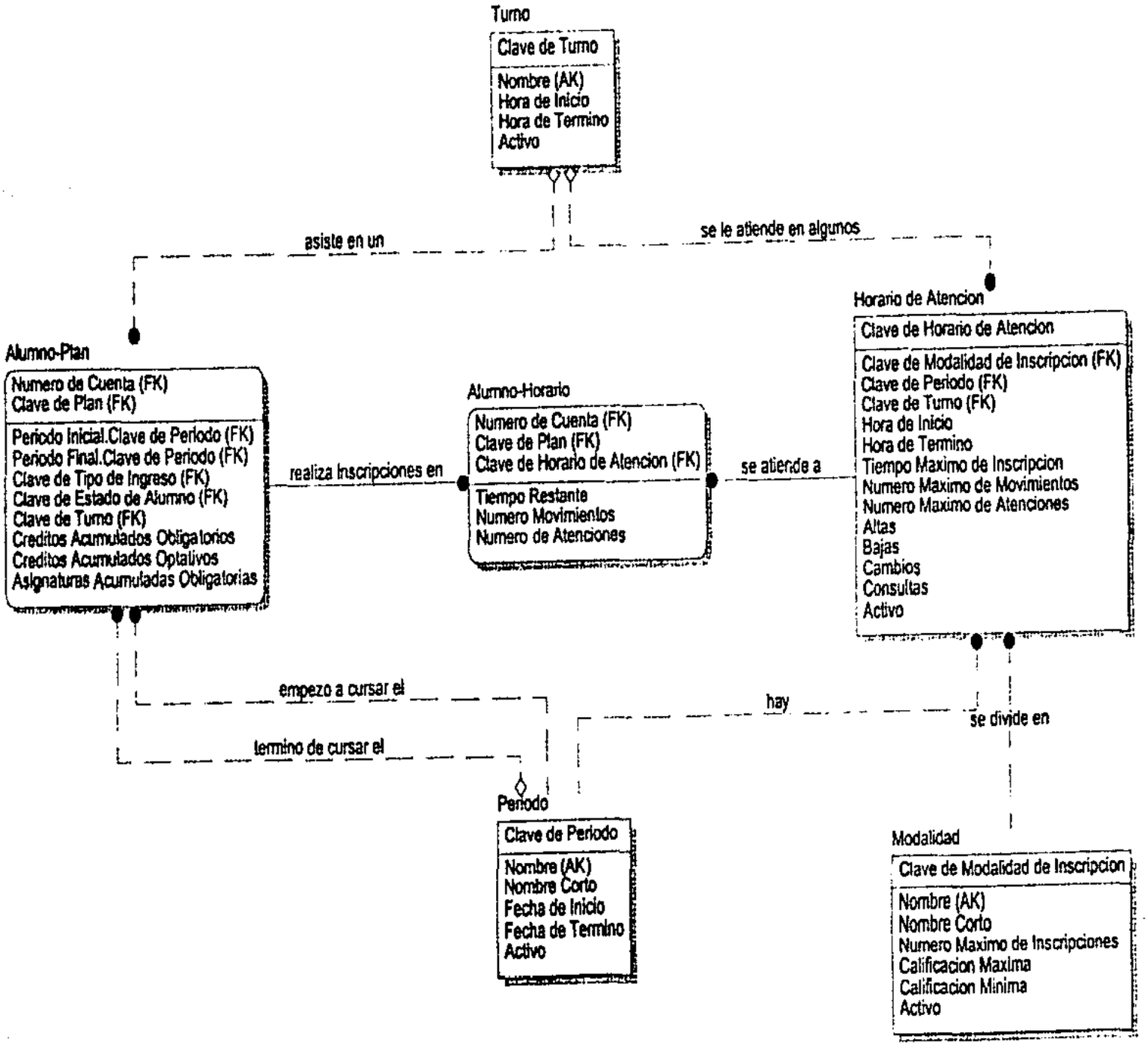
Módulo de campus-carrera



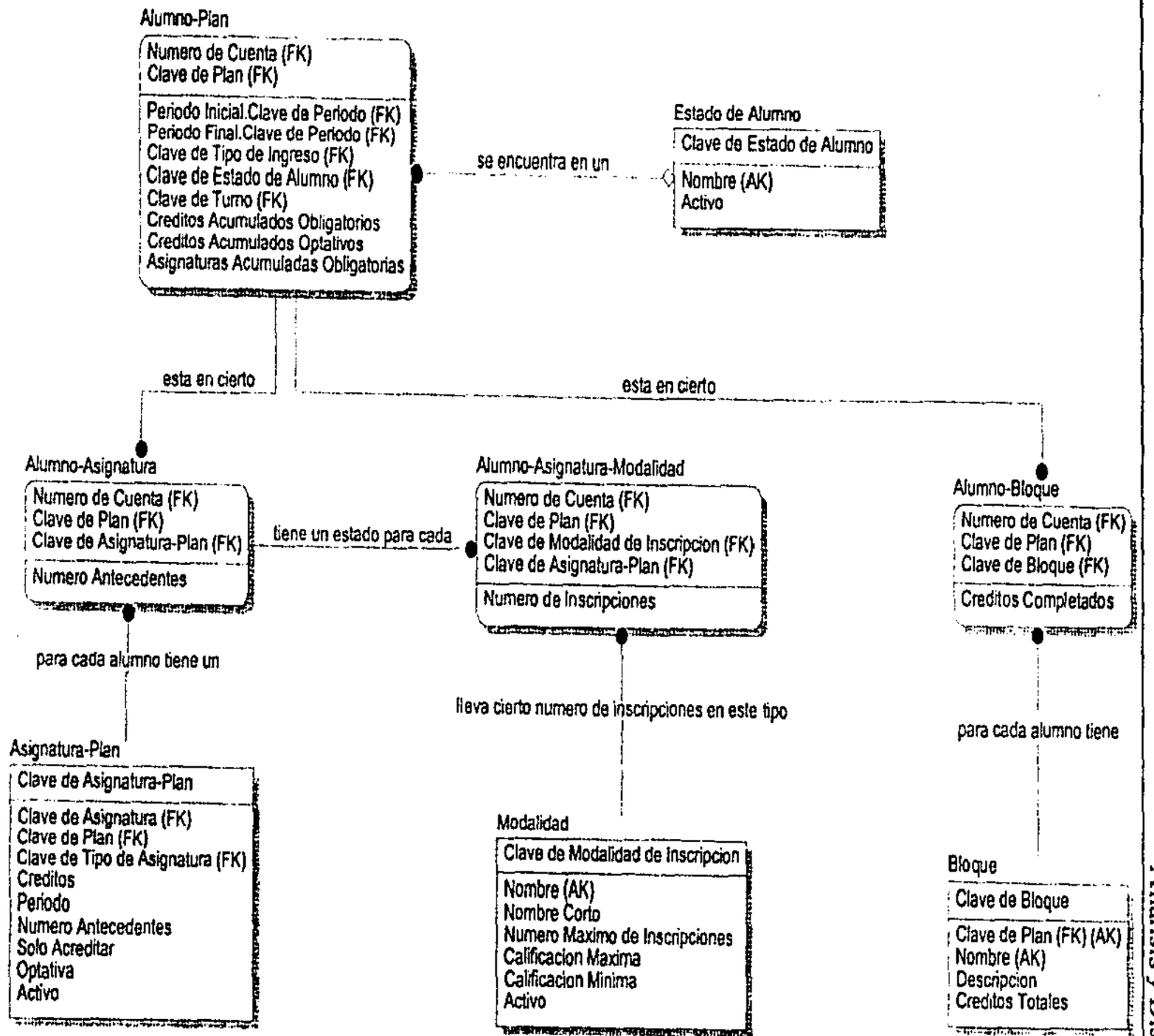
Módulo de grupo



Módulo de horario de atención



Módulo de estado de alumno



Módulo de DGAE

DGAE-Inscripcion_Entrada
- Numero de Cuenta
- Clave de Plantel
- Clave de Asignatura
- Clave de Grupo
- Filler

DGAE-ABC de Inscripcion
- Numero de Cuenta
- Clave de Plantel
- Clave de la Asignatura
- Clave de Grupo de Baja
- Clave de Grupo de Alta
- Filler

DGAE-Modificacion a Historia Academica
- Clave de Plantel
- Numero de Cuenta
- Clave de la Asignatura
- Periodo
- Tipo de Examen
- Calificacion
- Folio
- Grupo
- Tipo de Movimiento

DGAE-ABC de Grupo
- Clave de Plantel
- Clave de la Asignatura
- Grupo
- Cupo
- Profesor
- Nombre del Profesor
- RFC
- Movimiento
- Filler

DGAE-Cambio de Carrera Interna
- Clave de Plantel
- Numero de Cuenta
- Clave de Carrera Origen
- Clave de Carrera Destino

DGAE-Cambio de Unidad Academica
- Numero de Cuenta
- Clave del Plantel Origen
- Clave de Carrera Origen
- Clave del Plantel Destino
- Clave de Carrera Destino

DGAE-Baja de Inscripcion
- Clave del Plantel
- Clave de Carrera
- Numero de Cuenta
- Filler

DGAE-Resumen de Historias Academicas
- Numero de Cuenta
- MB
- B
- S
- 6
- 7
- 8
- 9
- 10
- NA
- NP
- Revalidadas
- Acreditadas
- Aprobadas en Ordinario
- Aprobadas en Extraordinario
- Reprobadas en Ordinario
- Reprobadas en Extraordinario
- Creditos Obligatorios Acumulados
- Creditos Optativos Acumulados
- Periodo Inicial
- Periodo Ultimo
- Covalidadas

DGAE-Modificacion al Registro del Alumno
- Clave del Plantel
- Numero de Cuenta
- Clave de la Carrera
- Clave de Causa de Baja

DGAE-Relacion de Asignaturas
- Nombre de la Asignatura
- Clave del Plantel
- Clave de la Asignatura
- Creditos de la Asignatura
- Semestre de la Asignatura
- Nivel
- Filler

DGAE-Relacion de Carreras
- Clave del Plantel
- Clave de la Carrera
- Nombre de la Carrera
- Nivel
- Plan
- Duracion de la Carrera
- Creditos Obligatorios
- Creditos Optativos
- Filler

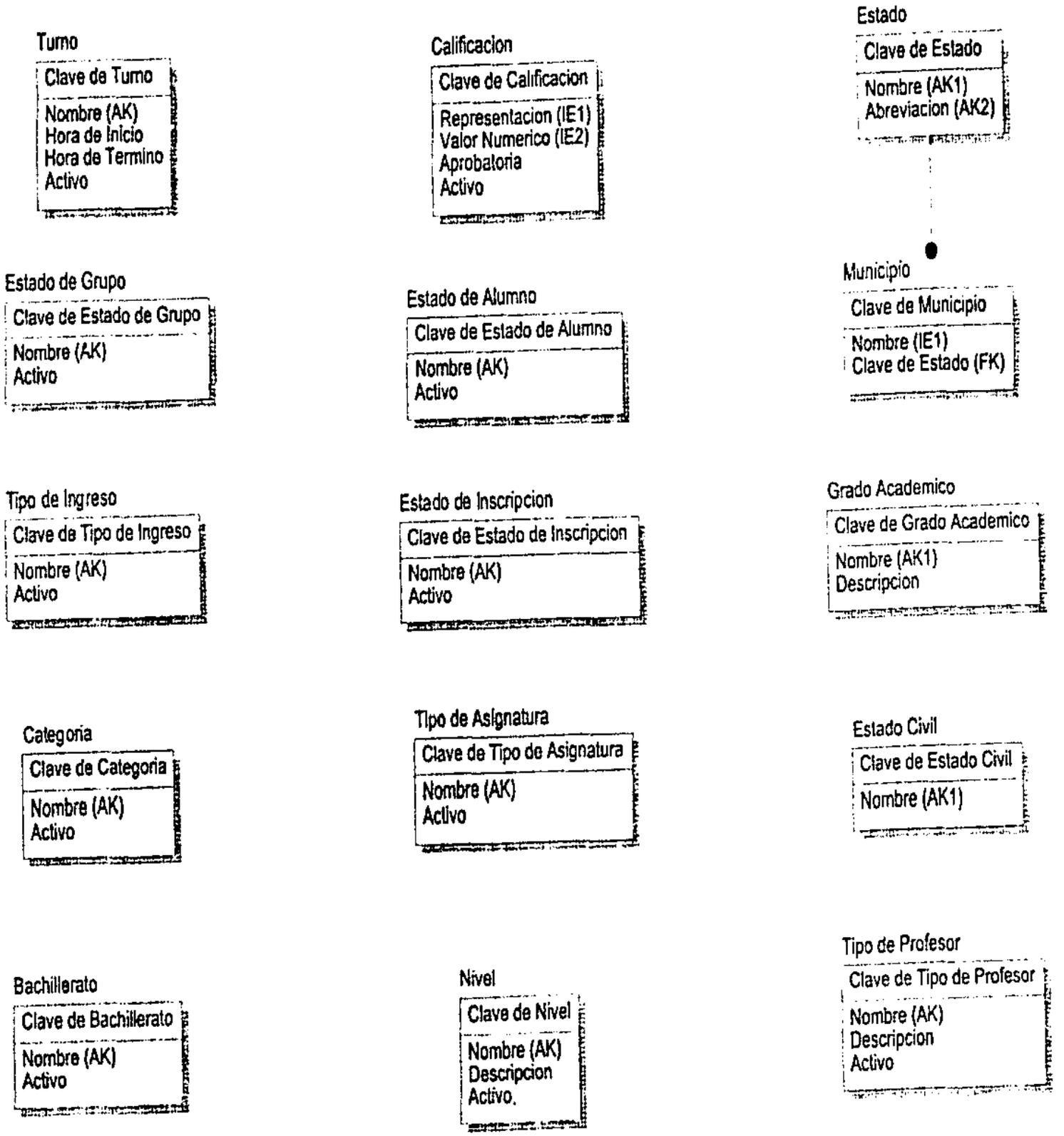
DGAE-Actas Emitidas
- Folio
- Plantel
- Asignatura
- Tipo de Examen
- Semestre Escolar
- Año Escolar
- Actas del Grupo
- Alumnos del Grupo
- Fecha de Emision
- Grupo
- Numero de Cuenta
- Calificacion
- Tipo de Acta
- Filler

DGAE-Directorio de Alumnos
- Nombre del Alumno
- Numero de Cuenta
- Clave del Plantel
- Clave de Carrera o Turno
- Año de Primer Ingreso
- Nacionalidad
- Causa de Ingreso
- Causa de Exalumno
- Sexo
- Fecha de Nacimiento
- Fecha de Movimiento
- Marca de Inscrito

DGAE-Historias Academicas
- Numero de Cuenta
- Clave del Plantel
- Clave de la Asignatura
- Año Semestre
- Calificacion
- Grupo
- Folio del Acta
- Tipo de Examen

DGAE-Inscripcion_Salida
- Numero de Cuenta
- Clave de Plantel
- Clave de la Asignatura
- Clave de Grupo
- Filler

Módulo de catálogos



4.2.2.2.3 Definición de índices y constraints.

Nombre de la entidad	Descripción	Indices	Constraints
Acta	Es el documento oficial donde se registran los alumnos que quedan inscritos en un grupo y, en su momento, sus calificaciones.	In_acta_principal Unique CLUSTERED Index Columns: folio , In_acta_grupo Index Columns: grupo	Sin constraints.
Alumno	Es una de las entidades principales del sistema, aloja la información necesaria de los alumnos candidatos a inscribirse.	In_alumno_principal Unique CLUSTERED Index Columns: cuenta , in_alumno_nombre Index Columns: paterno materno nombre, in_alumno_bachillerato Index Columns: bachillerato, in_alumno_municipio Index Columns: municipio, in_alumno_edo_civil Index Columns: edo_civil	co_apellidos, datalength (isnull(paterno "")+isnull(materno ""))>0
Alumno-Asignatura	Entidad asociativa que representa el hecho de que un alumno puede estar cursando varias asignaturas y de que una asignatura tiene muchos alumnos.	in_alumno_asig_principal Unique CLUSTERED Index Columns: cuenta programa asignatura , in_alumno_asig_asigplan Index Columns: asignatura, in_alumno_asig_alumnplan Index Columns: cuenta programa, in_alumno_asig_alumnplan Index Columns: cuenta programa	Sin constraints.
Alumno-Asignatura-Modalidad	Entidad asociativa que representa el hecho de que un alumno puede estar cursando varias asignaturas y son de varias modalidades y viceversa.	in_alum_asig_mod_principal Unique CLUSTERED Index Columns: cuenta programa modalidad asignatura , in_alum_asig_mod_modalidad Index Columns: modalidad, in_alum_asig_mod_alumnasig Index Columns: cuenta programa asignatura	Sin constraints.

Entidad	Descripción	Indices	Constraints
Alumno-Bloque	Entidad asociativa que representa el hecho de que un alumno pueda tener uno o más bloques.	in_alumno_bloque_principal Unique CLUSTERED Index Columns: cuenta programa bloque , in_alumno_bloque_bloque Index Columns: bloque, in_alumno_bloque_alumno Index Columns: cuenta programa	Sin constraints.
Alumno-Horario	Entidad asociativa que representa el hecho de que un alumno pueda tener uno o más horarios de atención.	in_alumno_horario_principal Unique CLUSTERED Index Columns: cuenta programa horario , in_alumno_horario_horario Index Columns: horario, in_alumno_horario_alumno Index Columns: cuenta programa	Sin constraints.
Alumno-Plan	Entidad asociativa que representa el hecho de que un alumno esté cursando uno o más planes de estudios y de que un plan de estudio puede tener uno o muchos alumnos.	in_alumno_plan_principal Unique CLUSTERED Index Columns: cuenta programa , in_alumno_plan_alumno Index Columns: cuenta, in_alumno_plan_plan Index Columns: programa, in_alumno_plan_estado Index Columns: estado, in_alumno_plan_turno Index Columns: turno, in_alumno_plan_ingreso Index Columna: ingreso, in_alumno_plan_perinicio Index Columns: inicio, in_alumno_plan_pertermino Index Columns: termino	Sin constraints.

Entidad	Descripción	Indices	Constraints
Asignatura	Entidad que define las características de las materias que se imparten en cierta escuela.	in_asignatura_principal Unique CLUSTERED Index Columns: clave , in_asignatura_nombre Unique Index Columns: nombre, in_asignatura_nombre_corto Unique Index Columns: nombre_corto,	Sin constraints.
Asignatura-Bloque	Entidad asociativa que representa el hecho de que un bloque pueda tener uno o más asignaturas y viceversa.	in_asignatura_bloque_principal Unique CLUSTERED Index Columns: bloque asignatura , in_asignatura_bloque_asigatura Index Columns: asignatura, in_asignatura_bloque_bloque Index Columns: bloque	Sin constraints.
Asignatura-Plan	Entidad asociativa que representa el hecho de que una asignatura pueda tener uno o más planes de estudio y viceversa.	in_asignatura_plan_principal Unique CLUSTERED Index Columns: clave , in_asignatura_plan_plan Index Columns: programa, in_asignatura_plan_asignatura Index Columns: asignatura, in_asignatura_plan_tipo Index Columns: tipo	Sin constraints.
Aula	Es el lugar donde se imparte la cátedra.	in_aula_principal Unique CLUSTERED Index Columns: clave , in_aula_nombre Unique Index Columns: nombre	Sin constraints.
Bachillerato	Lugar donde se realizaron los estudios anteriores a la licenciatura.	in_bachillerato_principal Unique CLUSTERED Index Columns: clave , in_bachillerato_nombre Unique Index Columns: nombre	Sin constraints.
Bloque	Entidad que controla la seriación existente entre asignaturas de un plan de estudios.	in_bloque_principal Unique CLUSTERED Index Columns: clave , in_bloque_nombre_plan Unique Index Columns: nombre programa, in_bloque_plan Index Columns: programa,	Sin constraints.

Entidad	Descripción	Indices	Constraints
Calificación	Catálogo de calificaciones que pueden otorgarse a un alumno en una asignatura.	in_calificacion_principal Unique CLUSTERED Index Columns: clave , in_calificacion_representacion Index Columns: representacion, in_calificacion_valor Index Columns: valor	Sin constraints.
Campus	Es el lugar donde se albergan las instalaciones universitarias.	in_campus_principal Unique CLUSTERED Index Columns: clave , in_campus_nombre Unique Index Columns: nombre, in_campus_municipio Index Columns: municipio,	Sin constraints.
Campus-Carrera	Entidad asociativa que representa el hecho de que un campus puede tener varias carreras y viceversa.	in_campus_carrera_principal Unique CLUSTERED Index Columns: campus carrera , in_campus_carrera_carrera Index Columns: carrera, in_campus_carrera_campus Index Columns: campus	Sin constraints.
Carrera	Representa todas las carreras posibles que se pueden impartir en una escuela.	in_carrera_principal Unique CLUSTERED Index Columns: clave , in_carrera_nombre Unique Index Columns: nombre	Sin constraints.
Categoría	La categoría me define el grado de severidad de una observación hecha a un alumno.	in_categoria_principal Unique CLUSTERED Index Columns: clave , in_categoria_nombre Unique Index Columns: nombre	Sin constraints.
DGAE-ABC de Grupo	Tabla de entrada a DGAE que contiene información de alta de grupo y/o alta baja o cambio de profesor.	No tiene índices.	Sin constraints.
DGAE-ABC de Inscripcion	Tabla de entrada a DGAE que contiene altas bajas y cambios de inscripción.	No tiene índices.	Sin constraints.

Entidad	Descripción	Indices	Constraints
DGAE-Actas Emitidas	Tabla de salida de DGAE que tiene la información de actas emitidas.	No tiene índices.	Sin constraints.
DGAE-Baja de Inscripción	Tabla de entrada a DGAE que contiene la información de baja de inscripción.	No tiene índices.	Sin constraints.
DGAE-Cambio de Carrera Interna	Tabla de entrada a DGAE de cambio interno de carrera.	No tiene índices.	Sin constraints.
DGAE-Cambio de Unidad Académica	Tabla de entrada a DGAE que contiene el cambio de unidad académica.	No tiene índices.	Sin constraints.
DGAE-Directorio de Alumnos	Tabla de salida de DGAE que tiene la información de alumnos y exalumnos.	No tiene índices.	Sin constraints.
DGAE-Historias Académicas	Tabla de salida de DGAE que contiene la información del historial académico.	No tiene índices.	Sin constraints.
DGAE-Inscripción _Entrada	Tabla de entrada a DGAE que tiene las asignaturas que un alumno ha cursado y las que está actualmente cursando.	No tiene índices.	Sin constraints.
DGAE-Inscripción _Salida	Tabla de salida de DGAE que tiene las asignaturas que un alumno ha cursado y las que está actualmente cursando.	No tiene índices.	Sin constraints.
DGAE-Modificación a Historia Académica	Tabla de entrada a DGAE que contiene información de modificación a la historia académica.	No tiene índices.	Sin constraints.

Entidad	Descripción	Indices	Constraints
DGAE-Modificación al Registro del Alumno	Tabla de entrada a DGAE que tiene cambios al registro de los alumnos.	No tiene índices.	Sin constraints.
DGAE-Relación de Asignaturas	Tabla de salida de DGAE que contiene la relación de asignaturas.	No tiene índices.	Sin constraints.
DGAE-Relación de Carreras	Tabla de salida de DGAE que contiene la relación de carreras.	No tiene índices.	Sin constraints.
DGAE-Resumen de Historias Académicas	Tabla de salida de DGAE que contiene el resumen de la historia académica.	No tiene índices.	Sin constraints.
Estado	Es el estado del país al que pertenece cierta dirección.	in_estado_principal Unique CLUSTERED Index Columns: clave , in_estado_nombre Unique Index Columns: nombre, in_estado_abreviacion Unique Index Columns: abreviacion	Sin constraints.
Estado Civil	Es el estado civil en que se encuentre una persona: soltero, casado, divorciado, viudo o unión libre.	in_civil_principal Unique Index Columns: clave , in_civil_nombre Unique CLUSTERED Index Columns: nombre	Sin constraints.
Estado de Alumno	Categoría en la que se encuentra un alumno.	in_estado_alumno_principal Unique CLUSTERED Index Columns: clave , in_estado_alumno_nombre Unique Index Columns: nombre	Sin constraints.
Estado de Grupo	Estado en que se puede encontrar un grupo v.gr. cerrado activo.	in_estado_grupo_principal Unique CLUSTERED Index Columns: clave , in_estado_grupo_nombre Unique Index Columns: nombre.	Sin constraints.

Entidad	Descripción	Indices	Constraints
Estado de Inscripción	Estado en el que se puede encontrar una inscripción.	in_edo_inscripcion_principal Unique CLUSTERED Index Columns: clave , in_estado_inscripcion_nombre Unique Index Columns: nombre	Sin constraints.
Grado Académico	Es el grado académico máximo que ha alcanzado un profesor.	in_grado_principal Unique Index Columns: clave , in_grado_nombre Unique CLUSTERED Index Columns: nombre	Sin constraints.
Grupo	Profesor asignado a un conjunto de alumnos para darles clases de una asignatura.	in_grupo_principal Unique CLUSTERED Index Columns: clave , in_grupo_periodo Index Columns: periodo, in_grupo_modalidad Index Columns: modalidad, in_grupo_turno Index Columns: turno, in_grupo_asignatura Index Columns: asignatura, in_grupo_estado Index Columns: estado	co_cupo, ocupados<=cupo
Horario	Datos de rango de horas y días en los que se imparte una asignatura.	in_horario_principal Unique CLUSTERED Index Columns: grupo aula , in_horario_grupo Index Columns: grupo, in_horario_aula Index Columns: aula,	co_hora, inicio<término
Horario de Atención	Rango de horas de atención para la inscripción de asignaturas de una modalidad dada.	in_horario_principal Unique Index Columns: clave , in_horario_modalidad Index Columns: modalidad, in_horario_periodo Index Columns: periodo, in_horario_turno Index Columns: turno,	co_hora, inicio<término

Entidad	Descripción	Indices	Constraints
Inscripción	Es una de las entidades principales que controla las asignaturas que un alumno ha cursado y las que está actualmente cursando.	in_inscripcion_principal Unique CLUSTERED Index Columns: programa cuenta grupo , in_inscripcion_calificacion Index Columns: calificacion, in_inscripcion_grupo Index Columns: grupo, in_inscripcion_estado Index Columns: estado, in_inscripcion_alumno Index Columns: cuenta programa	Sin constraints.
Modalidad	Tipo de inscripción que puede hacerse.	in_modalidad_principal Unique CLUSTERED Index Columns: modalidad , in_modalidad_nombre Unique Index Columns: nombre	co_calificación, mínima<=máxima
Modalidad-Tipo	Entidad asociativa que representa el hecho de que una modalidad pueda tener varios tipos de asignatura y viceversa.	in_modalidad_tipo_principal Unique CLUSTERED Index Columns: tipo modalidad , in_modalidad_tipo_modalidad Index Columns: modalidad, in_modalidad_tipo_asignatura Index Columns: tipo	co_calificación, mínima<=máxima
Municipio	Es el municipio físico al cual pertenece una dirección.	in_municipio_principal Unique CLUSTERED Index Columns: clave , in_municipio_nombre Index Columns: nombre, in_municipio_estado Index Columns: estado	Sin constraints.
Nivel	Nivel al que pertenece cierto plan de estudios.	in_nivel_principal Unique CLUSTERED Index Columns: clave , in_nivel_nombre Unique Index Columns: nombre	Sin constraints.

Entidad	Descripción	Indices	Constraints
Observación	Observaciones diversas que pueden hacerse a un alumno como reconocimientos o sanciones.	in_observacion_principal Unique Index Columns: clave , in_observacion_alumno Index Columns: cuenta, in_observacion_categoria Index Columns: categoria	Sin constraints.
Observaciones Profesor	Observaciones que se le hacen al profesor ya sean reconocimientos o sanciones.	in_obs_prof_principal Unique CLUSTERED Index Columns: clave , in_obs_prof_profesor Index Columns: profesor	Sin constraints.
Parámetro	Parámetro	XPKparametros Unique CLUSTERED Index Columns: clave	Sin constraints.
Periodo	Rango de fechas en las que se imparte un semestre.	in_periodo_principal Unique CLUSTERED Index Columns: clave , in_periodo_nombre Unique Index Columns: nombre	co_hora, inicio<término
Plan de Estudios	Conjunto de asignaturas que conforman una carrera.	in_plan_principal Unique CLUSTERED Index Columns: clave , in_plan_nombre Unique Index Columns: nombre carrera, in_plan_carrera Index Columns: carrera, in_plan_nivel Index Columns: nivel	co_hora, inicio<termino
Profesor	Personas que imparten las clases de una asignatura.	profesor_principal Unique CLUSTERED Index Columns: clave , in_profesor_rfc Unique Index Columns: rfc, in_profesor_tipo Index Columns: tipo, in_profesor_trabmunicipio Index Columns: trab_municipio, in_profesor_municipio Index Columns: municipio, in_profesor_civil Index Columns: edo_civil, in_profesor_grado Index Columns: grado	co_nacimiento, nacimiento <ingreso

Entidad	Descripción	Indices	Constraints
Profesor-Grupo	Entidad asociativa que representa el hecho de que un profesor puede impartir clases a varios grupos y que un grupo puede darles clases varios profesores.	in_profesor_grupo_principal Unique CLUSTERED Index Columns: grupo profesor , in_profesor_grupo_grupo Index Columns: grupo, in_profesor_grupo_profesor Index Columns: profesor	co_hora, inicio<termino
Seriación Asignatura	Entidad que maneja la seriación entre asignaturas.	in_s_asignatura_principal Unique CLUSTERED Index Columns: antecedente consecuente , in_s_asignatura_consecuente Index Columns: consecuente, in_s_asignatura_antecedente Index Columns: antecedente	co_ser_asignatura, antecedente<> consecuente
Seriación Bloque	Entidad que maneja la seriación entre bloques.	in_s_bloque_principal Unique CLUSTERED Index Columns: antecedente consecuente , in_s_bloque_consecuente Index Columns: consecuente, in_s_bloque_antecedente Index Columns: antecedente	co_seriacionbloque , (a_plan=c_plan) and (a_bloque<> c_bloque)
Tipo de Asignatura	Tipo de asignatura. Por ejemplo: laboratorio, teórica, de campo etc.	in_tipo_asignatura_principal Unique CLUSTERED Index Columns: clave , in_tipo_asignatura_nombre Unique Index Columns: nombre	Sin constraints.
Tipo de Ingreso	Forma en que se ingresó al nivel escolar actual.	in_ingreso_principal Unique CLUSTERED Index Columns: clave , in_ingreso_nombre Unique Index Columns: nombre	Sin constraints.
Tipo de Profesor	Indica los tipos de profesores que reconoce la escuela. Ej. asignatura, medio tiempo, titular A, asociado B, técnico académico, etc.	in_tipo_profesor_principal Unique CLUSTERED Index Columns: clave , in_tipo_profesor_nombre Unique Index Columns: nombre	Sin constraints.

	A, asociado B, técnico académico, etc.	Columns: nombre	
Entidad	Descripción	Indices	Constraints
Turno	Horario de clases de una asignatura.	in_turno_principal Unique CLUSTERED Index Columns: clave , in_turno_nombre Unique Index Columns: nombre	co_hora, inicio<termino

4.2.2.2.4 Definición de triggers.

El objetivo de los triggers es garantizar la integridad de los datos. Para cualquier evento que ocasione un cambio en una base de datos, se puede especificar una acción que ejecutará el DBMS en ese momento. Los tres casos que pueden disparar una acción son intentos de insertar, borrar o actualizar registros de una tabla. Al conjunto de secuencias SQL que la conforman se le denomina trigger y se utilizan para implementar las reglas de integridad referencial.

La manera como funciona un trigger consiste en definir un conjunto de instrucciones SQL, que al ocurrir un evento se ejecutan, y abortan un movimiento a la base de datos en caso de que no se cumplan las reglas en él establecidas.

Además de la integridad referencial, los triggers se pueden utilizar para definir las reglas del negocio en un sistema y asegurar que se cumplan, reduciendo así, la complejidad de los programas que lo conforman.

En el sistema de inscripciones existen dos tipos de triggers: los creados exclusivamente para garantizar la integridad referencial y los encargados de cuidar que se cumplan las reglas de la institución que el sistema maneja.

El contenido total de los triggers viene en el Apéndice B.

A continuación se presenta una lista de los triggers de reglas de negocio:

- Asegurar que las tablas cuyo campo "activo" esté desactivado, no puedan usarse por otras tablas.
- Validar que sólo existan relaciones y movimientos entre datos del mismo plan de estudios.
- Sólo permitir modificaciones entre datos del mismo bloque.
- Si hay alumnos inscritos en la tabla "alumno_plan", no modificar las tablas relacionadas con ese plan.
- No hacer movimientos a planes de estudios terminados.
- No permitir asignar créditos manualmente.

4.2.2.2.5 Definición de stored procedures.

Los procedimientos almacenados, a diferencia de los triggers, no se disparan automáticamente, en algún programa se indica su ejecución en cierto momento. Tienen la característica de que una vez que se ejecutan por primera vez residen en memoria, logrando con esto mayor velocidad. Además de ejecutar acciones del sistema de forma efectiva, vía instrucciones SQL, sirven para reducir el tráfico de red.

El sistema contempla seis procedimientos almacenados en el módulo principal de inscripciones, orientados al cumplimiento de las validaciones establecidas en el Reglamento General de Inscripciones de la UNAM y en las disposiciones internas del consejo técnico de la Facultad de

Cada tabla corresponde a un procedimiento almacenado, mismo que se ejecutará al estar en la pantalla de inscripciones del sistema, realizando la función específica, pudiendo ser altas, bajas o cambios.

- **Altas.**

No.	Validaciones del alumno
1	Su número de cuenta exista.
2	No se haya inscrito previamente.
3	El tiempo de inscripción en la UNAM no exceda en 50% lo recomendado para cursarse.
4	Tenga horario de inscripción.
5	Se esté inscribiendo dentro de su horario de atención.

No.	Validaciones de la asignatura
1	Exista.
2	Corresponda al plan de estudios del alumno.
3	No haya sido previamente acreditada.
4	No haya sido inscrita dos veces en ordinario.
5	El alumno debió haber cursado las asignaturas antecedentes.

No.	Validaciones de grupo.
1	Exista.
2	No debe existir traslape entre las asignaturas.
3	Exista cupo.

- **Bajas.**

1	Su número de cuenta exista.
2	Tenga horario de inscripción.
3	Se esté inscribiendo dentro de su horario de atención.

- **Cambios.**

1	Su número de cuenta exista.
2	Tenga horario de inscripción.
3	Se esté inscribiendo dentro de su horario de atención.

1	Debe haber cupo en el grupo destino.
2	No debe existir traslape con los horarios de las asignaturas previamente inscritas.

En cualquiera de los casos, si se cumplen las validaciones, se procede con la acción deseada.

4.2.2.2.6 Cálculo del tamaño de la base de datos.

Algo de suma importancia en la implantación del sistema es saber el tamaño que ocupará la base de datos, y cómo irá modificándose con el tiempo de acuerdo al uso normal. Existen varios mecanismos para realizar esta labor. Los dos que se consideraron en este trabajo, consisten en:

Método 1

1. - Calcular el tamaño en bytes que ocupa cada registro de cada tabla y multiplicarlo por el número de registros esperados promedio.

2. - Sumar todos los cálculos anteriores y darle un margen de sobra de cierto porcentaje, v.gr. 25%.

Método 2

Utilizar un procedimiento almacenado propio del manejador de base de datos, cuya estructura es:

Sp_estspace table_name, no_of_rows, fill_factor, cols_to_max, textbin_len, iosec

Donde:

table_name: Nombre de la tabla.
no_of_rows: Número de registros.
fill_factor: Factor de llenado del indice.
cols_to_max: Lista de las columnas de longitud variable a su máximo valor.
textbin_len: Longitud de texto y campos binarios por renglón,
iosec: Entradas y salidas por segundo en la computadora donde resida la base.

Esto se debe aplicar para todas las tablas del sistema y sumar los resultados parciales.

Con la idea de ser más precisos en el cálculo se utilizó la segunda opción, con los siguientes resultados.

Concepto	Resultados
Número de tablas	60
Tamaño global	128 Mb
Margen de crecimiento	100 %
Tamaño de la base considerando el margen	256 Mb
Tamaño del log de transacciones (20 % de la B.D.)	52 Mb
Tamaño final	**308 Mb**

Capítulo V

Implementación

5.1

Generación de la base de datos en SYBASE

En esta parte se detalla el proceso de creación de la base de datos desde su definición en el CASE hasta su implementación en Sybase.

5.1.1 Especificación de entidades.

La siguiente tabla concentra la información de las entidades, atributos, llaves, tipos de datos, nulidad y descripción de los campos.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
Acta	Folio del Acta	(PK)	numeric(7)	NOT NULL	Es un número único de identificación del acta.
	Clave de Grupo	(FK)	numeric(16)	NOT NULL	Identificador único del grupo.
	Fecha de emisión		smalldatetime	NULL	Es la fecha en que se emite el acta.
	Acta complementaria		bit	NOT NULL	Es un indicador de que el acta en cuestión es complementaria, es decir, forma parte de otra acta.
Alumno	Número de Cuenta	(PK)	numeric(9)	NOT NULL	Identificador único oficial del alumno.
	Clave de estado civil	(FK)	numeric(4)	NULL	Es un identificador único del estado civil.
	Clave de bachillerato	(FK)	numeric(4)	NULL	Identificador único del bachillerato.
	Clave de municipio	(FK)	numeric(8)	NULL	Es un identificador único del municipio.
	Apellido paterno	(IE1)	varchar(20)	NULL	Apellido paterno del alumno.
	Apellido materno	(IE1)	varchar(20)	NULL	Apellido materno del alumno.
	Nombres	(IE1)	varchar(25)	NOT NULL	Nombre (s) del alumno.
	Nombre corto		varchar(32)	NOT NULL	Nombre, apellido paterno y materno del alumno.
	Sexo		bit	NOT NULL	Sexo del alumno.
	Fecha de nacimiento		smalldatetime	NULL	Fecha de nacimiento del alumno.
	Calle y número		varchar(40)	NULL	Calle y número del domicilio del alumno.
	Colonia		varchar(30)	NULL	Colonia del domicilio del alumno.
	Código postal		numeric(5)	NULL	Código postal del domicilio del alumno.
	Teléfono		numeric(12)	NULL	Teléfono del domicilio del alumno.
	Extranjero		bit	NOT NULL	Indicador de que el alumno es extranjero.
	Bachillerato en el extranjero		bit	NOT NULL	Indicador de que el alumno realizó su bachillerato en el extranjero.
	NIP		numeric(6)	NULL	Número de identificación personal del alumno que usa para trámites vía electrónica.
	Correo electrónico		varchar(30)	NULL	Clave de correo electrónico de internet.
	Promedio de bachillerato		numeric(4,2)	NULL	Promedio obtenido en el bachillerato.
	Folio		char(5)	NULL	Folio que solicita DGAE.
	Activo		bit	NOT NULL	Indicador de que el alumno está activo.
Alumno asignatura	Número de cuenta	(PK)	numeric(9)	NOT NULL	Identificador único oficial del alumno.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Clave de plan	(PK)	numeric(8)	NOT NULL	Identificador único de clave de plan.
	Clave de asignatura plan	(PK)	numeric(8)	NOT NULL	Identificador único de la asignatura-plan.
	Número antecedentes		Tinyint	NULL	Número de asignaturas que deben ser cursadas con anterioridad.
Alumno asignatura modalidad	Número de cuenta	(PK)	Numeric(9)	NOT NULL	Identificador único oficial del alumno.
	Clave de plan	(PK)	Numeric(8)	NOT NULL	Identificador único de la clave de plan.
	Clave de modalidad de inscripción	(PK)	Numeric(4)	NOT NULL	Es el tipo de inscripción que se realiza: ordinario, extraordinario y acuerdo de pasantes.
	Clave de asignatura plan	(PK)	numeric(8)	NOT NULL	Identificador único de la asignatura plan.
	Número de inscripciones		tinyint	NULL	Cantidad de inscripciones a asignaturas que se han realizado.
Alumno bloque	Número de cuenta	(PK)	numeric(9)	NOT NULL	Identificador único oficial del alumno.
	Clave de plan	(PK) (FK)	numeric(8)	NOT NULL	Identificador único de la clave de plan.
	Clave de bloque	(PK) (FK)	numeric(8)	NOT NULL	Identificador único de la clave de bloque.
	Créditos completados		smallint	NULL	Cantidad de créditos que se deben completar para terminar el bloque.
Alumno horario	Número de cuenta	(PK)	numeric(9)	NOT NULL	Identificador único oficial del alumno.
	Clave de plan	(PK)	numeric(8)	NOT NULL	Identificador único de la clave de plan.
	Clave de horario de atención	(PK)	numeric(8)	NOT NULL	Identificador único de la clave del horario de atención.
	Tiempo restante		smalldatetime	NULL	Tiempo que le queda a un alumno para hacer movimientos de inscripción.
	Número de movimientos		tinyint	NULL	Número de movimientos de inscripción que hizo un alumno.
	Número de atenciones		tinyint	NULL	Número de atenciones que realizó un alumno.
Alumno plan	Número de cuenta	(PK)	numeric(9)	NOT NULL	Identificador único oficial del alumno.
	Clave de plan	(PK)	numeric(8)	NOT NULL	Identificador único de clave de plan.
	Periodo inicial	(FK)	numeric(8)	NOT NULL	Identificador único de la clave de periodo inicial.
	Periodo final	(FK)	numeric(8)	NULL	Identificador único de la clave de periodo final.
	Clave de tipo de ingreso	(FK)	numeric(4)	NULL	Identificador único de la forma de ingreso.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Clave de estado de alumno	(FK)	numeric(4)	NULL	Estado en que se puede encontrar un alumno.
	Clave de turno	(FK)	numeric(4)	NULL	Identificador único del turno.
	Créditos acumulados obligatorios		smallint	NULL	Créditos acumulados obligatorios que ha cursado el alumno.
	Créditos acumulados optativos		smallint	NULL	Créditos acumulados optativos que ha cursado el alumno.
	Asignaturas acumuladas obligatorias		tinyint	NULL	Número de asignaturas acumuladas obligatorias que ha cursado el alumno.
Asignatura	Clave de asignatura	(PK)	numeric(8)	IDENTITY	Identificador único de la asignatura.
	Clave corta		numeric(4)	NULL	Clave que asigna DGAE a la asignatura.
	Nombre	(AK1)	varchar(50)	NOT NULL	Nombre de la asignatura.
	Nombre corto	(AK2)	varchar(28)	NOT NULL	Nombre que asigna DGAE a la asignatura.
	Descripción		text	NULL	Texto descriptivo de la asignatura.
	Temario		text	NULL	Temas que se estudian en la asignatura.
	Activo		bit	NOT NULL	Indicador de que una asignatura está activa para impartirse.
Asignatura-bloque	Clave de bloque	(PK)	numeric(8)	NOT NULL	Identificador único de la clave de bloque.
	Clave de asignatura plan	(PK)	numeric(8)	NOT NULL	Identificador único de la asignatura-plan.
Asignatura plan	Clave de asignatura plan	(PK)	numeric(8)	IDENTITY	Identificador único de la asignatura plan.
	Clave de asignatura	(FK)	numeric(8)	NOT NULL	Identificador único de la asignatura.
	Clave de plan	(FK)	numeric(8)	NOT NULL	Identificador único de clave de plan.
	Clave de tipo de asignatura	(FK)	numeric(4)	NOT NULL	Identificador único de la clave de tipo de asignatura.
	Créditos		smallint	NULL	Créditos que se obtienen al cursar la asignatura.
	Periodo		tinyint	NULL	Periodo en el que se cursa la materia.
	Número de antecedentes		tinyint	NULL	Número de materias antecedentes que debe tener una asignatura.
	Sólo acreditar		bit	NOT NULL	Indicador de que sólo es necesario acreditar la asignatura sin una calificación.
	Optativa		tinyint	NULL	Indicador de que la asignatura es optativa.
	Activo		bit	NOT NULL	Indicador de que la asignatura está activa.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
"Aula	Clave de aula	(PK)	numeric(8)	IDENTITY	Es un identificador único para cada aula.
	Nombre	(AK)	varchar(25)	NOT NULL	Es el nombre con el que se conoce a cada aula., ejemplos: salón Dr. Ignacio Burgoa Orihuela, salón Dr. Máximo C.
	Descripción		text	NULL	Es un texto explicativo de las características del aula.
	Capacidad		tinyint	NULL	Es el número máximo de alumnos que puede albergar un aula.
	Ubicación		text	NULL	Es el lugar físico donde se encuentra el aula.
	Activo		bit	NOT NULL	Indica si está disponible un aula para impartir clases.
Bachillerato	Clave de bachillerato	(PK)	numeric(4)	IDENTITY	Identificador único del bachillerato.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre que tiene el bachillerato.
	Activo		bit	NOT NULL	Indicador si un bachillerato está activo.
Bloque	Clave de bloque	(PK)	numeric(8)	IDENTITY	Identificador único de la clave de bloque.
	Clave de plan	(FK) (AK)	numeric(8)	NOT NULL	Identificador único de clave de plan.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre con el que se identifica a un bloque.
	Descripción		text	NULL	Texto descriptivo de un bloque.
	Créditos totales		smallint	NULL	Créditos totales que puede tener un bloque.
Calificación	Clave de calificación	(PK)	numeric(4)	IDENTITY	Identificador único de calificación.
	Representa-ción	(IE1)	char(2)	NOT NULL	Representación literal o numérica de la calificación.
	Valor numérico	(IE2)	numeric(4,2)	NULL	Valor numérico que corresponde a la representación literal.
	Aprobatoria		bit	NOT NULL	Indicador de que una calificación es aprobatoria.
	Activo		bit	NOT NULL	Indicador de que una calificación está activa.
Campus	Clave de campus	(PK)	numeric(4)	IDENTITY	Es un identificador único de cada campus.
	Clave de municipio	(FK)	numeric(8)	NULL	Es un identificador único del municipio.
	Nombre	(AK)	varchar(25)	NOT NULL	Es el nombre con el que se conoce al campus, ejemplo: Ciudad Universitaria, ENEP Aragón, etc.
	Nombre corto		numeric(3)	NOT NULL	Es un nombre corto común del campus.
	Calle y número		varchar(40)	NULL	Es la calle y número donde se encuentra el campus.
	Colonia		varchar(30)	NULL	Es la colonia donde se encuentra el campus.
	Código postal		numeric(5)	NULL	Es el código postal al que pertenece el campus.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Teléfono		numeric(12)	NULL	Es el teléfono del campus.
	Descripción		text	NULL	Es una descripción breve del campus.
	Activo		bit	NOT NULL	Indicador de que el campus está activo para poder impartir clases.
Campus carrera	Clave de campus	(PK) (FK)	numeric(4)	NOT NULL	Es un identificador único del campus.
	Clave de carrera	(PK) (FK)	numeric(4)	NOT NULL	Es un identificador único de la clave de carrera.
Carrera	Clave de carrera	(PK)	numeric(4)	IDENTITY	Identificador único de la clave de la carrera.
	Nombre	(AK)	varchar(36)	NOT NULL	Nombre oficial de la carrera.
	Nombre corto		numeric(2)	NOT NULL	Nombre común con el que se identifica la carrera.
	Descripción		text	NULL	Texto descriptivo de la carrera.
	Activo		bit	NOT NULL	Indicador de que la carrera está activa para impartirse.
Categoría	Clave de categoría	(PK)	numeric(4)	IDENTITY	Es el número único que identifica a cada categoría.
	Nombre	(AK)	varchar(25)	NOT NULL	Es el nombre que se le da a la categoría.
	Activo		bit	NOT NULL	Indicador de que una categoría está vigente para su aplicación.
DGAE-ABC de grupo	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Clave de asignatura		numeric(4)	NULL	Identificador único de la asignatura.
	Grupo		char(4)	NULL	Grupo a realizar movimiento.
	Cupo		numeric(4)	NULL	Cantidad de alumnos en el grupo.
	Profesor		numeric(1)	NULL	Clave de profesor asignado al grupo.
	Nombre del profesor		char(32)	NULL	Nombre del profesor asignado al grupo.
	RFC		char(13)	NULL	RFC del profesor.
	Movimiento		char(2)	NULL	Tipo de acción a realizar: alta, baja o cambio.
	Filler		char(3)	NULL	Campo solicitado por DGAE.
DGAE-ABC de Inscripción	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Clave de plantel		numeric(3)	NULL	Clave de plantel del alumno.
	Clave de la asignatura		numeric(4)	NULL	Clave de la asignatura a cursar.
	Clave de grupo de baja		char(4)	NULL	Clave del grupo origen.
	Clave de grupo de alta		char(4)	NULL	Clave del grupo destino.
	Filler		char(1)	NULL	Campo solicitado por DGAE.
DGAE - actas emitidas	Folio		numeric(7)	NULL	Identificador único del acta emitida.
	Plantel		numeric(3)	NULL	Plantel del acta.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Asignatura		numeric(4)	NULL	Asignatura que cubre el acta.
	Tipo de examen		numeric(1)	NULL	Ordinario o extraordinario.
	Semestre escolar		numeric(1)	NULL	Semestre en que se cursó.
	Año escolar		numeric(2)	NULL	Año escolar en que se cursó.
	Actas del grupo		numeric(2)	NULL	Número de actas correspondientes al grupo.
	Alumnos del grupo		numeric(4)	NULL	Alumnos que contiene el acta.
	Fecha de emisión		numeric(6)	NULL	Fecha de emisión del acta.
	Grupo		char(4)	NULL	Grupo del acta.
	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Calificación		numeric(1)	NULL	Calificación obtenida.
	Tipo de acta		char(1)	NULL	Tipo de acta emitida.
	Filler		char(4)	NULL	Campo solicitado por DGAE.
DGAE Baja de inscripción	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Clave de carrera		numeric(2)	NULL	Identificador de la carrera.
	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Filler		char(2)	NULL	Campo solicitado por DGAE.
DGAE Cambio de carrera interna	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Clave de carrera origen		numeric(2)	NULL	Identificador de la carrera origen.
	Clave de carrera destino		numeric(2)	NULL	Identificador de la carrera destino.
DGAE Cambio de unidad académica	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Clave del plantel origen		numeric(3)	NULL	Identificador del plantel origen.
	Clave de carrera origen		numeric(2)	NULL	Identificador de la carrera origen.
	Clave del plantel destino		numeric(3)	NULL	Identificador del plantel destino.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Clave de carrera destino		numeric(2)	NULL	Identificador de la carrera destino.
DGAE Directorio de alumnos	Nombre del alumno		char(32)	NULL	Nombre del alumno.
	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Clave de carrera o turno		numeric(2)	NULL	Identificador de la carrera o turno.
	Año de primer ingreso		numeric(2)	NULL	Año en que ingresó a la UNAM.
	Nacionalidad		numeric(1)	NULL	Indicador de si es mexicano o no.
	Causa de ingreso		numeric(2)	NULL	Identificador de la causa de ingreso.
	Causa de exalumno		numeric(2)	NULL	Identificador de la causa de exalumno.
	Sexo		char(1)	NULL	Sexo del alumno.
	Fecha de nacimiento		numeric(6)	NULL	Fecha en que nació el alumno.
	Fecha de movimiento		numeric(6)	NULL	Fecha en que se hizo alguna modificación a su registro.
	Marca de inscrito		numeric(1)	NULL	Señal de que se encuentra actualmente inscrito.
DGAE Historias Académicas	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Clave de asignatura		numeric(4)	NULL	Identificador de la asignatura.
	Año semestre		numeric(3)	NULL	Año semestre en que se cursó la asignatura.
	Calificación		char(2)	NULL	Calificación obtenida.
	Grupo		char(4)	NULL	Grupo en el que se cursó la asignatura.
	Folio del acta		numeric(7)	NULL	Identificación del acta.
	Tipo de examen		char(2)	NULL	Tipo de examen sustentado.
DGAE Inscripción entrada	Número de cuenta		tinyint	NULL	Número de cuenta del alumno.
	Clave de plantel		tinyint	NULL	Identificador del plantel.
	Clave de asignatura		tinyint	NULL	Identificador de la asignatura.
	Clave de grupo		tinyint	NULL	Identificador del grupo.
	Filler		char(5)	NULL	Campo solicitado por DGAE.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
DGAE Inscripción Salida	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Clave de asignatura		numeric(4)	NULL	Identificador de la asignatura.
	Clave de grupo		char(4)	NULL	Identificador del grupo.
	Filler		char(5)	NULL	Campo solicitado por DGAE.
DGAE modificación a historia académica	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Clave de asignatura		numeric(4)	NULL	Identificador de la asignatura.
	Periodo		numeric(3)	NULL	Periodo en que se cursó la asignatura.
	Tipo de examen		char(1)	NULL	Ordinario o extraordinario.
	Calificación		char(2)	NULL	Calificación obtenida.
	Folio		numeric(7)	NULL	Identificador del acta.
	Grupo		char(4)	NULL	Identificador del grupo.
	Tipo de movimiento		char(2)	NULL	Movimiento a realizar: alta, baja o cambio.
DGAE modificación al registro del alumno	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	Clave de carrera		numeric(2)	NULL	Identificador de la carrera.
	Clave de causa de baja		char(2)	NULL	Clave de la causa de baja.
DGAE relación de asignaturas	Nombre de asignatura		char(28)	NULL	Nombre con el que se conoce a la asignatura.
	Clave de plantel		numeric(3)	NULL	Identificador del plantel.
	Clave de la asignatura		numeric(4)	NULL	Identificador de la asignatura.
	Créditos de la asignatura		numeric(2)	NULL	Créditos que otorga la asignatura.
	Semestre de la asignatura		numeric(2)	NULL	Semestre en que debe cursarse la asignatura.
	Nivel		char(1)	NULL	Nivel académico en que debe cursarse.
	Filler		char(2)	NULL	Campo solicitado por DGAE.
DGAE Relación de carreras	Clave de plantel		numeric(3)	NULL	Identificador del plantel.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Clave de la carrera		numeric(2)	NULL	Identificador de la carrera.
	Nombre de la carrera		char(36)	NULL	Nombre con el que se conoce a la carrera.
	Nivel		char(1)	NULL	Nivel académico en que debe cursarse.
	Plan		numeric(1)	NULL	Clave de plan de estudios que le corresponde.
	Duración de la carrera		numeric(2)	NULL	Tiempo en que debe cursarse la carrera.
	Créditos obligatorios		numeric(3)	NULL	Número de créditos obligatorios de la carrera.
	Créditos optativos		numeric(3)	NULL	Número de créditos optativos de la carrera.
	Filler		char(5)	NULL	Campo solicitado por DGAE.
DGAE resumen de historias académicas	Número de cuenta		numeric(8)	NULL	Número de cuenta del alumno.
	MB		tinyint	NULL	Número de calificaciones con MB.
	B		tinyint	NULL	Número de calificaciones con B.
	S		tinyint	NULL	Número de calificaciones con S.
	6		tinyint	NULL	Número de calificaciones con seis.
	7		tinyint	NULL	Número de calificaciones con siete.
	8		tinyint	NULL	Número de calificaciones con ocho.
	9		tinyint	NULL	Número de calificaciones con nueve.
	10		tinyint	NULL	Número de calificaciones con diez.
	NA		tinyint	NULL	Número de calificaciones con NA.
	NP		tinyint	NULL	Número de calificaciones con NP.
	Revalidadas		tinyint	NULL	Número de asignaturas revalidadas.
	Acreditadas		tinyint	NULL	Número de asignaturas acreditadas.
	Aprobadas en ordinario		tinyint	NULL	Número de asignaturas aprobadas en ordinario.
	Aprobadas en extraordinario		tinyint	NULL	Número de asignaturas aprobadas en extraordinario.
	Reprobadas en ordinario		tinyint	NULL	Número de asignaturas reprobadas en ordinario.
	Reprobadas en extraordinario		tinyint	NULL	Número de asignaturas reprobadas en extraordinario.
	Créditos obligatorios acumulados		numeric(3)	NULL	Cantidad de créditos obligatorios acumulados.
	Créditos optativos acumulados		numeric(3)	NULL	Cantidad de créditos optativos acumulados.
	Periodo inicial		numeric(3)	NULL	Periodo inicial que comprende el resumen.
	Periodo último		numeric(3)	NULL	Periodo final que comprende el resumen.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Covalidadas		tinyint	NULL	Cantidad de asignaturas covalidadas.
Estado	Clave de estado	(PK)	numeric(4)	IDENTITY	Es un identificador único del estado.
	Nombre	(AK1)	varchar(25)	NOT NULL	Es el nombre oficial del estado, ejemplo: Veracruz, Colima, Nuevo León, etc.
	Abreviación	(AK2)	char(4)	NOT NULL	Abreviación del estado.
Estado civil	Clave de estado civil	(PK)	numeric(4)	IDENTITY	Es un identificador único del estado civil.
	Nombre	(AK1)	varchar(25)	NOT NULL	Es el nombre común del estado civil: soltero, casado, etc.
Estado de alumno	Clave de estado de alumno	(PK)	numeric(4)	IDENTITY	Identificador único del estado de alumno.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre del estado.
	Activo		bit	NOT NULL	Indicador de que el estado está activo.
Estado de grupo	Clave de estado de grupo	(PK)	numeric(4)	IDENTITY	Identificador único del estado de grupo.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre del estado de grupo.
	Activo		bit	NOT NULL	Indicador de que el estado está activo.
Estado de inscripción	Clave de estado de inscripción	(PK)	numeric(4)	IDENTITY	Identificador único del estado de inscripción.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre del estado.
	Activo		bit	NOT NULL	Indicador de que el estado está activo.
Grado académico	Clave de grado académico	(PK)	numeric(4)	IDENTITY	Identificador único del grado académico.
	Nombre	(AK1)	varchar(25)	NOT NULL	Nombre común del grado académico.
	Descripción		text	NULL	Texto descriptivo del grado académico.
Grupo	Clave de grupo	(PK)	numeric(16)	IDENTITY	Identificador único del grupo.
	Clave de asignatura plan	(FK)	numeric(8)	NOT NULL	Identificador único de la asignatura plan.
	Clave de periodo	(FK)	numeric(8)	NOT NULL	Identificador único de la clave de periodo.
	Clave de modalidad inscripción	(FK)	numeric(4)	NOT NULL	Identificador único de la clave de la modalidad de inscripción.
	Nombre corto		char(4)	NOT NULL	Nombre que DGAE le asigna al grupo.
	Clave de estado de grupo	(FK)	numeric(4)	NULL	Identificador único de la clave del estado grupo.
	Clave de turno	(FK)	numeric(4)	NULL	Identificador único del turno.
	Cupo		tinyint	NULL	Cupo máximo del grupo.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Lugares ocupados		tinyint	NULL	Lugares ocupados que tiene un grupo durante cierto proceso de inscripción.
Horario	Clave de grupo	(PK) (FK)	numeric(16)	NOT NULL	Identificador único de la clave de grupo.
	Clave de aula	(PK) (FK)	numeric(8)	NOT NULL	Es un identificador para cada aula.
	Lunes		bit	NOT NULL	Indica si la materia se imparte en lunes.
	Martes		bit	NOT NULL	Indica si la materia se imparte en martes.
	Miércoles		bit	NOT NULL	Indica si la materia se imparte en miércoles.
	Jueves		bit	NOT NULL	Indica si la materia se imparte en jueves.
	Viernes		bit	NOT NULL	Indica si la materia se imparte en viernes.
	Sábado		bit	NOT NULL	Indica si la materia se imparte en sábado.
	Hora de inicio		smalldatetime	NULL	Hora de inicio de la cátedra.
	Hora de término		smalldatetime	NULL	Hora de término de la cátedra.
Horario de atención	Clave de horario de atención	(PK)	numeric(8)	IDENTITY	Identificador único de la clave del horario de atención.
	Clave de modalidad de inscripción	(FK)	numeric(4)	NOT NULL	Identificador único de la clave de modalidad de inscripción.
	Clave de periodo	(FK)	numeric(8)	NOT NULL	Identificador único de la clave de periodo.
	Clave de turno	(FK)	numeric(4)	NULL	Identificador único del turno.
	Hora de inicio		smalldatetime	NULL	Hora de inicio de la inscripción.
	Hora de termino		smalldatetime	NULL	Hora de término de la inscripción.
	Tiempo máximo de inscripción		smalldatetime	NULL	Tiempo máximo posible que se puede permanecer haciendo movimientos de inscripción.
	Número máximo de movimientos		tinyint	NULL	Límite para hacer movimientos de inscripción.
	Número máximo de atenciones		tinyint	NULL	Número máximo de atenciones permitidas durante la inscripción.
	Altas		bit	NOT NULL	Indicador booleano que indica que se puede hacer un alta de asignatura.
	Bajas		bit	NOT NULL	Indicador booleano que indica que se puede hacer una baja de asignatura.
	Cambios		bit	NOT NULL	Indicador booleano que indica que se puede hacer un cambio de asignatura.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Consultas		bit	NOT NULL	Indicador booleano que indica que se puede hacer una consulta de asignatura.
	Activo		bit	NOT NULL	Indicador booleano que indica si un horario de atención está activo.
Inscripción	Clave de plan	(PK) (FK)	numeric(8)	NOT NULL	Identificador único de clave de plan.
	Número de cuenta	(PK) (FK)	numeric(9)	NOT NULL	Identificador único oficial del alumno.
	Clave de grupo	(PK) (FK)	numeric(16)	NOT NULL	Identificador único del grupo.
	Clave de estado de inscripción	(FK)	numeric(4)	NULL	Identificador único de la clave de estado de inscripción.
	Clave de calificación	(FK)	numeric(4)	NULL	Identificador único de calificación.
	Fecha de inscripción		smalldatetime	NULL	Fecha en que se realizó la inscripción a una asignatura.
Modalidad	Clave de modalidad de inscripción	(PK)	numeric(4)	IDENTITY	Identificador único de la clave de modalidad de inscripción.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre de la modalidad.
	Nombre corto		char(2)	NOT NULL	Nombre corto abreviado de la modalidad.
	Número máximo de inscripciones		tinyint	NULL	Número máximo de inscripciones permitidas en la modalidad.
	Calificación máxima		numeric(4,2)	NULL	Calificación máxima permitida en la modalidad.
	Calificación mínima		numeric(4,2)	NULL	Calificación mínima permitida en la modalidad.
	Activo		bit	NOT NULL	Señal de que la modalidad está activa.
Modalidad tipo	Clave de tipo de asignatura	(PK) (FK)	numeric(4)	NOT NULL	Identificador único de la clave de tipo de asignatura.
	Clave de modalidad de inscripción	(PK) (FK)	numeric(4)	NOT NULL	Identificador único de la clave de la modalidad de inscripción.
	Número máximo de inscripciones		tinyint	NULL	Número máximo de inscripciones permitidas en la modalidad tipo.
	Calificación máxima		numeric(4,2)	NULL	Calificación máxima permitida en la modalidad tipo.
	Calificación mínima		numeric(4,2)	NULL	Calificación mínima permitida en la modalidad tipo.
Municipio	Clave de municipio	(PK)	numeric(8)	IDENTITY	Es un identificador único del municipio.
	Nombre	(IE1)	varchar(25)	NOT NULL	Es el nombre oficial del municipio, ejemplo: Ecatepec de Morelos, Minatitlán, Tlanepantla de Baz, etc.
	Clave de estado	(FK)	numeric(4)	NOT NULL	Es un identificador único del estado.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
Nivel	Clave de nivel	(PK)	numeric(4)	IDENTITY	Identificador único del nivel.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre con el que se identifica al nivel: licenciatura, maestría, etc.
	Descripción		text	NULL	Texto descriptivo del nivel académico.
	Activo		bit	NOT NULL	Indicador de que un nivel está activo para utilizarse.
Observación	Clave de observación	(PK)	numeric(16)	IDENTITY	Identificador único de la observación.
	Número de cuenta	(FK)	numeric(9)	NOT NULL	Número de cuenta del alumno.
	Titulo		varchar(50)	NOT NULL	Titulo de la observación realizada.
	Autor		varchar(50)	NULL	Persona que hace la observación al alumno.
	Fecha		smalldatetime	NULL	Fecha en que se realiza la observación.
	Clave de categoría	(FK)	numeric(4)	NOT NULL	Es el número que identifica a cada categoría.
	Descripción		varchar(255)	NULL	Texto descriptivo de la observación hecha.
Observacio-nes profesor	Clave de observación	(PK)	numeric(4)	IDENTITY	Identificador único de la observación.
	Clave de profesor	(FK)	numeric(8)	NOT NULL	Identificador único del profesor.
	Descripción		varchar(250)	NULL	Texto descriptivo de la observación realizada.
	Fecha		smalldatetime	NULL	Fecha de realización de la observación.
	Autor		varchar(50)	NULL	Persona que realizó la observación.
	Titulo		varchar(50)	NOT NULL	Texto con el que se identifica una observación.
Parámetro	Clave de parámetro	(PK)	numeric(4)	NOT NULL	Identificador único del parámetro.
	Nombre		varchar(25)	NOT NULL	Nombre del parámetro.
	Valor booleano		bit	NOT NULL	Valor booleano del parámetro.
	Valor numérico		int	NULL	Valor numérico del parámetro.
	Valor cadena		varchar(255)	NULL	Valor como cadena de caracteres del parámetro.
	Activo		bit	NOT NULL	Indicador de que un nivel parámetro está activo para utilizarse.
Periodo	Clave de periodo	(PK)	numeric(8)	IDENTITY	Identificador único de la clave de periodo.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre con el que se conoce al periodo.
	Nombre corto		numeric(5)	NOT NULL	Nombre que asigna DGAE al periodo.
	Fecha de inicio		smalldatetime	NULL	Fecha de inicio del semestre.
	Fecha de término		smalldatetime	NULL	Fecha de término del semestre.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Activo		bit	NOT NULL	Indicador de que un periodo está activo.
Plan de estudios	Clave de plan	(PK)	numeric(8)	IDENTITY	Identificador único de la clave de plan.
	Clave de nivel	(FK)	numeric(4)	NOT NULL	Identificador único del nivel.
	Clave de carrera	(FK) (AK1)	numeric(4)	NOT NULL	Identificador único de la clave de la carrera.
	Nombre	(AK1)	varchar(25)	NOT NULL	Nombre del plan de estudios.
	Descripción		text	NULL	Texto descriptivo del plan de estudios.
	Periodo inicial		smalldatetime	NULL	Periodo en el que inicia el plan de estudios.
	Periodo final		smalldatetime	NULL	Periodo en el que termina el plan de estudios.
	Créditos obligatorios		smallint	NULL	Créditos obligatorios que contiene el plan.
	Créditos optativos		smallint	NULL	Créditos optativos que contiene el plan.
	Asignaturas obligatorias		tinyint	NULL	Asignaturas obligatorias que deben cursarse en el plan.
	Periodos		tinyint	NULL	Periodos en los que debe cursarse el plan.
	Terminado		bit	NOT NULL	Señal de que el plan de estudios está cerrado.
	Activo		bit	NOT NULL	Indicador de que el plan está activo.
Profesor	Clave de profesor	(PK)	numeric(8)	NOT NULL	Identificador único del profesor.
	RFC	(AK)	char(13)	NOT NULL	Registro federal de causantes del profesor.
	Clave de grado académico	(FK)	numeric(4)	NULL	Identificador único del grado académico.
	Clave de estado civil	(FK)	numeric(4)	NULL	Es un identificador único del estado civil.
	Clave de tipo de profesor	(FK)	numeric(4)	NULL	Identificador único del tipo de profesor.
	Trabajo municipio	(FK)	numeric(8)	NULL	Es un identificador único del municipio.
	Nombre		varchar(25)	NOT NULL	Nombre del profesor.
	Apellido paterno		varchar(20)	NULL	Apellido paterno del profesor.
	Apellido materno		varchar(20)	NULL	Apellido materno del profesor.
	Nombre corto		varchar(32)	NOT NULL	Apellido paterno, materno y nombre del profesor.
	Extranjero		bit	NOT NULL	Indicador de que el profesor es extranjero.
	Sexo		bit	NOT NULL	Sexo del profesor.
	Fecha de nacimiento		smalldatetime	NULL	Fecha de nacimiento del profesor.
	Cédula profesional		numeric(9)	NULL	Cédula profesional del profesor.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Observaciones		text	NULL	Observaciones diversas que se le pueden hacer al profesor.
	Casa calle y número		varchar(40)	NULL	Calle y número del domicilio del profesor.
	Casa colonia		varchar(30)	NULL	Colonia del domicilio del profesor.
	Casa código postal		numeric(5)	NULL	Código Postal del domicilio del profesor.
	Casa teléfono		numeric(12)	NULL	Teléfono del domicilio del profesor.
	Trabajo calle y número		varchar(40)	NULL	Calle y número del domicilio de trabajo del profesor.
	Trabajo colonia		varchar(30)	NULL	Colonia del domicilio de trabajo del profesor.
	Trabajo código postal		numeric(5)	NULL	Código postal del domicilio de trabajo del profesor.
	Trabajo teléfono		numeric(12)	NULL	Teléfono del domicilio de trabajo del profesor.
	Correo electrónico		varchar(30)	NULL	Correo electrónico del profesor.
	Fecha de ingreso		smalldatetime	NULL	Fecha en que ingresó como profesor.
	Casa municipio	(FK)	numeric(8)	NULL	Es un identificador único del municipio.
	Activo		bit	NOT NULL	Indicador de que un profesor está activo para impartir clases.
Profesor grupo	Clave de grupo	(PK) (FK)	numeric(16)	NOT NULL	Identificador único del grupo.
	Clave de profesor	(PK) (FK)	numeric(8)	NOT NULL	Identificador único del profesor.
	Fecha de inicio		smalldatetime	NULL	Fecha de inicio de asignación al grupo.
	Fecha de término		smalldatetime	NULL	Fecha de término de asignación al grupo.
	Número de profesor		tinyint	NULL	Clave de identificación del profesor al grupo.
Seriación Asignatura	Antecedente	(PK) (FK)	numeric(8)	NOT NULL	Identificador único de la asignatura plan
	Consecuente	(PK) (FK)	numeric(8)	NOT NULL	Identificador único de la asignatura plan.
	Simultánea		bit	NOT NULL	Indicador de que las asignaturas se cursan simultáneamente.
Seriación bloque	Antecedente	(PK) (FK)	numeric(8)	NOT NULL	Identificador único de la clave de bloque.
	Consecuente	(PK) (FK)	numeric(8)	NOT NULL	Identificador único de la clave de bloque.
	Créditos de avance mínimo		smallint	NULL	Créditos de avance mínimo que se deben cumplir de un bloque.
Tipo de asignatura	Clave de tipo de asignatura	(PK)	numeric(4)	IDENTITY	Identificador único de la clave de tipo de asignatura.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre con el que se conoce al tipo de asignatura.

Entidad	Atributo	Llave	Tipo de dato	Nulidad	Descripción del campo
	Activo		bit	NOT NULL	Indicador de si un tipo de asignatura está activo.
Tipo de ingreso	Clave de tipo de ingreso	(PK)	numeric(4)	IDENTITY	Identificador único de la forma de ingreso.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre de la forma de ingreso.
	Activo		bit	NOT NULL	Indicador de si la forma de ingreso está activa.
Tipo de profesor	Clave de tipo de profesor	(PK)	numeric(4)	IDENTITY	Identificador único del tipo de profesor.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre con el que se conoce al tipo de profesor.
	Descripción		text	NULL	Texto descriptivo del tipo de profesor.
	Activo		bit	NOT NULL	Indicador de que un tipo de profesor está activo.
Turno	Clave de turno	(PK)	numeric(4)	IDENTITY	Identificador único del turno.
	Nombre	(AK)	varchar(25)	NOT NULL	Nombre común que se le asigna a un turno, ej. matutino, vespertino.
	Hora de inicio		smalldatetime	NULL	Hora de inicio de clases del turno.
	Hora de término		smalldatetime	NULL	Hora de término de clases del turno.
	Activo		bit	NOT NULL	Indicador de que un turno está activo.

5.1.2 Implementación de la base de datos en Sybase.

La versión de Sybase utilizada es la 11, y para la generación de la base de datos del sistema de inscripciones denominada "SI", se utilizó Sybase SQL Server Manager. Esta herramienta, como su nombre lo indica, es un administrador de bases de datos Sybase para el servidor SQL. Los elementos que administra son: los servidores SQL y los recursos físicos del SQL, las bases de datos y sus objetos, el acceso a los servidores y los nuevos objetos de bases de datos y del servidor.

Los pasos necesarios para dar de alta la base de datos son:

1. - Conexión al servidor SQL.
   - Elegir el icono de conexión al servidor SQL.
   - Escoger un servidor.
   - Introducir el nombre de usuario y password y dar aceptar.

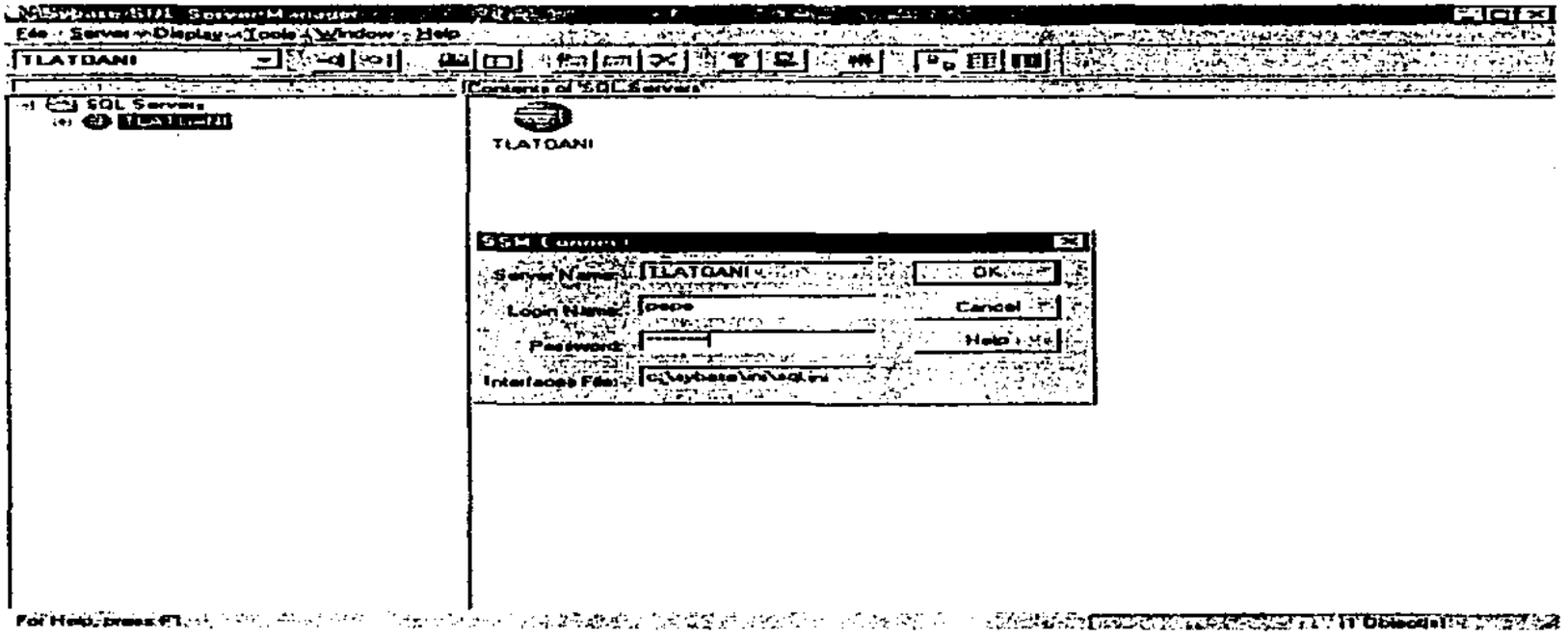


Figura 5.1.2.1 Conexión al servidor SQL.

2. – Creación de un dispositivo de la base de datos.

- Elegir el ícono del servidor SQL.
- Elegir Server/Create/Database Device.
- Asignarle un nombre al dispositivo.
- Dar el nombre completo del dispositivo físico (Partición de disco duro o archivo de S.O.).
- Introducir el número del controlador del disco.
- Asignar el tamaño del dispositivo en megabytes, en caso de que sea una partición de disco duro, introducir el tamaño de ésta. El tamaño de la base de datos fue el que se indicó en el diseño, que asciende a 308 megabytes, incluyendo el log de transacciones.
- El administrador de SQL indica el número secuencial disponible para el dispositivo virtual.
- Cuando es el caso, se puede usar el Mirror Name para tener un espejo del dispositivo.
- Finalmente, se indica Create.

3. - Creación de la base de datos.

- Elegir Server/Create/Database.
- Introducir el nombre de la base de datos: "SI".
- Dar el nombre del dueño de la base de datos: "JBD".
- Reservar el espacio en los dispositivos para almacenar la base de datos.
- Seleccionar el nombre del dispositivo.
- Asignar el espacio que se requiere reservar.
- Elegir si se desea el log de transacciones en el mismo dispositivo que los datos o por separado. Por seguridad, en nuestro caso se eligió por separado. De esta manera, en caso de algún daño físico o lógico al dispositivo de los datos, se puede monitorear en el log el estatus de la base de datos hasta ese momento.
- Escoger Add para dar de alta la reservación del dispositivo.
- Seleccionar de una lista el tamaño del buffer del log. Este determina la cantidad de entradas y salidas que puede ejecutar el servidor SQL para el buffer de la memoria, en la memoria cache designada para la entrada/salida del log de transacciones.
- Dar Create.

Las estructuras creadas para la administración y funcionamiento de la base de datos son:

- Users. Contiene a los usuarios de la base de datos.
- Groups. Contiene las diferentes categorías de usuarios que existen.
- System tables. Aquí se pueden ver las tablas que usa Sybase para operar.
- User tables. Aquí se visualizan las tablas definidas por el usuario.
- Views. Son las vistas que el usuario definió.
- Indexes. Son los índices definidos por el usuario.
- Triggers. Contiene los triggers o disparadores que se especificaron para integridad de la información.
- Procedures. Permite visualizar los procedimientos almacenados definidos en el sistema.
- Rules. Contiene las reglas respecto a los valores de los datos.
- Defaults. Muestra los valores de default definidos para los datos.

- User Datatypes. Contiene los tipos de datos que el usuario definió.
- Segments. Contiene los diferentes segmentos de la base de datos, en caso de que sean más de uno.

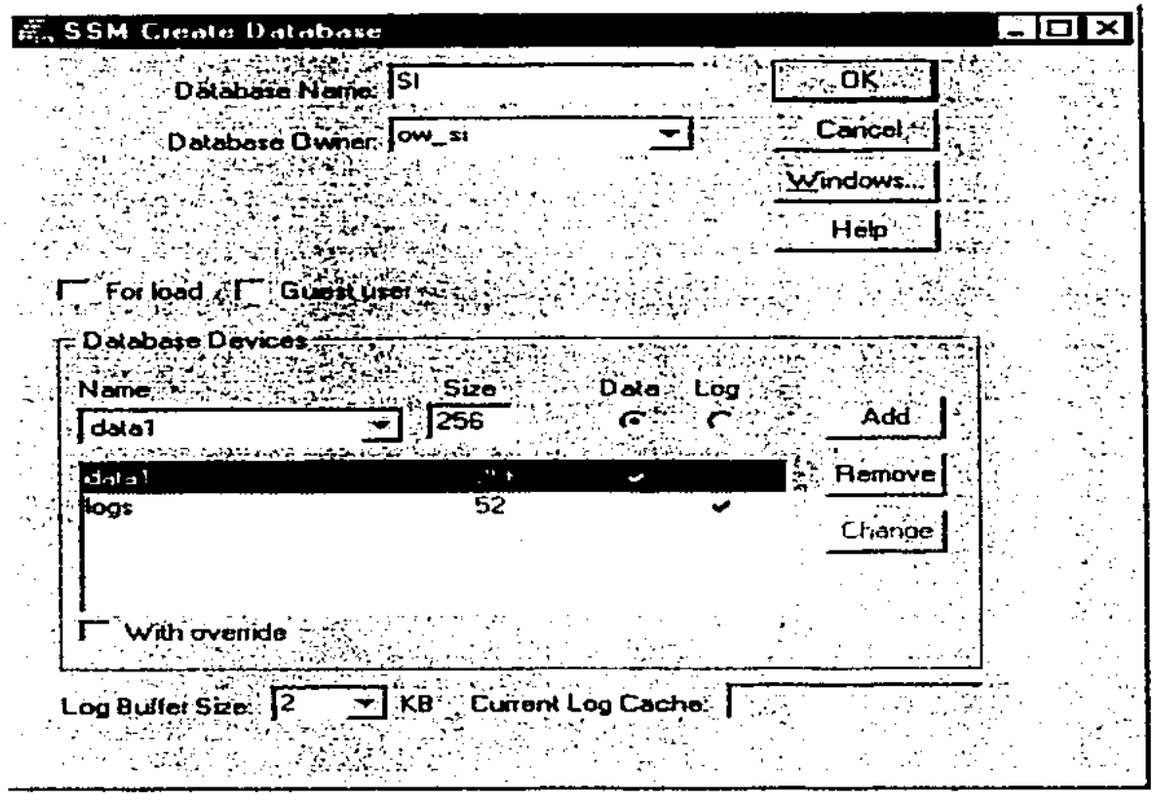


Figura 5.1.2.2 Creación de la base de datos.

En el caso de la base de datos del sistema de inscripciones contiene todos los elementos anteriores, excepto el último, por estar definido en un solo dispositivo. Vale la pena resaltar que para Sybase el término usuario es el que diseñó e implementó la base de datos.

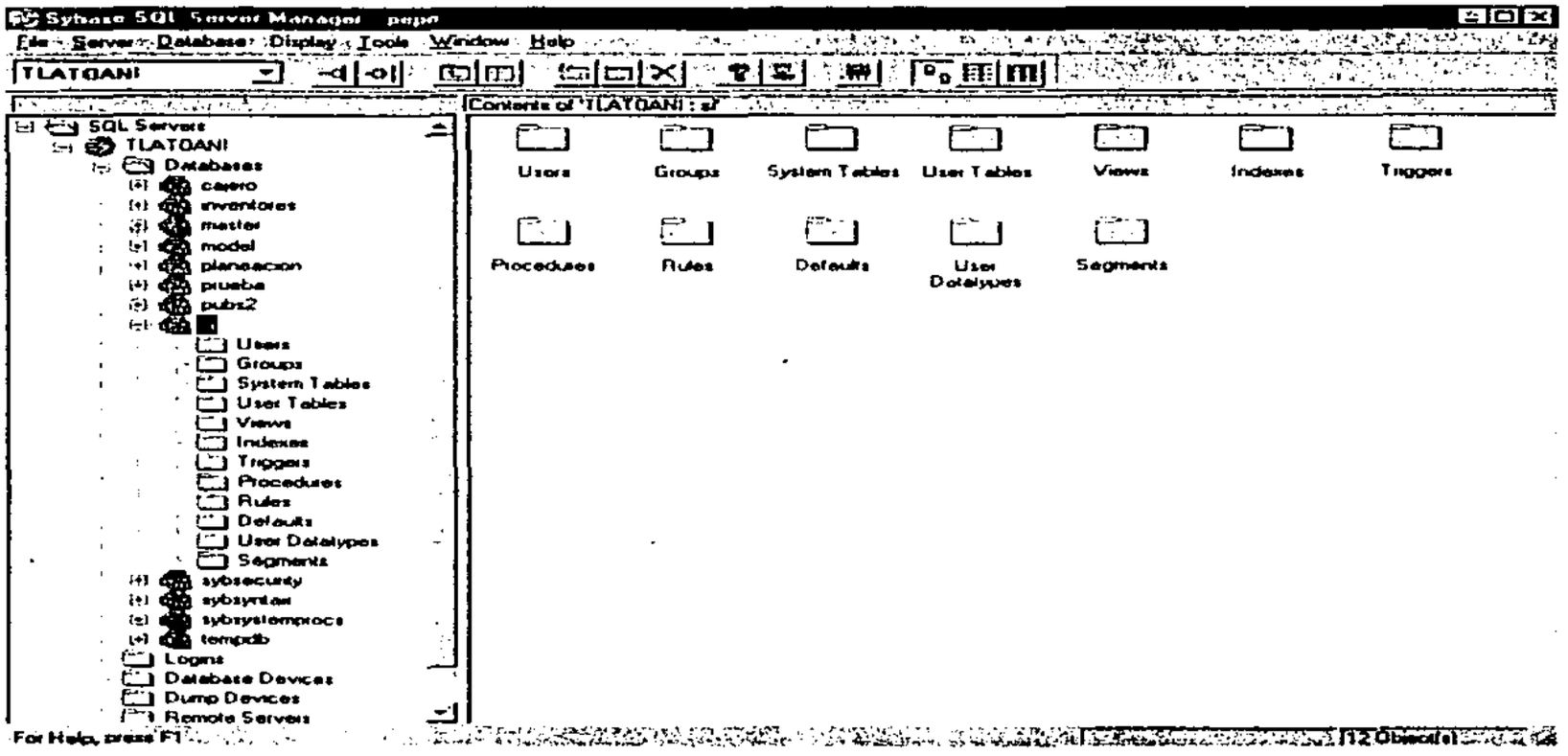


Figura 5.1.2.3 Esquema gráfico del contenido de la base de datos.

5.1.3 Proceso de creación de la base de datos en ERwin.

Como se mencionó en el Capítulo 4, el análisis y diseño de la base de datos se realizó con el CASE ERwin, permitiendo la creación del modelo físico a partir del lógico. Los componentes que se pueden crear sin teclear instrucciones SQL son: tablas, vistas, índices, triggers, stored procedures, rules, defaults y tipos de datos. Todos éstos fueron utilizados en el presente sistema.

Otra característica importante de ERwin que no fue necesario utilizar en este proyecto, es la creación del modelo de datos a partir de una base de datos existente, proceso que se conoce como ingeniería en reversa.

En este capítulo se describe el proceso de creación del modelo lógico y físico en ERwin.

1. - Definición de entidades y atributos.

Primeramente se definen las entidades, especificando el nombre de la entidad, los atributos que contiene y clasificándolos en los que forman parte de la llave y los que no son llave.

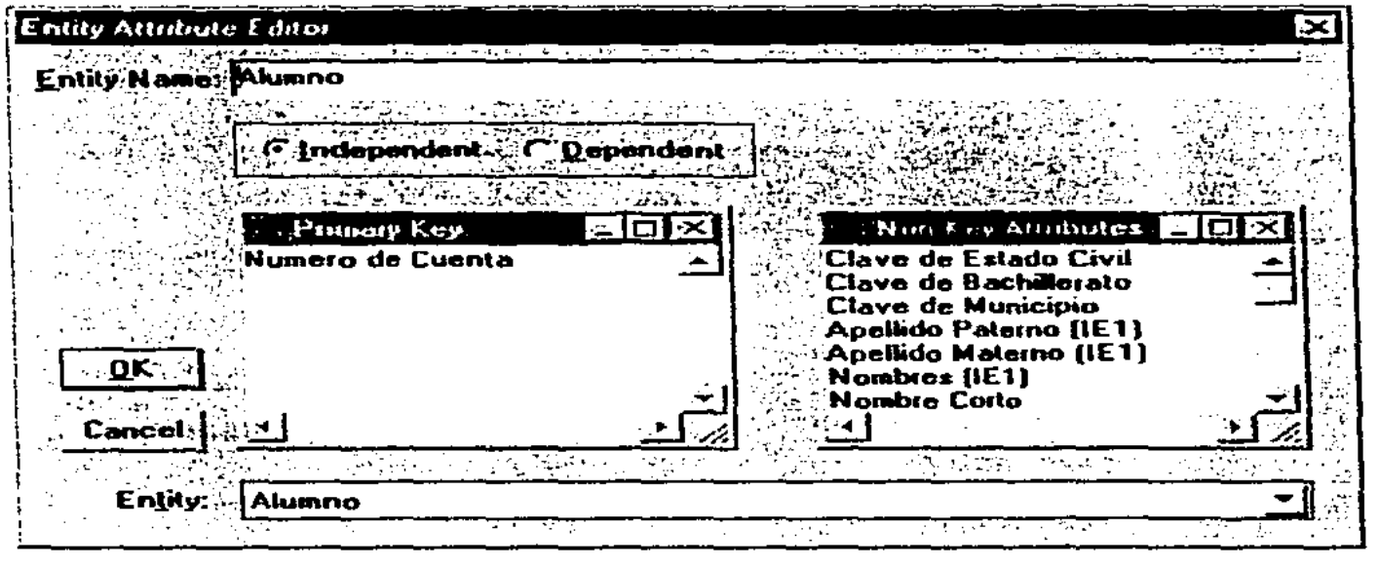


Figura 5.1.3.1 Definición de la tabla de alumno y sus atributos.

2. - Relación entre entidades.

Una vez que se tienen definidas todas las entidades se especifica la relación que existe entre ellas, pudiendo ser: identifica, no identifica y muchos a muchos. En el primer caso, la llave primaria de la primer entidad forma parte de los atributos de la segunda, pudiendo ser llave primaria.

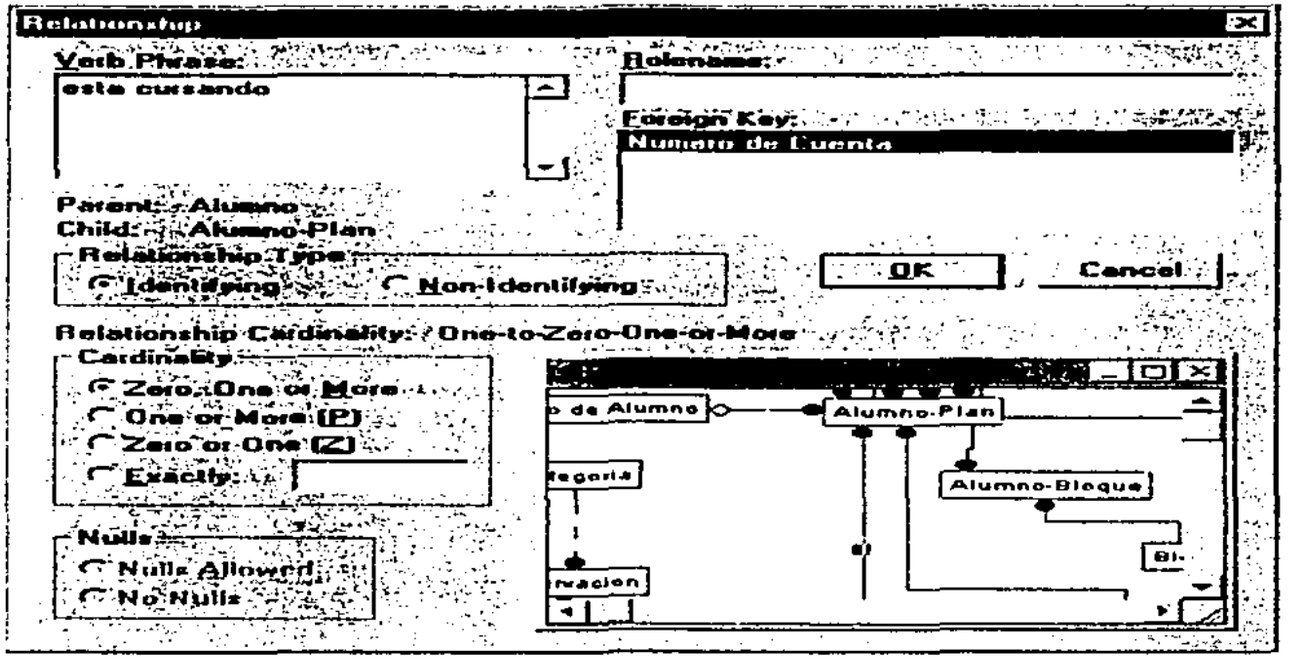


Figura 5.1.3.2 Definición de la relación entre la tabla de alumno y alumno_plan.

3. - Defaults.

La ventaja de definir defaults, al igual que las constraints y datatypes, es que se especifican una sola vez y pueden usarse en cualquier entidad. El default es un valor que se indica poseerá un dato en caso de que no se proporcione ninguno por el sistema o como resultado de algún proceso.

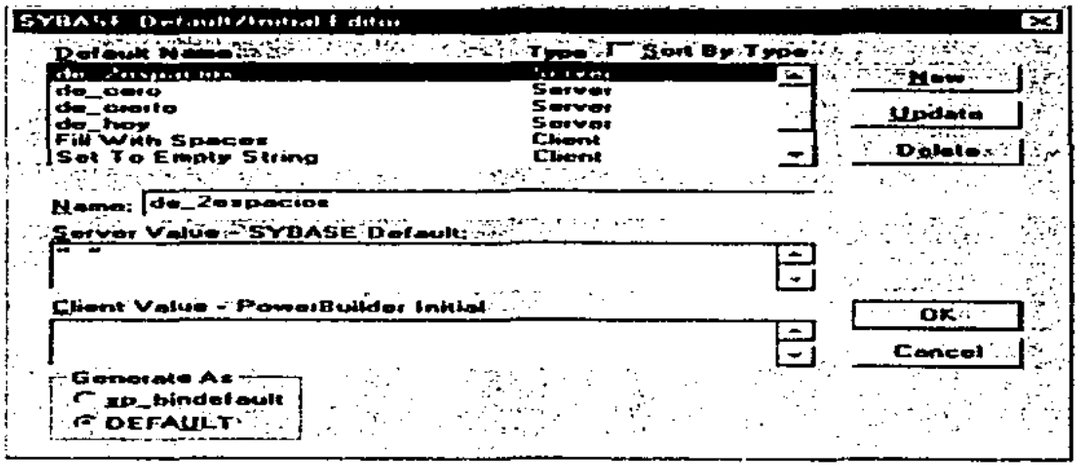


Figura 5.1.3.3 Definición de valores por default.

En nuestro sistema los defaults que existen son:

- Dos espacios. Pone dos espacios a un campo.
- Cierto. Asigna un "true" a un campo.
- Cero. Inserta un cero.

4. - Reglas (constraints).

En esta parte se definen los rangos válidos que puede tomar un campo en una inserción o un cambio.

5. - Tipos de datos (datatypes).

La ventaja de definir tipos de datos, es que es muy frecuente que se repitan a lo largo del modelo, y por lo tanto ahorra tiempo y reduce la posibilidad de error al crearlos una vez y utilizarlos durante la definición del esquema de la base de datos.

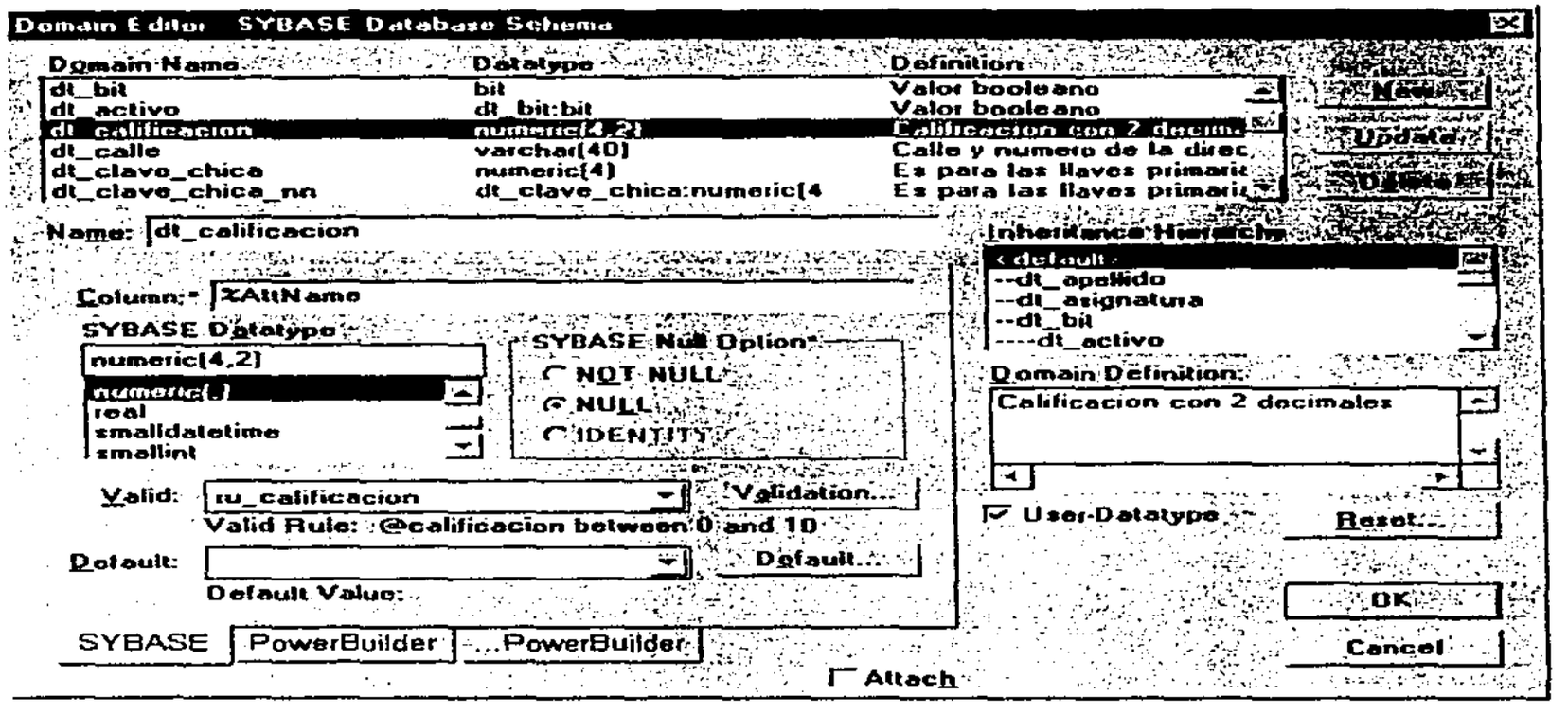


Figura 5.1.3.4 Definición de tipos de datos.

6. - Esquema de la base de datos.

En esta parte se definen los atributos, pudiendo escoger un tipo de dato previamente creado, o elegir uno diferente. Asi mismo, cuando es el caso se especifican las reglas y los defaults de un campo.

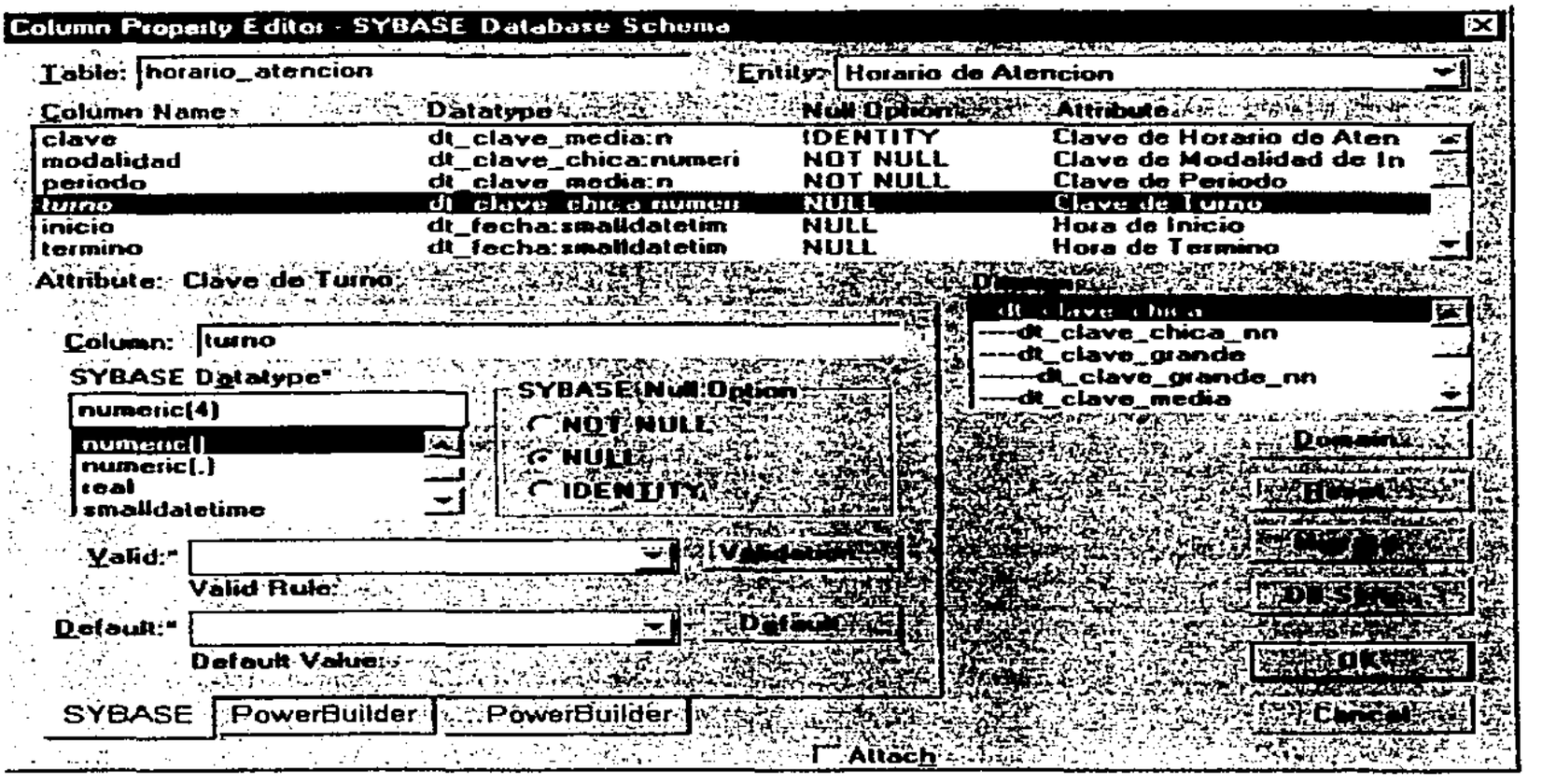


Figura 5.1.3.5 Definición de tipos de datos.

7. - Integridad referencial.

Sobre cada una de las relaciones definidas, se especifica la acción del trigger al momento de que ocurre una acción de borrado, inserción o actualización, tanto para el padre como para la hija. Para el caso de la figura, la descripción la encontramos en la siguiente tabla:

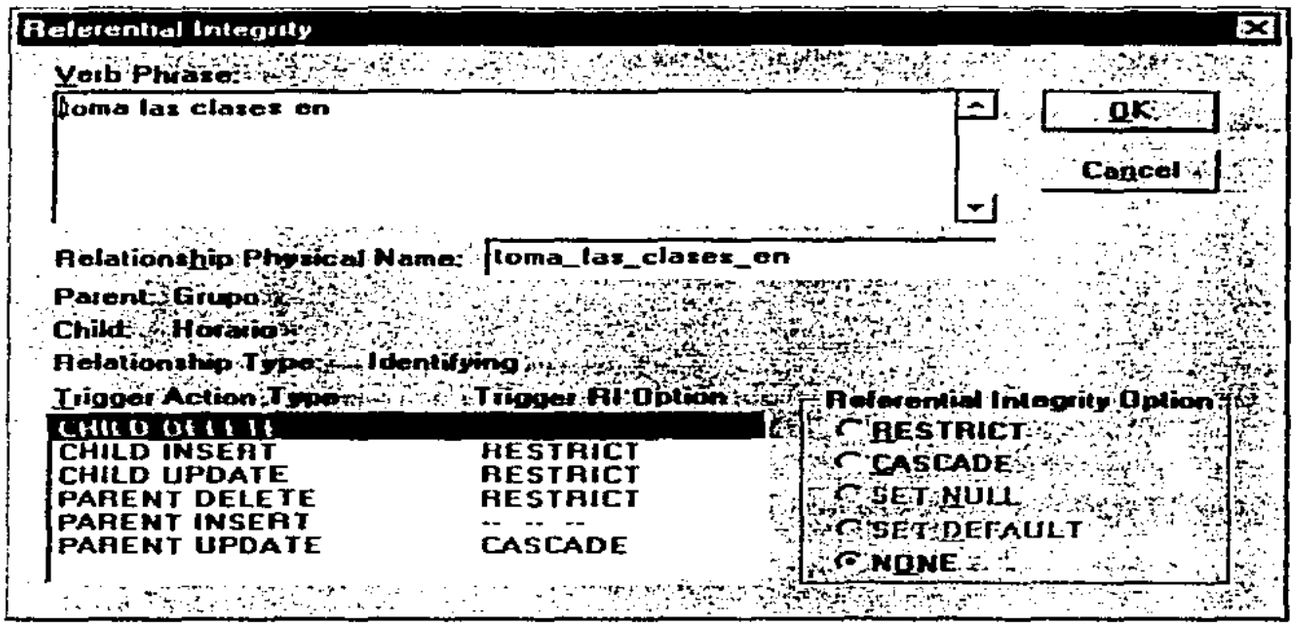


Figura 5.1.3.6 Definición de integridad referencial.

Acción	Opción de integridad referencial del trigger
Borrar en la hija.	No hay restricción.
Insertar en la hija.	Es necesario que exista su registro padre.
Actualizar en la hija.	Es necesario que exista su registro padre.
Borrar en el padre.	Es indispensable que no tenga hijos el registro que se quiere borrar.
Insertar en el padre.	No hay restricción.
Actualizar en el padre.	Se actualizan en cascada o automáticamente los registros hijos que tenga este padre.

8. - Triggers especiales.

Para el caso del sistema de inscripciones, se crearon seis tipos de triggers aplicables a varias bases de datos, definidos en el capitulo 4, que fueron creados desde una herramienta denominada Windows Interactive SQL, y posteriormente se incorporaron al esquema de triggers de ERwin.

9. - Indices.

Para cada una de las entidades se define los índices que tendrá, y dentro de cada uno de ellos se detallan los campos que lo forman, el tipo de índice, si es único, si contiene elementos de la llave foránea y si se trata de un índice físico denominado "clustered".

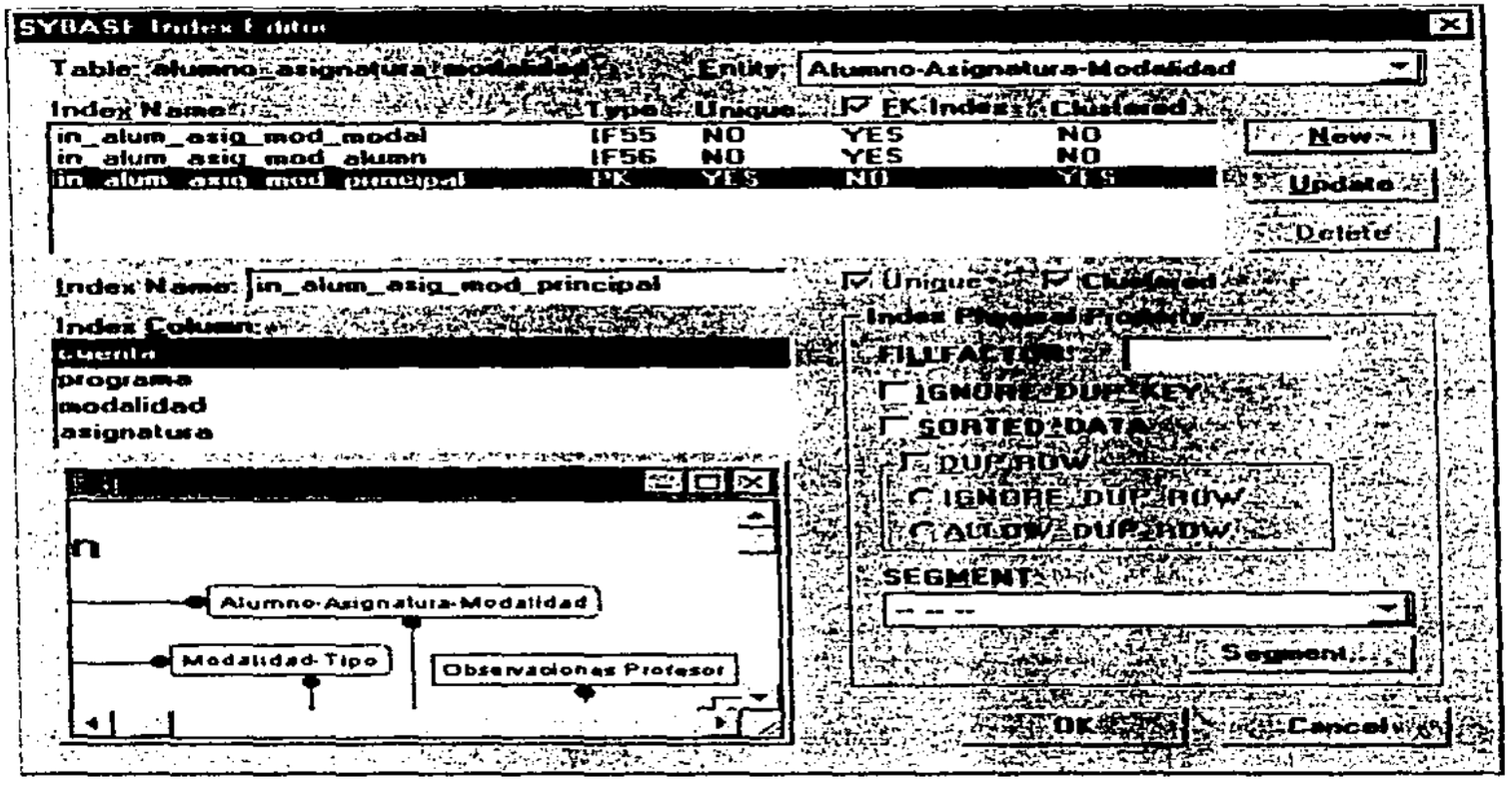


Figura 5.1.3.7 Definición de índices.

10. – Descripción de entidades, atributos y relaciones.

Es muy importante a lo largo del desarrollo y para fines de mantenimiento, tener una descripción completa y simple de las entidades, sus atributos y las relaciones existentes. Para lograr este objetivo, ERwin proporciona facilidades para que para cada elemento se introduzca texto explicativo del mismo.

11. - Generación del modelo físico.

Una vez que se tiene definido todo lo anterior a satisfacción, se procede a generar físicamente la base de datos desde ERwin. Para esto, primeramente se define la plataforma destino del modelo creado, que en nuestro caso es Sybase. Esto se realiza en Server/Target Server.

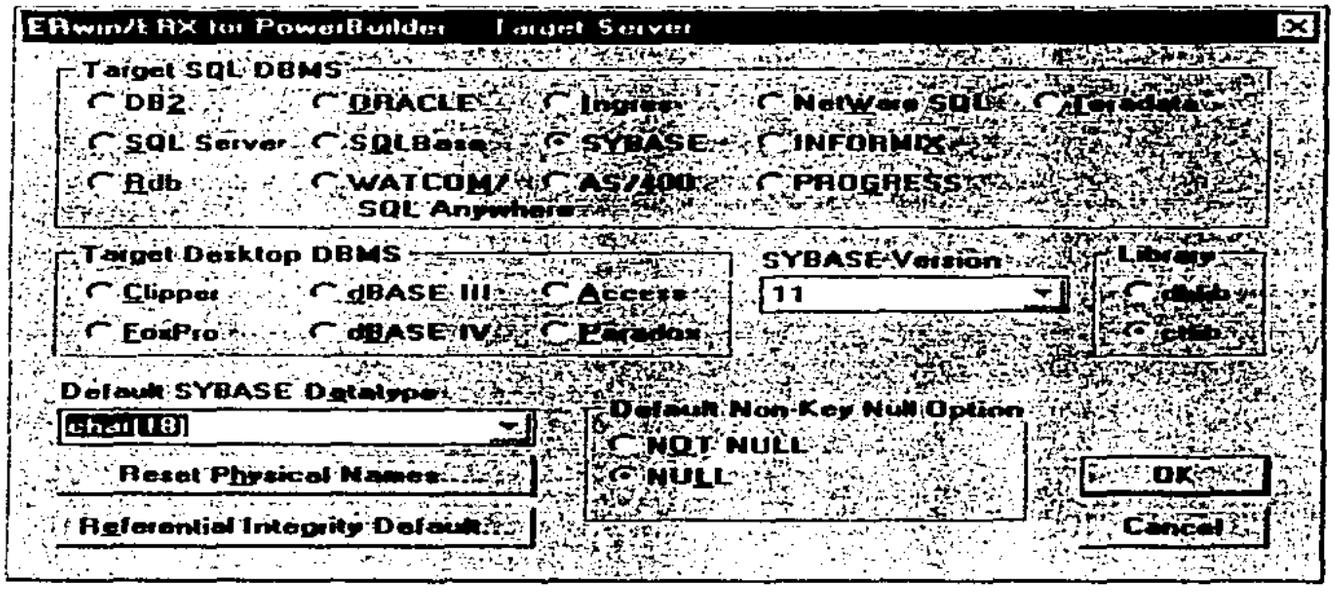


Figura 5.1.3.8 Elección de la base de datos.

En segundo término, se hace la conexión al servidor usando la opción Server/SYBASE Connection. Aquí se debe introducir el nombre del usuario y su password, mismo que deberá tener derechos de administrador.

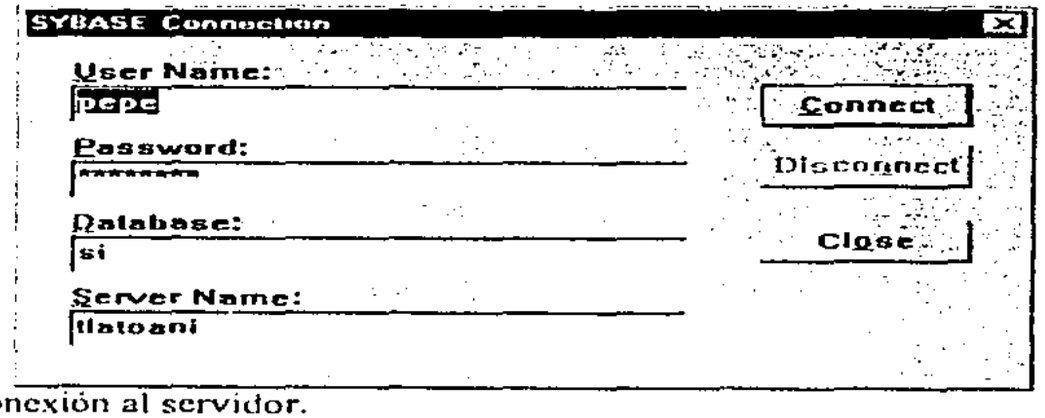


Figura 5.1.3.9 Conexión al servidor.

Finalmente, se procede a la generación del esquema usando la opción Server/SYBASE Schema Generation. Aquí se puede elegir generar: las llaves primarias y foráneas, los triggers, las opciones para las tablas, las reglas, los defaults, los tipos de usuarios y los índices.

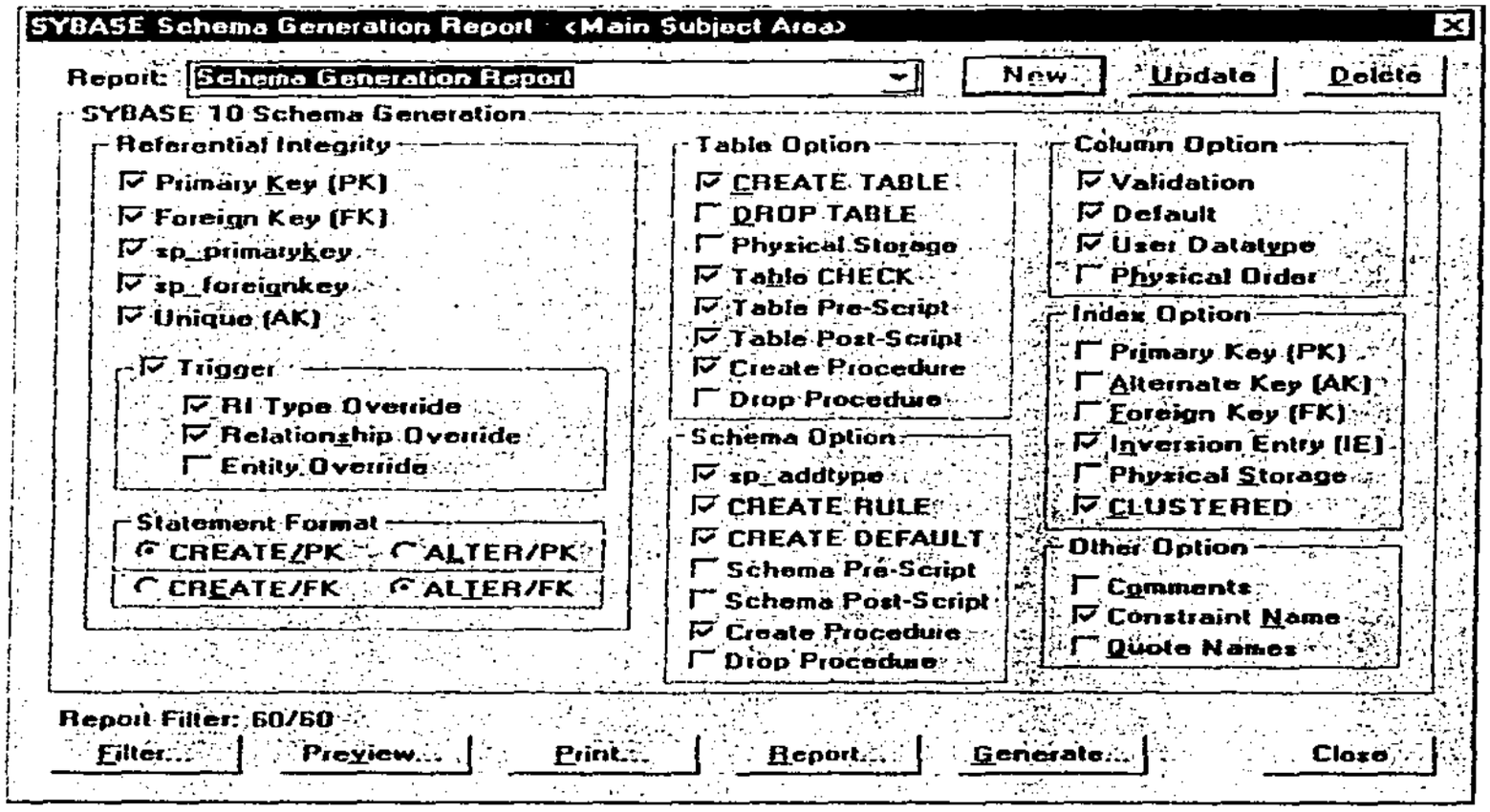
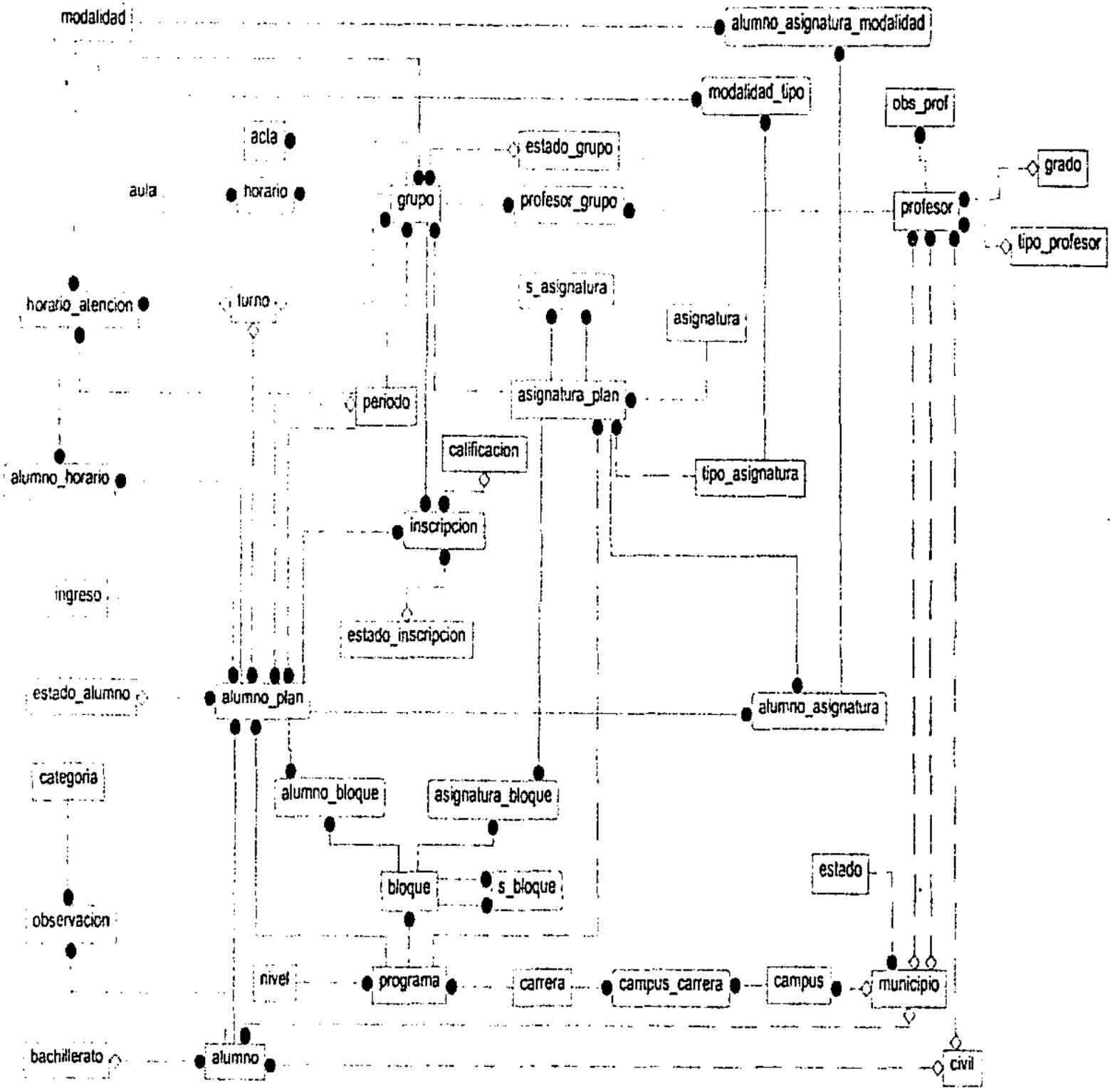


Figura 5.1.3.10 Generación del esquema físico.

Es posible ver una vista preliminar del código SQL que se ejecutará en el servidor, e inclusive se puede definir un archivo de resultados, por si se desea éstudiarlo posteriormente o ejecutarlo desde otra parte.

El código completo de la generación de la base de datos viene en el Apéndice B.

Diagrama físico



parametros

dgae_inscripcion_e

dgae_abc_inscripcion

dgae_abc_ha

dgae_abc_grupo

dgae_c_carrera

dgae_c_unidad_academica

dgae_b_inscripcion

dgae_b_alumno

dgae_carreras

dgae_asignaturas

dgae_alumnos

dgae_ha

dgae_r_ha

dgae_inscripcion_s

dgae_actas_emitidas

Migración de datos del viejo al nuevo sistema.

Como se cambió radicalmente de ambiente, fue necesario trasladar la información útil del viejo sistema que reside en Paradox 3.5 a Sybase 11. Para ello se utilizó la herramienta Pipeline de PowerBuilder.

El procedimiento consiste en definir, a ambas tablas, los campos que se utilizarán y los tipos de datos de la tabla destino. Posteriormente, se establece la conexión entre tablas origen y destino. Una vez realizado esto, se especifica el origen de los datos:

a) Quick Select. Para datos en forma de tabla y con llave.
b) SQL Select. Para relacionar a tablas que no tienen llaves en común.
c) Query. Para datos definidos por una consulta.
d) Stored Procedure. Cuando los datos se encuentran definidos por un procedimiento almacenado.

El siguiente paso es seleccionar las columnas a exportar y si se desea algún criterio de ordenamiento.

Finalmente, se indica lo que se hará con los datos de la tabla origen, que en nuestro caso es abrir la tabla destino y agregar los registros del origen.

5.2

Desarrollo de la aplicación cliente en PowerBuilder

PowerBuilder es un ambiente de desarrollo para aplicaciones gráficas. Como se comentó en el punto 1.5.14, esta herramienta nos permite crear aplicaciones cliente/servidor que trabajan con bases de datos. Sus características principales son:

- Las aplicaciones hechas con PowerBuilder tienen los elementos estándar de cualquier desarrollo en ambiente windows: menús, ventanas, botones, campos de edición, checkboxes, listboxes, etc.

- Reaccionan a eventos activados por los usuarios, y lo que se programa es lo que ocurrirá cuando un evento se dispare. Estos programas se escriben en Powerscript, el lenguaje de PowerBuilder, y se conforman de comandos, funciones y declaraciones que realizan cierta acción en respuesta a un evento.

- Cada ventana o menú que se crea con PowerBuilder es un módulo denominado objeto. En este sentido poseen propiedades, eventos y funciones. Las características de encapsulamiento, herencia y polimorfismo, se pueden implementar en el código generado.

- Es posible ejecutar la misma aplicación en diferentes plataformas, v.gr. crear el código bajo windows y ejecutarlo en UNIX.

- PowerBuilder tiene acceso a una amplia variedad de bases de datos por dos medios: ODBC (Open Database Connectivity), que es el estándar de conectividad de bases de datos de Microsoft, en el cual se utilizan manejadores para cada DBMS. Por otro lado, se tiene la interface ODBC de Powersoft, que es una conexión nativa a una base de datos. Cada interface de base de datos de Powersoft tiene su propia DLL que la comunica con la base de datos específica a través de la API (Application Programming Interface) del vendedor.

Dentro del ambiente de PowerBuilder lo primero que aparece es la ventana de inicio de sesión.

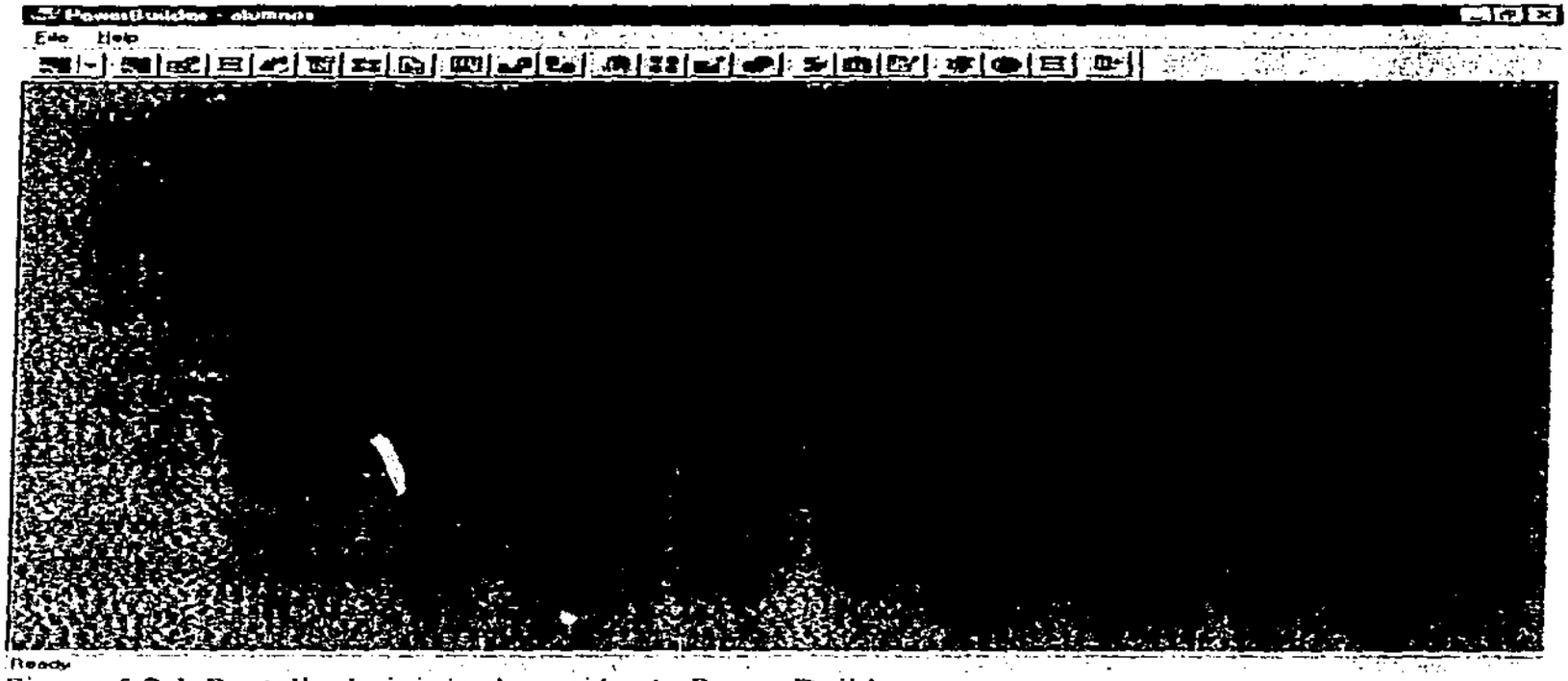


Figura 5.2.1 Pantalla de inicio de sesión de PowerBuilder.

Como podemos observar, en la parte superior se encuentran una serie de íconos que dan acceso a herramientas de propósito específico denominadas painters.

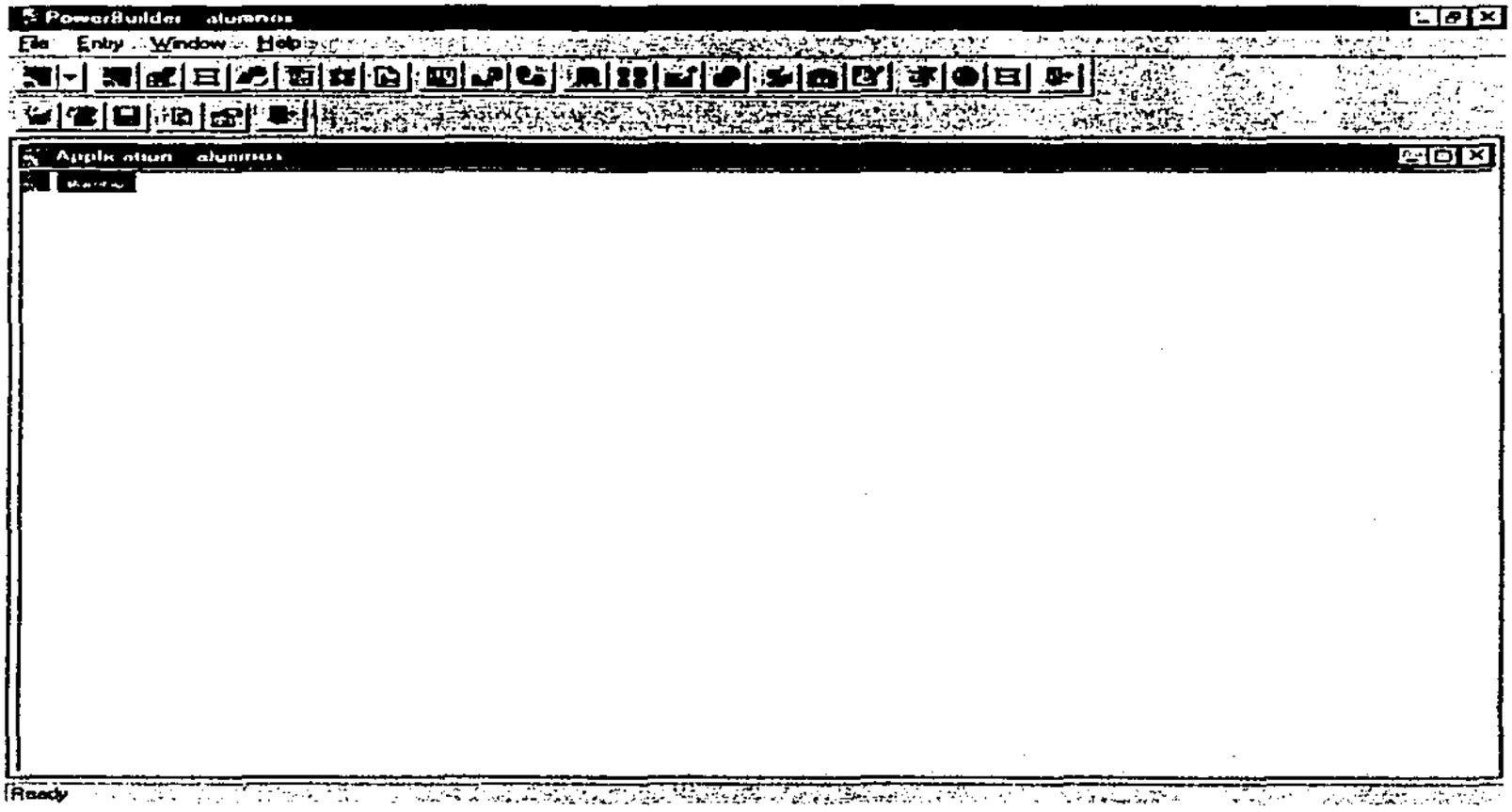


Figura 5.2.2 Pantalla del application painter.

Para crear una aplicación se selecciona File/New, se indica el nombre de la librería, donde residirá y el nombre que tendrá la aplicación. Al término de esta secuencia, aparecerá la ventana de la figura 5.2.2 con el nombre de la aplicación que se ingresó. Para abrir una aplicación existente se sigue un proceso análogo desde File/Open.

Una vez creada la aplicación, es necesario incorporarle los elementos que la conformarán. Uno de los más importantes para fines del sistema de inscripciones es la datawindow.

Para crear una datawindow se hace click en el icono de datawindow y aparece la ventana de selección.

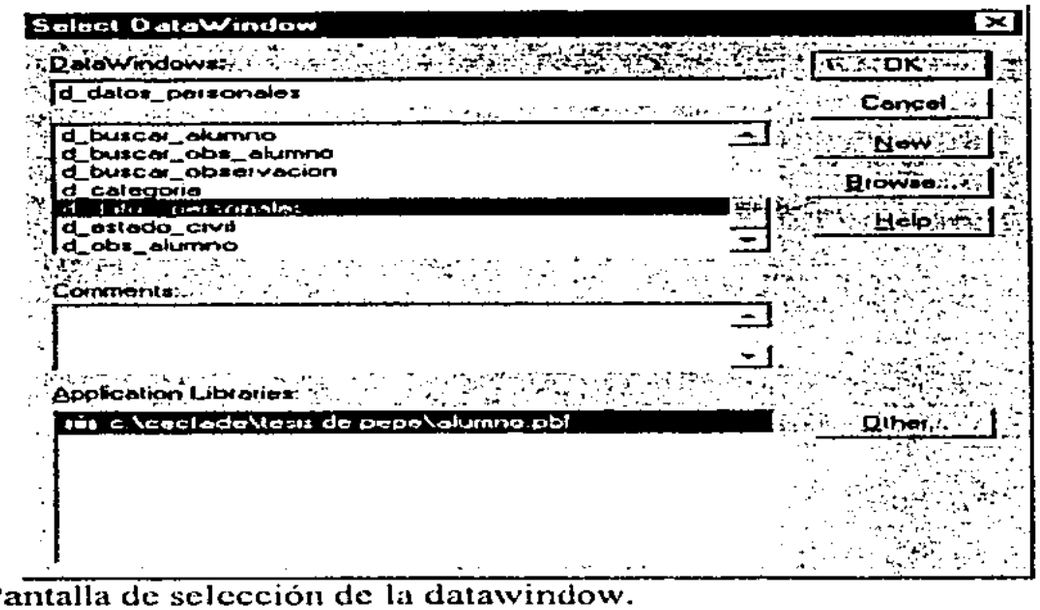


Figura 5.2.3 Pantalla de selección de la datawindow.

Al indicar New aparece la ventana para seleccionar la fuente de los datos (data source) y el estilo de presentación (presentation style). Cuando esto se concluye, se procede con el diseño de la datawindow.

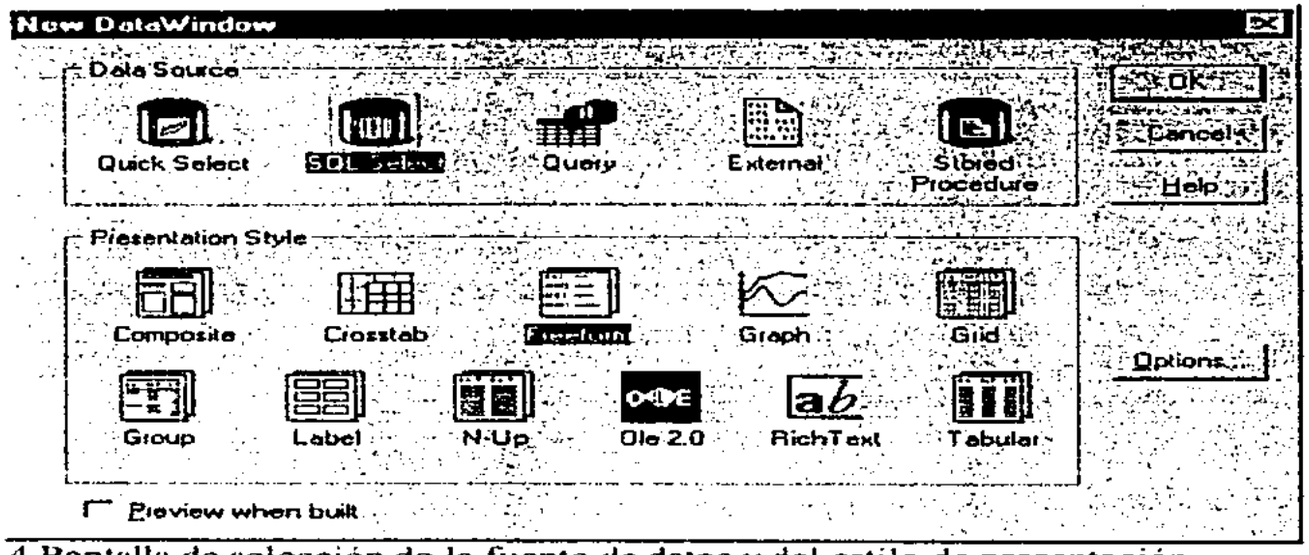


Figura 5.2.4 Pantalla de selección de la fuente de datos y del estilo de presentación.

Cuando se pulsa OK a la pantalla anterior, aparece otra con las tablas del sistema definidas en Sybase, y se seleccionan las que se deseen:

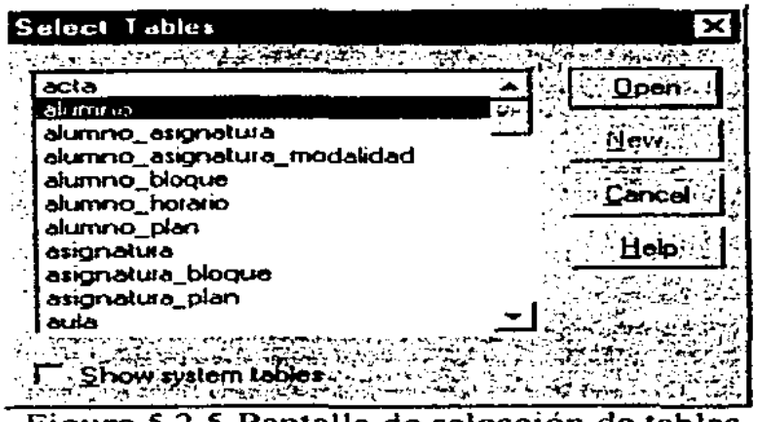


Figura 5.2.5 Pantalla de selección de tablas.

Al seleccionar la tabla alumno, aparece otra pantalla con el detalle de sus campos a seleccionar.

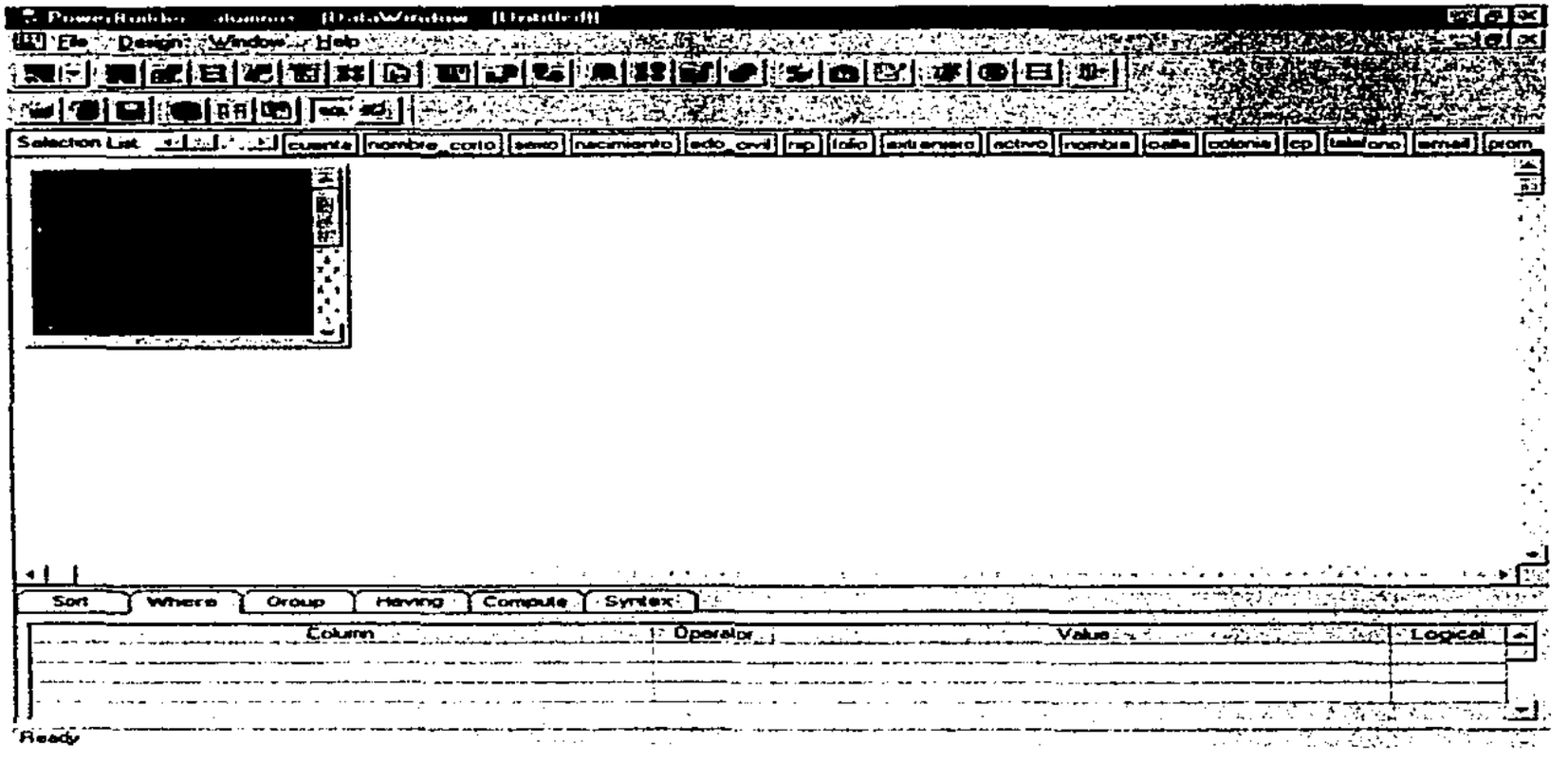


Figura 5.2.6 Pantalla de la tabla con sus campos respectivos seleccionados.

Para fines del ejemplo, el funcionamiento deseado de la datawindow es que el número de cuenta sea un dato necesario para operar el resto de los campos. Esto se logra especificando en la selección SQL que el campo alumno.cuenta=:cuenta, además es preciso declarar en Design/Retrieval arguments la variable cuenta como campo numérico.

Al dar click al ícono SQL aparecen los campos en el orden en que fueron seleccionados.

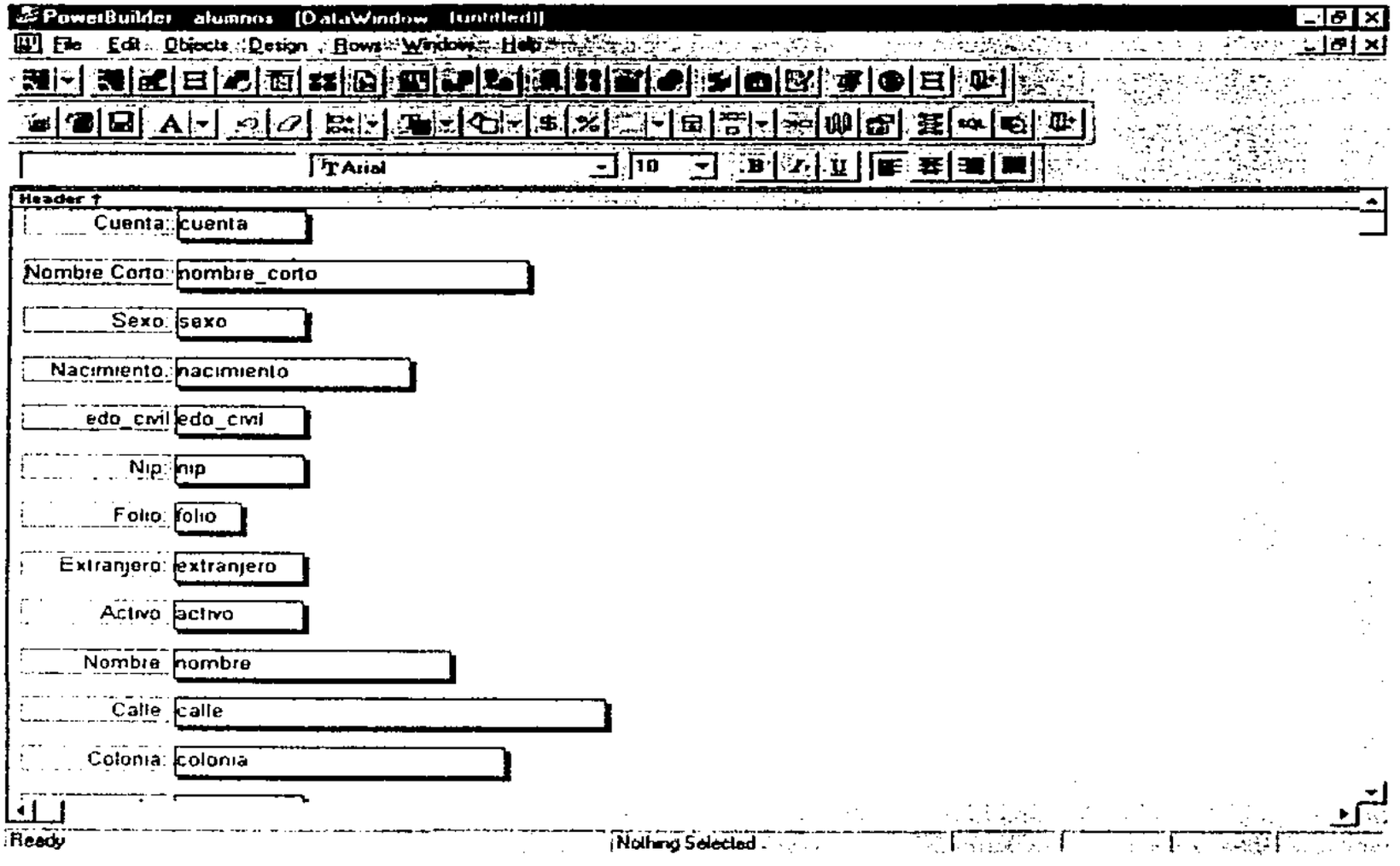


Figura 5.2.7 Pantalla de la datawindow con los campos seleccionados.

Es posible modificar la apariencia y el orden de los campos de modo que sean más agradables para el usuario. Esto se logra dando click derecho del ratón sobre cada elemento y seleccionando la opción properties. En esta parte es posible cambiar: el estilo del borde, la alineación, el tipo de letra y las propiedades de edición.

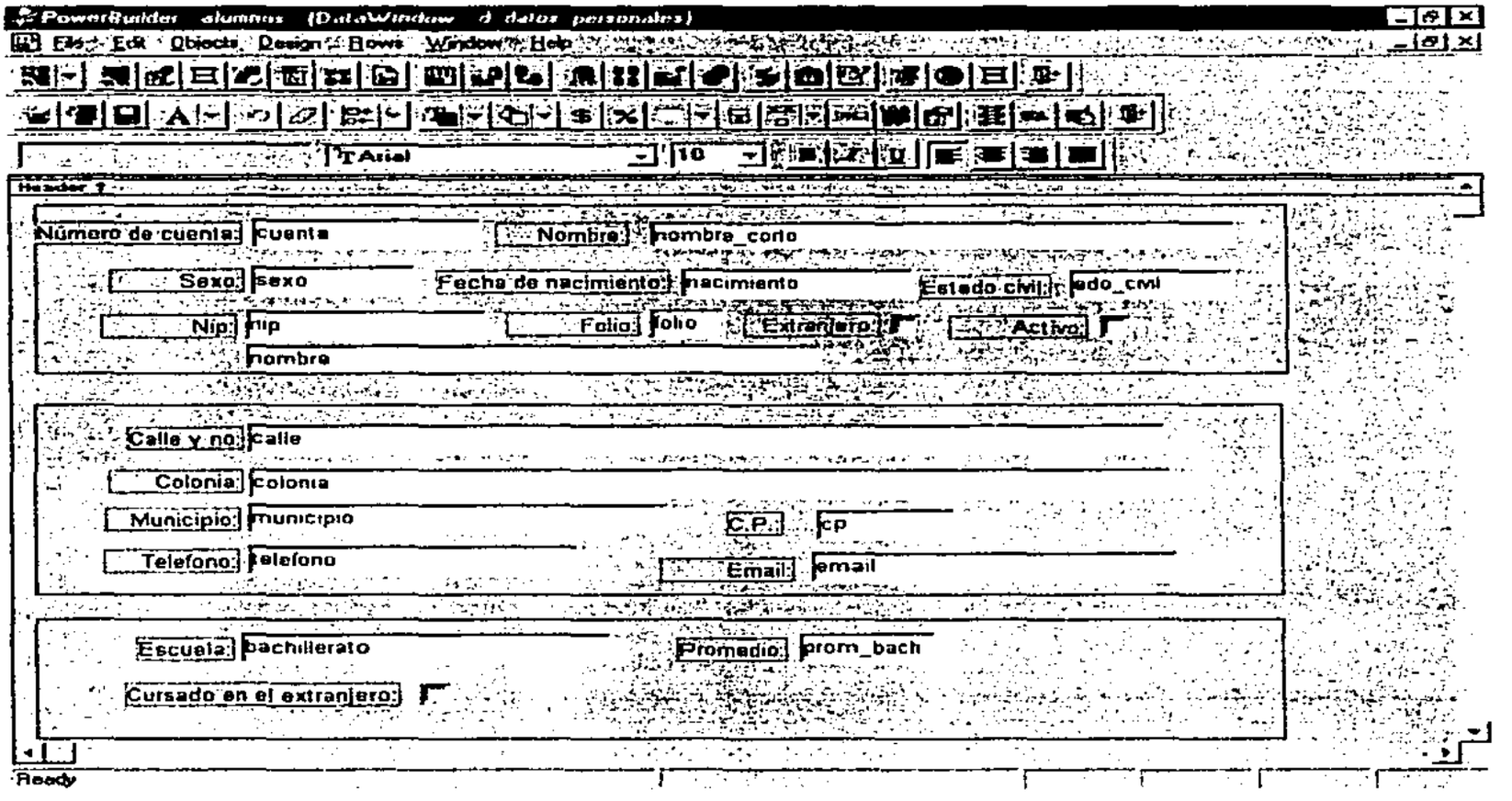


Figura 5.2.8 Pantalla de la datawindow con modificaciones a la apariencia de los campos.

Otro aspecto que se debe definir es las propiedades de actualización de la datawindow. Esto se realiza en Rows/Update Properties.

Al concluir la definición de la datawindow se guarda en File/Save con el nombre deseado.

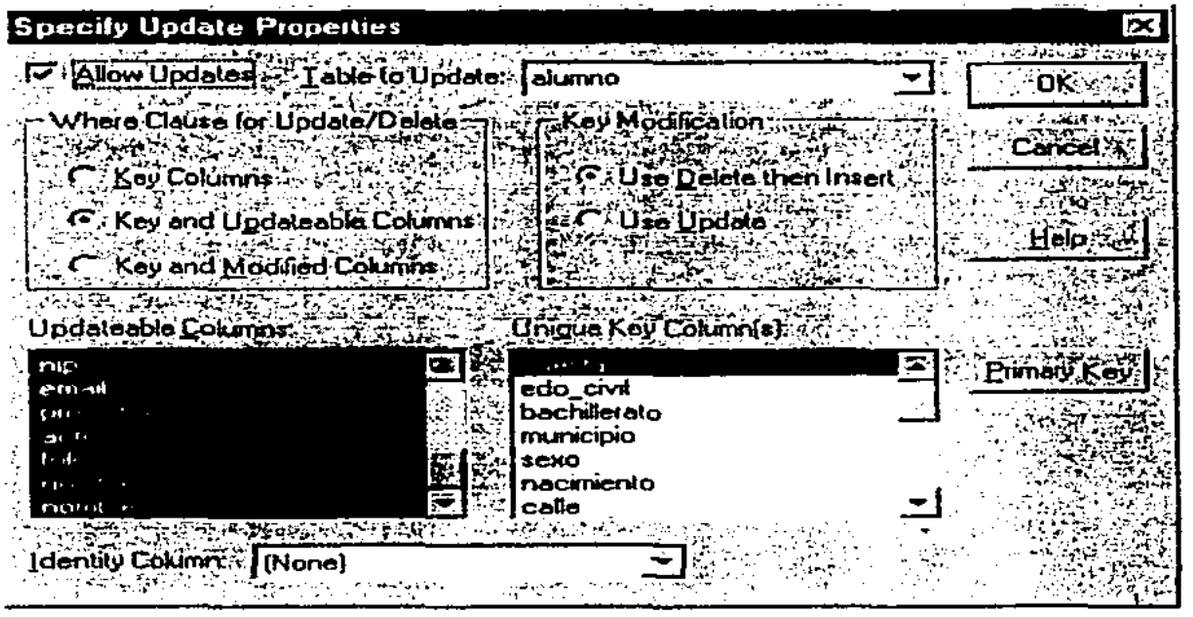


Figura 5.2.9 Pantalla de definición de propiedades de actualización de la datawindow.

Para poder visualizar la datawindow que se definió, en el painter de window se crea una nueva ventana y se selecciona Controls/DataWindow. Una vez que se da al control el tamaño deseado, se le incorpora la datawindow que se ha estado construyendo y se guarda con el nombre que corresponda.

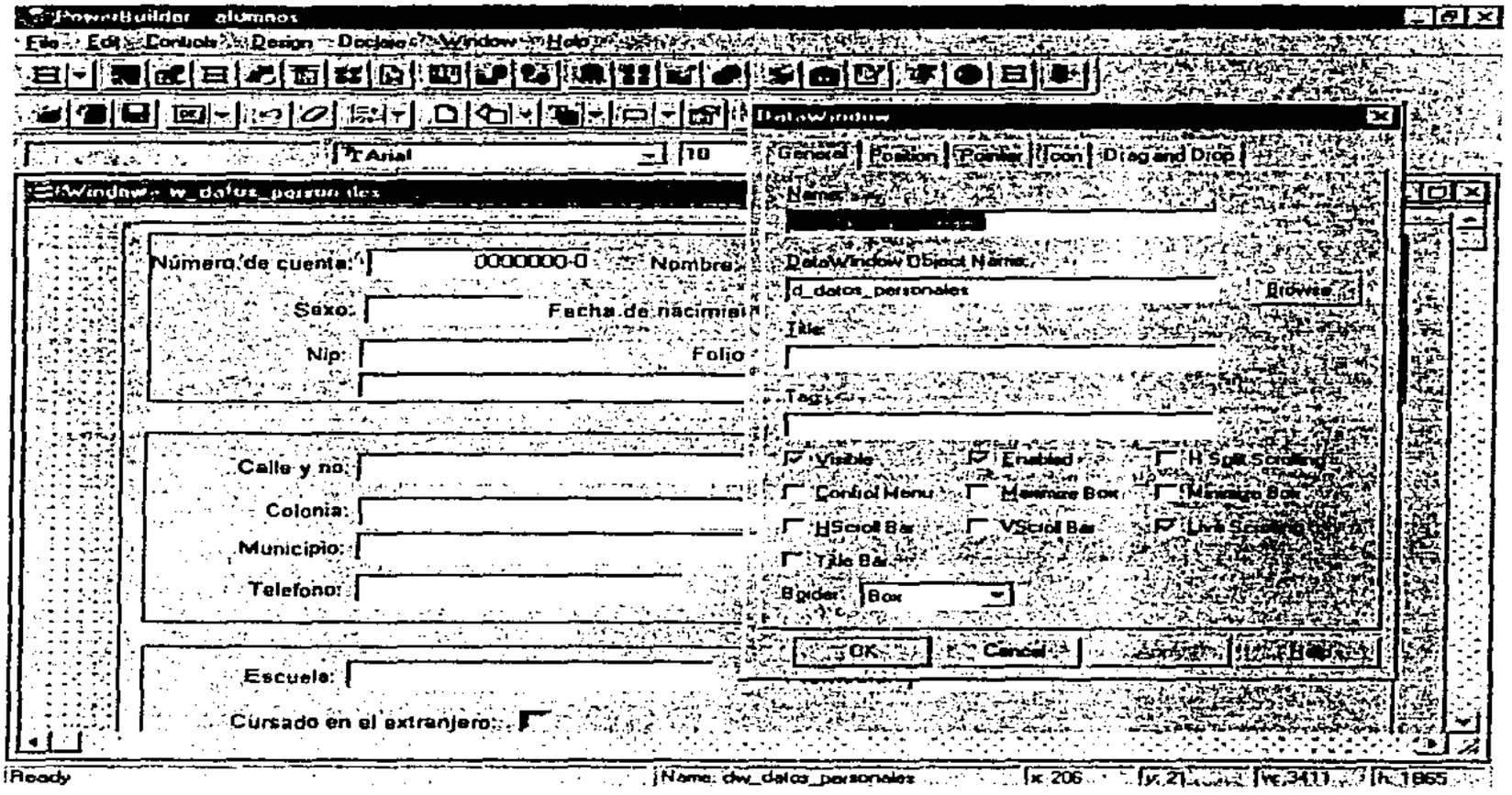


Figura 5.2.10 Pantalla de incorporación de la datawindow al control.

Para incorporar botones de comando a la ventana, el proceso consiste en seleccionar Controls/CommandButton y dar click, en donde se desee que quede posicionado el botón. El nombre del botón se asigna pulsando click derecho, seleccionando Properties y tecleando el texto deseado.

La definición del código que contendrá cada botón se hace en el Script. A continuación se analizarán los scripts de cada uno de los botones de la pantalla de datos generales de alumnos, que finalmente queda como en la figura 5.2.11.

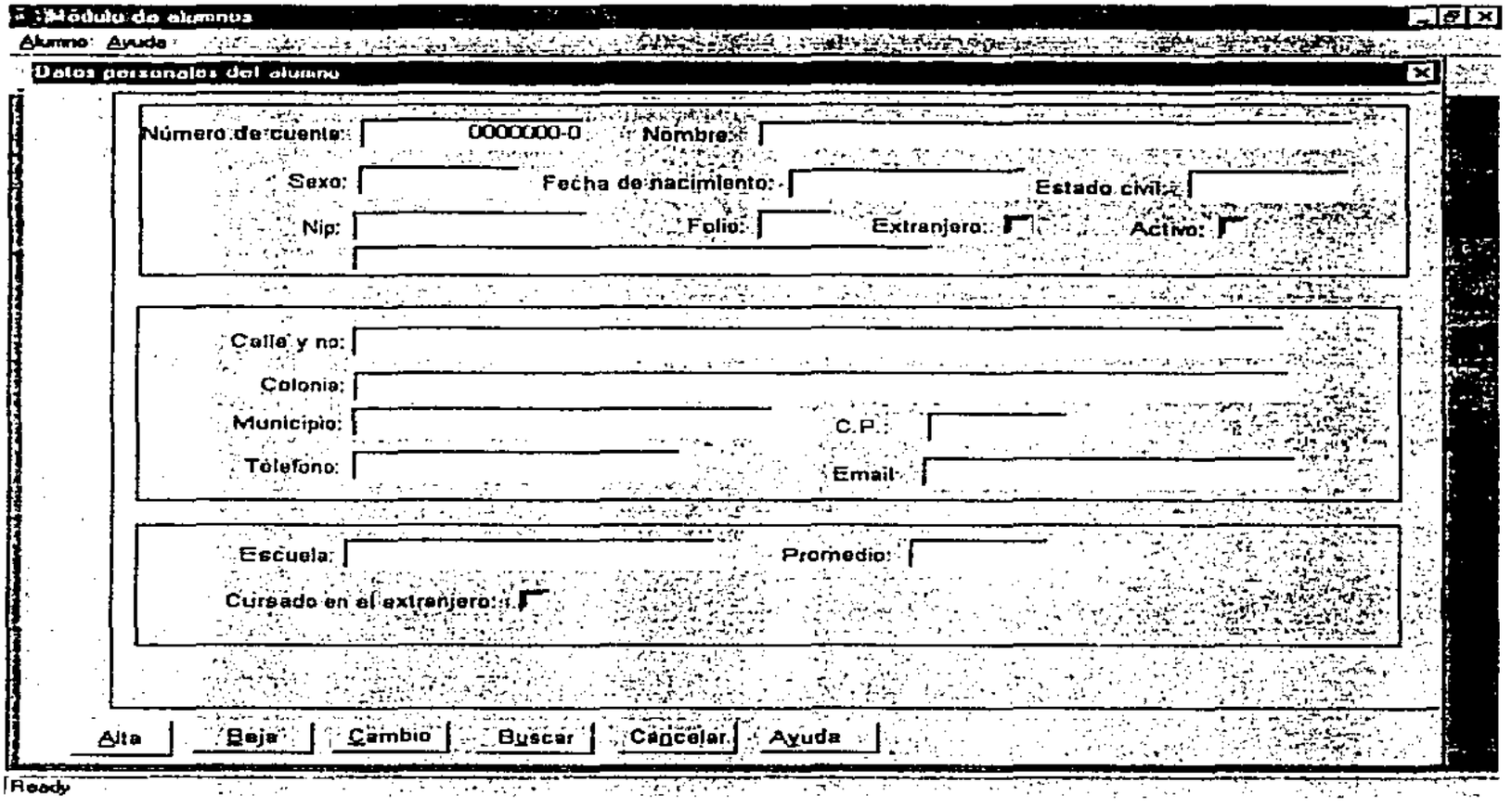


Figura 5.2.11 Pantalla final en ejecución de datos personales de alumnos.

En alguna zona de la pantalla donde no exista ningún elemento se debe insertar el siguiente código, dando click al botón derecho del ratón y entrando a la opción Script.

```
dw_datos_personales.settrans(sqlca)
dw_datos_personales.reset()
dw_datos_personales.insertRow(0)
```

La primer instrucción asigna el objeto de transacciones sqlca a la datawindow datos personales. La segunda línea borra cualquier dato de la datawindow y la tercera inserta un registro en blanco.

El código que contiene el botón de Alta es:

```
dw_datos_personales.update()
if sqlca.sqlcode =-1 THEN
    messagebox("Error !!",sqlca.sqlerrtext,StopSign!)
else
        dw_datos_personales.reset()
        dw_datos_personales.insertrow(0)
end if
```

La primera línea actualiza la datawindow denominada datos personales, posteriormente hay una condición en la cual en caso de existir un error con el objeto de transacciones, envía un cuadro de texto explicativo del mismo a pantalla; si no hay ningún problema borra los datos que aparecen en la datawindow e inserta un registro en blanco.

El código que contiene el botón de Baja es:

```
Long cuenta
if validacion_baja() then
        cuenta=(dw_datos_personales.getitemNumber(dw_datos_personales.getrow(),"cuenta"))
        DELETE FROM alumno
  WHERE alumno.cuenta=:cuenta;
        COMMIT;
        dw_datos_personales.reset()
        dw_datos_personales.insertrow(0)
end if
```

Primeramente se define una variable "cuenta" de tipo long. Después, dentro de la función validación baja, envía un cuadro de texto a pantalla para confirmar si se prosigue con la baja, en caso afirmativo, obtiene el registro que corresponde al número de cuenta tecleado, borra el

registro e indica que se completó la transacción. Finalmente, borra los datos que aparecen en la datawindow e inserta un registro en blanco.

El código que contiene el botón de Cambio es:

```
dw_datos_personales.update()
dw_datos_personales.reset()
dw_datos_personales.insertrow(0)
```

Actualiza la datawindow con los datos que se tengan en pantalla, borra los datos que aparecen en la datawindow e inserta un registro en blanco.

El código que contiene el botón de Buscar es:

```
open(w_buscar_alumnos)
```

Abre la ventana w_buscar_alumnos, que contiene lo siguiente:

```
d_buscar_alumno.reset()
if rb_nombre.Checked then
  d_buscar_alumno.retrieve(0,sle_1.text)
else
  d_buscar_alumno.retrieve(long(sle_1.text),"")
end if
d_buscar_alumno.setfocus()
cb_aceptar.default = true
cb_buscar.default = false
if d_buscar_alumno.RowCount()<1 then
        MessageBox("Error ", "No se encontró ningun registro")
        st_1.setfocus()
end if
```

Primeramente, borra los datos de la datawindow buscar_alumno, dependiendo del radiobutton seleccionado, pasa los parámetros para el retrieve que corresponda, ya sea buscar por número de cuenta o por nombre. Posteriormente manda el foco al primer registro, asigna estados a los botones de comando y en caso de que la búsqueda no haya sido exitosa, manda un cuadro de diálogo explicando esta situación.

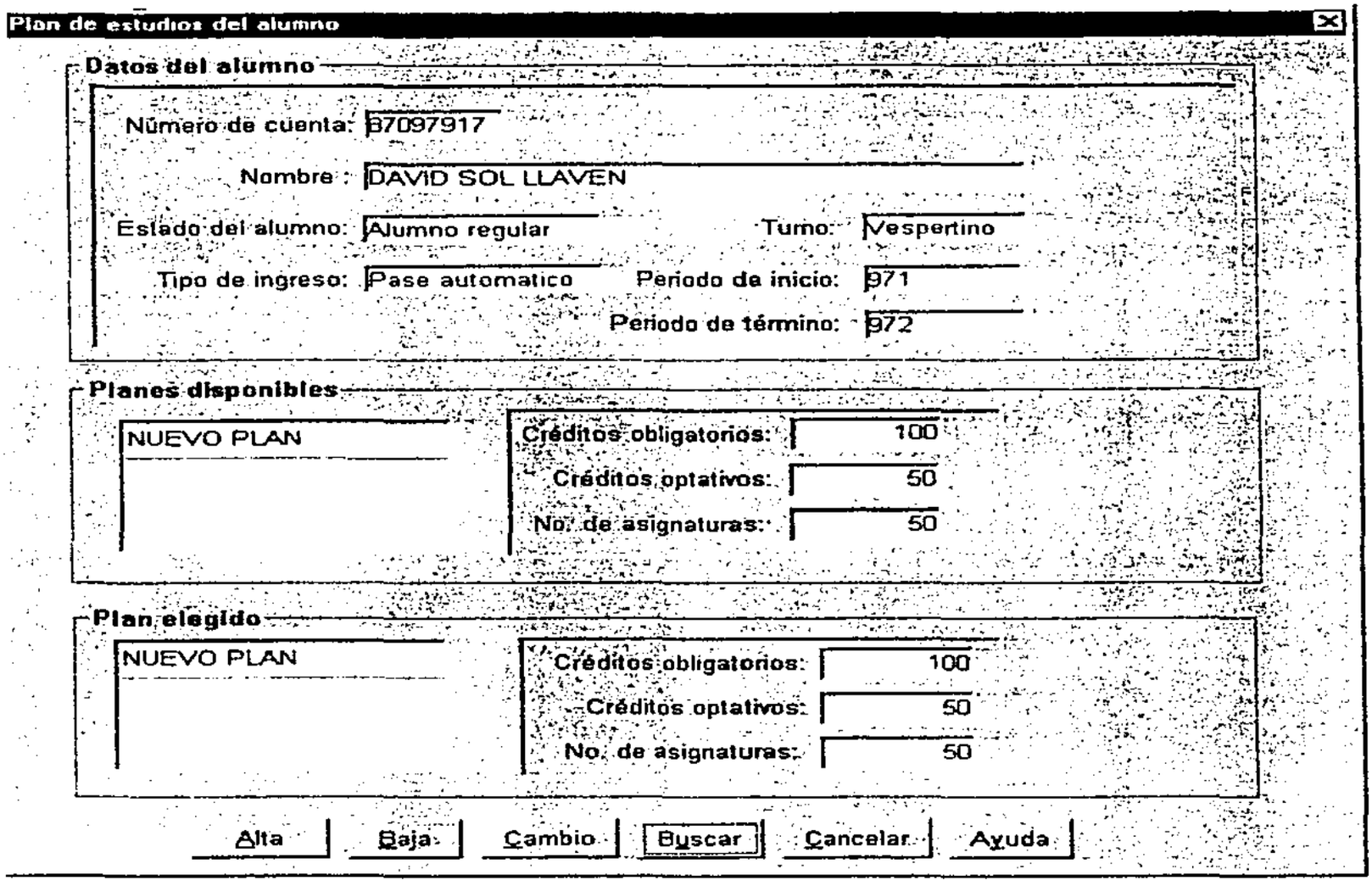
El código que contiene el botón de Cancelar es:

```
close(parent)
```

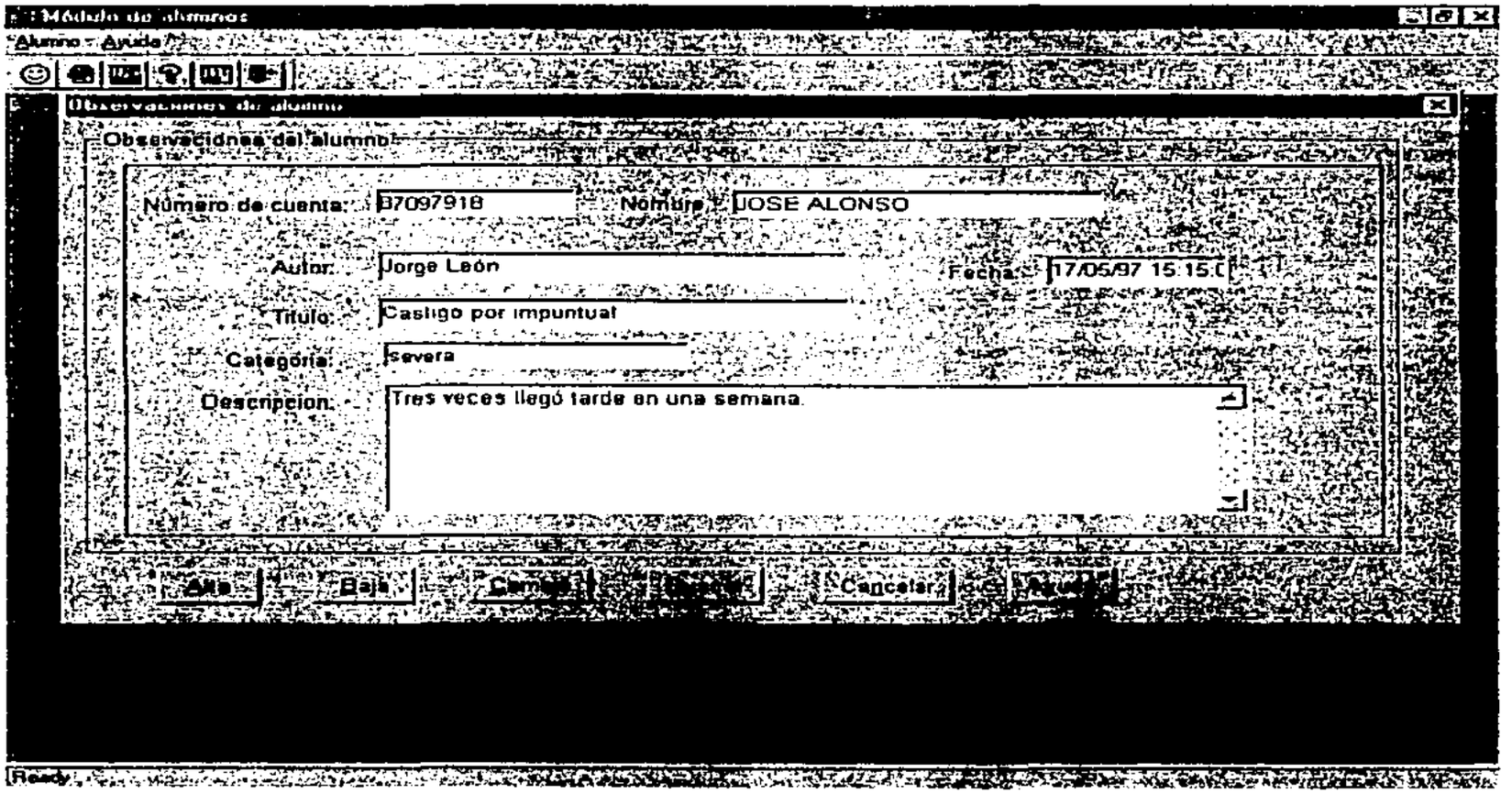
Simplemente cierra la ventana padre.

Pantallas muestra por módulo del sistema en ejecución.

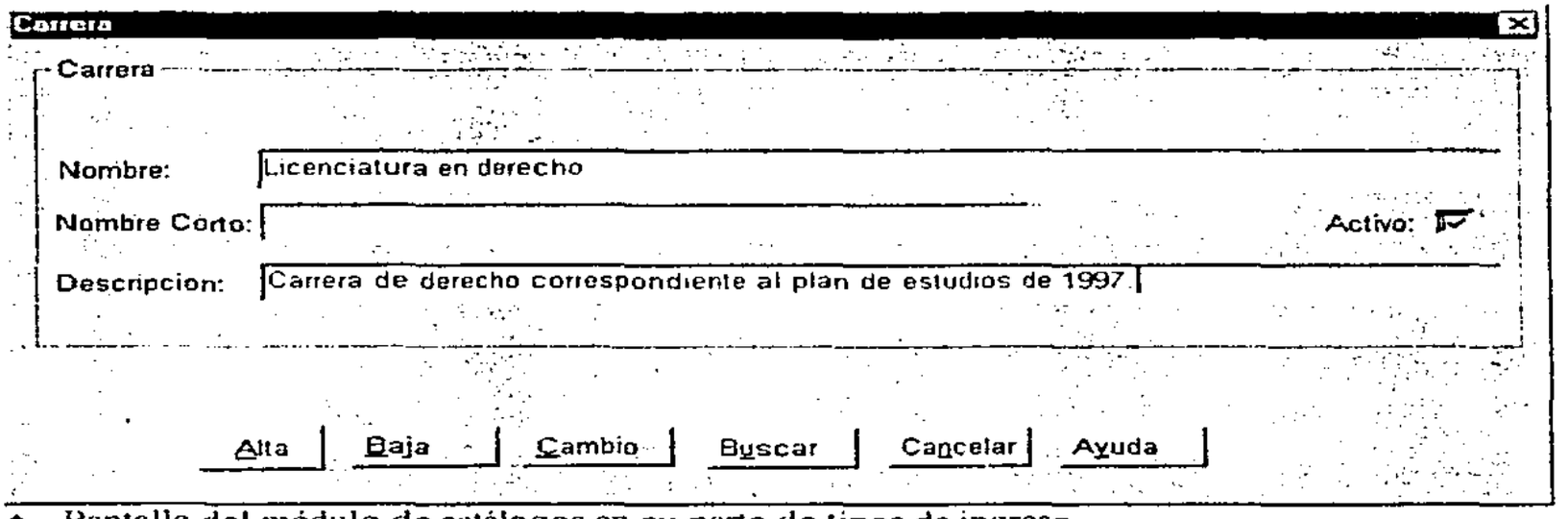
- Pantalla del módulo de alumnos correspondiente al plan de estudios.



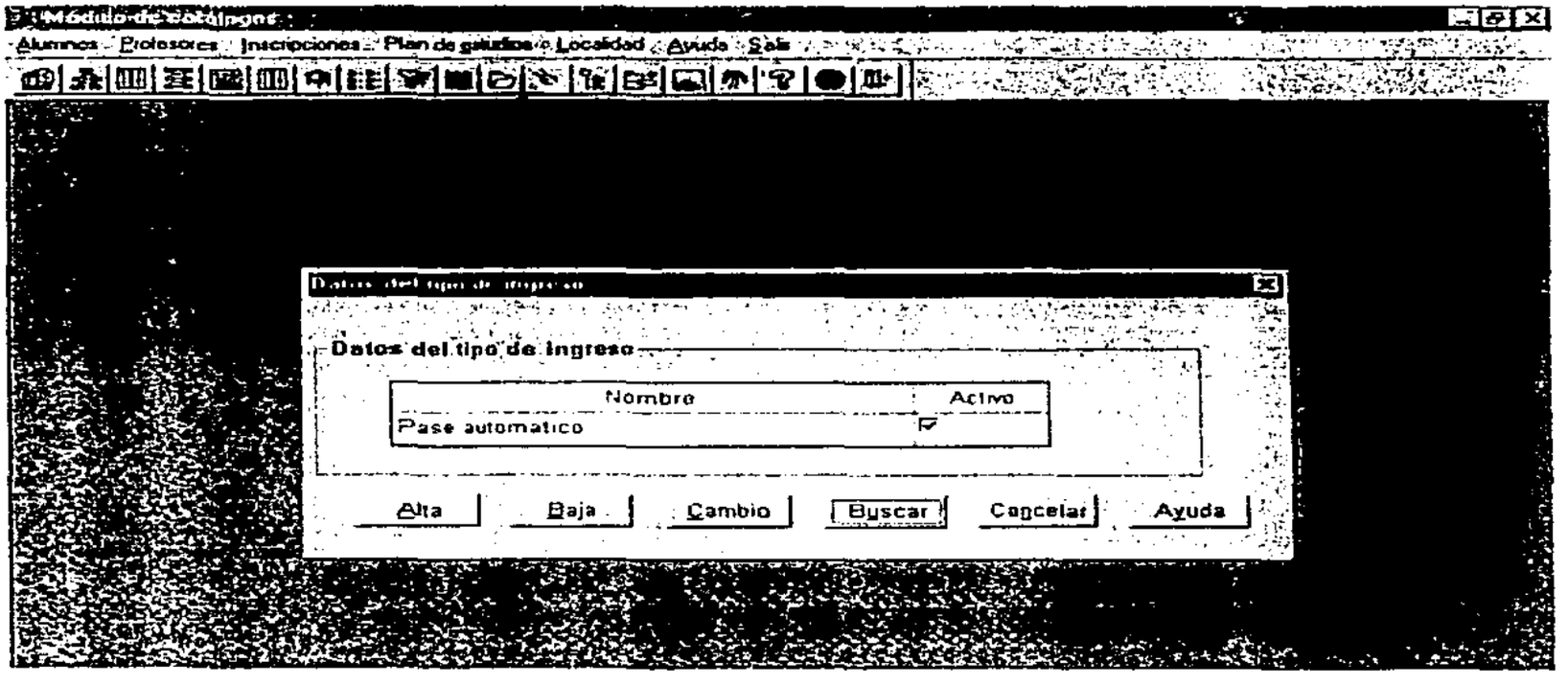
- Pantalla del módulo de alumnos correspondiente a observaciones del alumno.



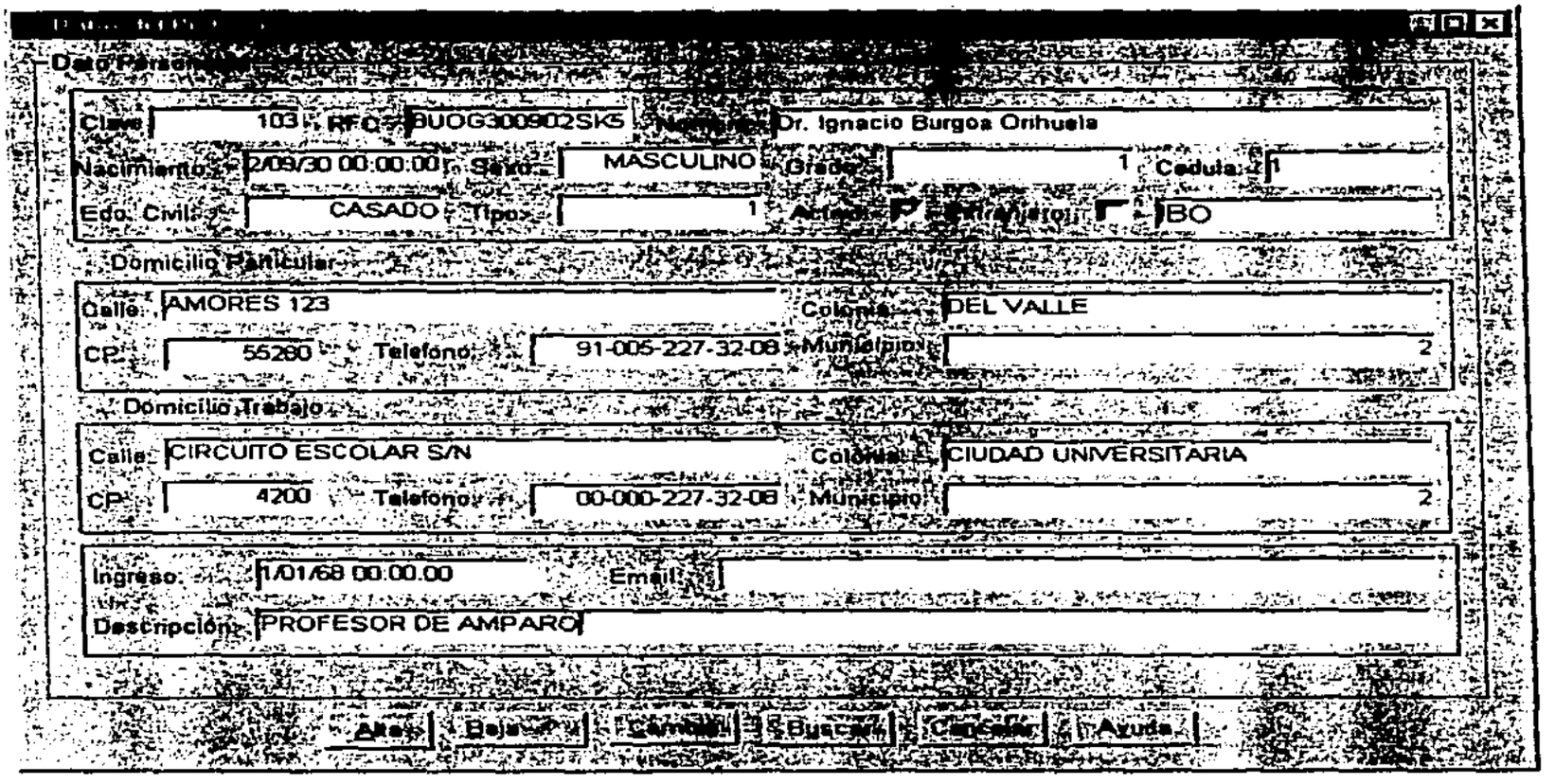
- Pantalla del módulo de campus-carrera correspondiente a la administración de carreras.



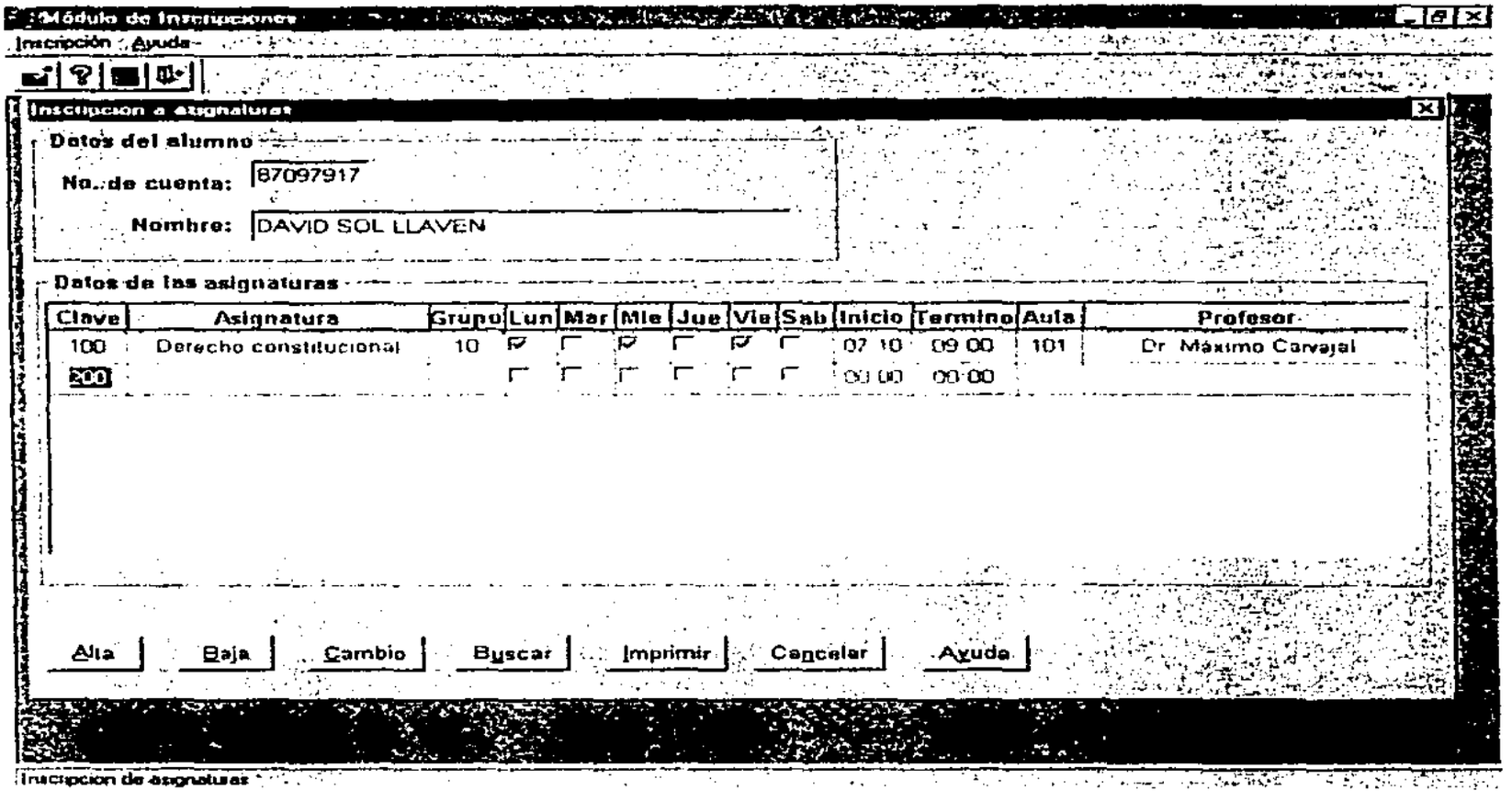
- Pantalla del módulo de catálogos en su parte de tipos de ingreso.



- **Pantalla de datos generales del módulo de profesor.**



- Pantalla de inscripción y movimientos a asignaturas del módulo de inscripciones.



El código que se creó respecto al objeto de transacciones es:

```
dw_sp_val_alumno.settrans(sqlca)
dw_sp_val_alumno.reset()
dw_sp_val_asignatura.settrans(sqlca)
dw_sp_val_asignatura.reset()
dw_sp_val_asignatura.insertrow(0)
dw_sp_val_grupo.settrans(sqlca)
dw_sp_val_grupo.reset()
dw_sp_val_grupo.insertrow(0)
dw_sp_val_inscritos.settrans(sqlca)
dw_sp_val_inscritos.reset()
dw_sp_val_inscritos.insertrow(0)
```

Lo que se hace en este código es asignar el objeto de transacciones sqlca a las datawindow cuyas datasource son procedimientos almacenados, éstos son: validaciones del alumno, validaciones de la asignatura, validaciones de grupo y de inscritos. Los detalles de cada uno se especifican en el punto 4.2.2.2.5, y el código SQL se encuentra en el Apéndice B.

El código que se ejecuta al dar tabulador en el campo de cuenta es:

```
double cuenta
dw_sp_val_alumno.reset()
em_cuenta.GetData(cuenta)
dw_sp_val_alumno.retrieve(cuenta)
dw_sp_val_inscritos.retrieve(cuenta)
renglon=dw_sp_val_inscritos.rowcount()
renglon ++
dw_sp_val_inscritos.insertrow(0)
dw_sp_val_inscritos.setrow(renglon)
dw_sp_val_inscritos.SetFocus()
```

Lo que se hace es definir una variable "cuenta", traer todos los registros de la datawindow dw_sp_val_alumno, obtener el número de cuenta de la captura, hacer un retrieve con este número de las datawindow val_alumno y val_inscritos. Cualquier error que se presente en el proceso de validación, se muestra en pantalla vía una caja de diálogo. Si todo procede correctamente, abre un nuevo renglón en el grid.

Al ingresar la clave de la asignatura y cambiar de campo se ejecuta el siguiente código:

```
integer clave
if getcolumnName()="clave" then
        clave = long(gettext())
        dw_sp_val_asignatura.retrieve(clave,long(em_cuenta.text))
        if dw_sp_val_asignatura.rowcount()>0 then
                dw_sp_val_inscritos.object.asignatura[renglon,renglon]=dw_sp_val_asignatura.get
                itemstring(1,"asignatura")
        end if
end if
string grupo
if getcolumnName()="grupo" then
        grupo = (gettext())
        dw_sp_val_grupo.retrieve(grupo)
        if dw_sp_val_grupo.rowcount()>0 then
                dw_sp_val_inscritos.object.lunes[renglon,renglon]=dw_sp_val_grupo.getitemstrin
                g(1,"lun")
                dw_sp_val_inscritos.object.martes[renglon,renglon]=dw_sp_val_grupo.getitemstri
                ng(1,"mar")
                dw_sp_val_inscritos.object.miercoles[renglon,renglon]=dw_sp_val_grupo.getitem
                string(1,"mie")
                dw_sp_val_inscritos.object.jueves[renglon,renglon]=dw_sp_val_grupo.getitemstri
                ng(1,"jue")
                dw_sp_val_inscritos.object.viernes[renglon,renglon]=dw_sp_val_grupo.getitemstr
                ing(1,"vie")
                dw_sp_val_inscritos.object.sabado[renglon,renglon]=dw_sp_val_grupo.getitemstri
                ng(1,"sab")
                dw_sp_val_inscritos.object.aula[renglon,renglon]=dw_sp_val_grupo.getitemstring
                (1,"aula")
                dw_sp_val_inscritos.object.profesor[renglon,renglon]=dw_sp_val_grupo.getitemst
                ring(1,"profesor")
         end if
 end if
```

Primeramente se define una variable "clave" que controlará las claves de asignaturas que el usuario vaya proporcionando. Al momento de ingresarla, se ejecuta la datawindow dw_sp_val_asignatura, que procesa en ese momento el procedimiento almacenado val_asignatura con los parámetros "clave" y "cuenta". Aún cuando las validaciones no hayan sido superadas, se despliega el nombre de la asignatura en el campo correspondiente.

Si todas las validaciones se han pasado correctamente hasta el momento, se define una variable "grupo", y se le asigna el valor que se ingrese por medio de la captura. En este momento se ejecuta el procedimiento almacenado val_grupo, en el cual, se realizan las validaciones mencionadas en el diseño y se despliegan los datos de horario de clases de lunes a sábado, el aula de clases y el nombre del catedrático.

En ese momento el alumno tiene toda la información para decidir si se hace una alta, simplemente dando un click al botón del mismo nombre. Se sigue un proceso similar para todas las asignaturas que se deseen inscribir.

Cuando se trata de una baja, se ejecuta el procedimiento almacenado val_alumno, validando en especial que el número de cuenta exista, que el alumno tenga horario de inscripción y que se esté inscribiendo dentro del mismo. Al final se pulsa el botón de Baja y el movimiento queda registrado.

Para el caso de los cambios, se valida todo lo anterior en val_alumno, más el hecho de que exista cupo en el grupo destino y que no exista traslape entre las asignaturas previamente inscritas. De la misma forma, es necesario oprimir el botón de Cambio para dar por terminado el movimiento.

Cuando todas las altas concluyen se procede a imprimir la tira de materias, pulsando el botón Imprimir.

5.3

Pruebas y ajustes del sistema

5.3.1 Esquema de pruebas utilizado.

El objetivo es ser consistente con los esquemas de control de calidad que existen en la industria de desarrollo de software. En este sentido, se entiende que la tarea de aseguramiento de que un sistema está construido conforme a las especificaciones y sin errores, es un proceso continuo a lo largo del ciclo de vida. Con esta base, al sistema de inscripciones se le aplicaron pruebas desde el análisis, diseño, durante la construcción y en el tiempo específico destinado a pruebas integrales y de volumen.

Las características generales que dominaron el proceso de pruebas durante el desarrollo del sistema fueron de estilo descendente, lo que significa que el orden de las mismas fue:

- Prueba de subsistemas.
- Prueba de módulos.
- Prueba de funciones.

5.3.2 Pruebas durante el desarrollo.

Es muy importante señalar, que la premisa de este desarrollo es que un control de calidad efectivo no se consigue al final del proceso mediante una sola etapa de pruebas, sino que se debe procurar en paralelo con la terminación de cada componente. Es por ello que existió una fase de pruebas durante el desarrollo.

La intención durante las pruebas de subsistemas fue, principalmente, asegurar que las especificaciones definidas en el diseño coincidieran plenamente con lo construido, tanto en forma como en contenido. Esto trajo consigo algunos ajustes a la programación para lograr el objetivo. La ventaja de hacer esto en una etapa temprana del desarrollo del sistema, es que al no tener todos los detalles concluidos, no se presenta el efecto en cadena de arrastrar un error hacia los detalles inferiores.

Cuando esto se concluyó para todos los subsistemas, el siguiente paso fue probar los módulos de cada subsistema en términos de funcionalidad. Finalmente, se probó la función de cada módulo en forma independiente.

5.3.3 Pruebas integrales.

Las pruebas integrales se realizaron al término de todos los subsistemas, y el objetivo era asegurar el correcto funcionamiento del sistema en su totalidad. A pesar de que se hicieron pruebas durante el desarrollo, en esta etapa surgieron errores de programación y de instalación.

5.3.4 Pruebas de volumen.

Esta fase inició una vez que se tenían cubiertos todos los errores y ajustes al diseño y programación aparecidos en las etapas anteriores.

La intención en esta parte fue asegurar que con un volumen de datos reales e incluso superior, el sistema se comportaba de manera estable y de acuerdo a los estimados de tiempo de respuesta. Aunque hubo que realizar ajustes y optimización de algunos triggers en la base de datos, se concluyó con éxito.

5.4

Requisitos operacionales

A continuación se presentan los requisitos deseables de hardware y software para operar el sistema de forma adecuada:

Computadoras cliente.

Procesador	Memoria RAM	Tamaño del disco duro	Software
Pentium a 120 Mhz	16 Mb	120 Mb	. Windows 95 . Cliente de Power Builder v. 5.0

Servidor.

Procesador	Memoria RAM	Tamaño del disco duro	Software
Modelo ULTRA 1 con procesador ULTRASPARC a 143 Mhz	64 Mb	7 Gb	. Sistema operativo Solaris 2.5.1 (UNIX de SUN) . DBMS: Sybase v. 11

Red.

Protocolo de transmisión	Protocolo de control de acceso	Topología	Medio de transmisión físico
TCP/IP	CSMA/CD (Ethernet)	Estrella	Fibra óptica y par trenzado

5.5

Instalación del sistema cliente/servidor

Al momento de concluir el sistema, se tenía previamente instalada la red de comunicaciones, el servidor y las computadoras cliente. Por la parte de software, se contaba con el manejador de base de datos (Sybase versión 11) y el sistema operativo del servidor (Solaris 2.5.1, UNIX de SUN). En este punto se cubrió la instalación de la versión final de todos los componentes del sistema de inscripciones:

1.-Inscripción.
2.- Alumno.
3.- Profesor.
4.- Programa.
5.- Campus – Carrera.
6.- Grupo.
7.- Horario de atención.
8.- Estado de alumno.
9.- DGAE.
10.- Catálogos.

Finalmente, se realizó la etapa denominada puesta a punto, que consistió en configurar de acuerdo a las últimas pruebas de desempeño, las computadoras cliente y sus respectivas impresoras de alta velocidad.

Capítulo VI

Conclusiones

Conclusiones

Las soluciones de cómputo sustentadas en un ambiente cliente/servidor, pueden ser la mejor alternativa para muchas empresas mexicanas con demandas de importante procesamiento de información, exigencia de ambientes de trabajo productivos, flexibilidad para el crecimiento acorde a la dinámica del medio y limitantes de recursos económicos.

La metodología de desarrollo de aplicaciones para cliente/servidor utilizada en el presente trabajo funcionó de una manera muy eficiente. Sustentada en su parte de análisis y diseño del software por la metodología de Yourdon, y apoyada por una herramienta de diseño de software asistida por computadora, brindó la posibilidad de observar desde el inicio el contexto global del problema y ubicar las diversas opciones de solución del mismo.

Fue una peculiaridad muy enriquecedora, asumir como premisa el generar varias opciones de solución al problema por resolver, y cumplir las puntuales expectativas de los diferentes clientes del entorno.

Se generaron tres opciones, cada una con sus ventajas y desventajas de acuerdo a un cuadro de características objetivamente evaluables, después de haber hecho el análisis y diseño en su totalidad, se demostró que la opción de crear un nuevo sistema bajo cliente/servidor fue la mejor elección. Esto se concluye considerando que el generar un nuevo mantenimiento al sistema actual

nos seguiría ubicando en desventaja tecnológica para aprovechar las nuevas tendencias, y lo más importante, resolvería el problema operativamente mientras persistan las condiciones actuales. La opción de actualizar a Paradox 7.0 significaría en el mediano plazo algo equivalente a lo anterior, los problemas de fondo referentes a falta de flexibilidad para con el entorno y su poca compatibilidad con las nuevas creaciones de software y hardware, la colocaban en plena desventaja.

El sistema creado bajo cliente/servidor, gracias a su altamente flexible y parametrizado diseño de bases de datos, permite responder a los cambios del exterior respecto a políticas y condiciones en muy poco tiempo. Esta flexibilidad cubre inclusive variaciones en el mediano y largo plazo.

Como en todo sistema de información, en la medida en que sea parte del entorno, lo modificará positivamente y se generará un conjunto de cambios que los mismos usuarios del producto solicitarán.

La metodología de desarrollo de aplicaciones cliente/servidor planteada en este trabajo, puede aprovecharse en proyectos basados en cliente/servidor.

El sistema de inscripciones generado, resuelve el problema real de ingeniería que representa para la Facultad de Derecho asignar a miles de estudiantes las materias que solicitan, con las limitantes de cupo y con apego a los reglamentos de la Universidad.

Resultó en una gran experiencia haber utilizado herramientas tecnológicas de vanguardia como el sistema operativo Unix, el administrador de bases de datos Sybase y el lenguaje de programación PowerBuilder, interactuando sobre una amplia red de cómputo. Sin duda los conocimientos adquiridos en la Facultad, me brindaron la formación para aprender y explotar las herramientas citadas.

Los resultados principales a los que se llegaron son:

- Brindar un ambiente de trabajo amigable para los operadores del sistema en sus diferentes módulos, lo que los hace mucho más eficientes y los libera de la tensión que un medio poco agradable genera. Todo esto para beneficio de ellos mismos y por supuesto de los alumnos, profesores y el resto del personal administrativo.

- Puesto que el sistema está soportado por una verdadera base de datos, la integridad y seguridad de la información es muy alta.

- La mayoría del procesamiento se hace en el servidor, lo que aunado al diseño modularizado, se traduce en tiempos de respuesta pequeños, pudiendo así atender más rapidamente a los alumnos.

- No habrá el riesgo del sistema actual de perder comprobantes de inscripción y alterar la información previamente registrada.

- Se tienen parametrizados una gran cantidad de aspectos de alta movilidad, lo que asegura una pronta adaptación cuando las circunstancias demanden un cambio, en el mediano y largo plazo.

- El tráfico por la red disminuyó considerablemente, siendo uno de los factores que evitan caídas del sistema, previniendo con ello el desagrado de los alumnos.

- El costo de operación y mantenimiento del sistema disminuirá notablemente comparado con el anterior, puesto que el código del cliente se encuentra bien documentado y la base de datos, diseñada en un ambiente automatizado.

- Los cambios en línea que requiere la Secretaría de Asuntos Escolares se pueden llevar a cabo por los usuarios de la misma.

Respecto a las perspectivas del producto en el mediano y largo plazo, se puede comentar que la idea es tener en un mismo ambiente de base de datos toda la información de la facultad que así lo requiera, brindando la posibilidad de compartir toda la información necesaria.

Es por ello que existe la idea de crear una herramienta que funja como integrador de toda la administración de la información. Cuando ésta se defina con detalle, seguramente producirá cambios en el actual sistema de inscripciones.

La tecnología de cómputo es muy cambiante, y desde hoy existen ideas para operar la parte cliente con otros medios, por ejemplo internet. En este caso, la parte del servidor está preparada para ello y sólo se afectaría la interfaz al usuario y el medio de acceso, vía un navegador de internet. Así como ésta, pueden aparecer tecnologías que se tendrán que ir incorporando.

Bibliografía

[BER92] Berson Alex
Client/Server Architecture
McGraw-Hill, 1992

[BIB96] Biberdof Daryl
Power Builder 5 How-To
Waite Group Press, 1996

[BOO94] Booch Grady
Object Oriented Analisys and Design with Applications
Benjamin/Cummings Publishing, 1994

[BUT94] Butler Brian
The Great Leap Forward (SQL Databases)
PC Magazine Vol. 13 No. 17, 1994

[DAT87] Date C.J.
Bases de datos, una guía práctica
Addison-Wesley, 1987

[DAT81] Date C.J.
Introducción a los sistemas de bases de datos
Addison-Wesley, 1981

[DAW93] Dawna Travis
Client/Server Computing
McGraw-Hill, 1993

[DEI87] Deitel Harvey M.
Introducción a los sistemas operativos
Addison-Wesley, 1987

[DOU91] Douglas Comer
Internetworking with TCP/IP
Prentice Hall, 1991

[FRE93] Freedman, Alan
Diccionario de computación
Mc Graw Hill, 1993

[GAL96] Gallagher Simon
PowerBuilder 5 Unleashed
Sams Publishing, 1996

[HAZ96] Hazlehurst Peter
Using Sybase System XI
QUE, 1996

[MAH95] Mahler Paul
Desarrollo de aplicaciones cliente/servidor con PowerBuilder
Prentice Hall, 1995

[MAR89] Martin James
Local Area Networks
Prentice-Hall, 1989

[PRE93] Pressman Roger
Ingeniería del software
McGraw-Hill, 1993

[POW196] Powersoft
Connecting to your Database
Powersoft, 1996

[POW296] Powersoft
Getting Started
Powersoft, 1996

[POW396] Powersoft
Project Primer
Powersoft, 1996

[SAN93] Sander Ronald
Local Area Networking and LAN integration
Sander Group Inc., 1993

[SHA90] Shakuntala Atre
Técnicas de bases de datos
Trillas, 1990

[SHE95] Sheldom Tom
LAN Times Encyclopedia of Networking
McGraw-Hill, 1995

[SHL95] Sheldom Tom
LAN Times Guide to interoperability
McGraw-Hill, 1995

[SOM88] Sommerville Ian
Ingeniería de software
Addison-Wesley Iberoamericana, 1988

[TAN91] Tanenbaum Andrew S.
Redes de ordenadores
Prentice-Hall, 1991

[TAN92] Tanenbaum Andrew S.
Sistemas operativos modernos
Prentice-Hall, 1992

[VAU94] Vaughn Larry T.
Client/Server System Design and Implementation
McGraw-Hill, 1994

[YOU911] Yourdon Edward
Information Systems Analysis Workshop
Yourdon Press, 1991

[YOU89] Yourdon Edward
Modern Structured Analysis
Prentice-Hall, 1989

[YOU912] Yourdon Edward
Structured Systems and Program Design
Yourdon Press, 1991

Apéndices

Apéndice A

Contenido de las entrevistas

Resultados de las entrevistas.

Secretario de Servicios Escolares de la facultad.

1.- ¿ Cuál es la principal problemática que representa actualmente el proceso de inscripciones en la facultad ?

Considerando que el objetivo primordial de la administración escolar es apoyar la academia, son demasiados cinco días para realizar el proceso de inscripciones; lo deseable serían dos días. Procurando tener más días de clase, los mecanismos de lectura de calificaciones y de reinscripción tendrían que ser más ágiles, para ello las actas se deberían emitir y procesar localmente.

Los horarios se ofertan de acuerdo a la capacidad de profesores, no de alumnos.

Se tienen dos periodos de extraordinarios al final del semestre, un problema grave es que el trámite consume tiempo que podría dedicarse a más clases, en adición, los alumnos no tienen tiempo de estudiar y la estadística que se tiene del total de alumnos que presentan exámen extraordinario es: 60 % lo reprueban, del 40 % restante, el 30 % no se presenta, y del 10 %, 90 % sacan S de calificación y sólo 1 % B o MB. Por lo que lo deseable es tener varios periodos de inscripciones a lo largo del semestre.

2.- ¿ Qué aspectos conservaría del proceso y del sistema de inscripciones actual ?

La atención personalizada es buena. Se deben seguir vigilando las reglas que fija la propia universidad en cuanto a seriación, horarios, tiempo para concluir los ciclos educativos, etc.

Es rápido y confiable dado que el índice de error es del 1%.

3.- ¿ Qué características debiera tener el proceso y el sistema de inscripciones en la facultad ?

Evitar en lo posible la movilización de importantes cantidades de alumnos durante las inscripciones.

Leer las actas de calificaciones localmente, aunque implicara una inversión en un lector óptico.

Es necesario contemplar más reglamentos de la universidad, como los referentes a sanciones universitarias, que de no cumplirse conduzcan a la no inscripción del alumno.

4.- ¿ Qué características relevantes introduciría en el proceso de inscripciones de la facultad que posean en otras dependencias ?

Miembros de la secretaria estudiaron los procesos de otras dependencias y copiaron lo relevante y aplicable para nuestra facultad.

5.- ¿ Qué beneficios obtendría usted directamente de un nuevo sistema para el proceso de inscripciones en la facultad ?

Lograr que en la medida que el tiempo de inscripciones se reduzca, por lo tanto los días de clases se incrementarán en la misma proporción.

6.- ¿ Que características debiera contemplar un sistema de inscripciones para soportar el proceso en el mediano y largo plazo ?

Simplificar el proceso buscando ganar más días de clase. Una alternativa sería hacer las inscripciones por adelantado.

Tener un experto en cómputo permanentemente y al servicio de la secretaría, dado que el centro de cómputo no se da abasto.

7.- ¿ Cuál es el tiempo promedio de espera en filas ?

Para el 60 % de los alumnos el tiempo efectivo es de 1.5 minutos, aunque formados duran horas.

8.- ¿ Qué porcentaje de las solicitudes se satisface ?

El 60 % de los alumnos se inscriben en su primera opción.

Jefe del centro de cómputo de la facultad.

1.- ¿ Cuál es la principal problemática que representa actualmente el proceso de inscripciones en la facultad ?

El tiempo de respuesta del sistema es muy elevado, además de que el equipo completo se inhabilita de uno a diez minutos por día, posiblemente por el funcionamiento de las tarjetas ethernet. Por otro lado, aunque el tiempo de atención efectivo de los primeros bloques es de seis minutos, se tienen que formar cuatro horas antes de que les corresponda.

2.- ¿ Qué aspectos conservaría del proceso y del sistema de inscripciones actual ?

Que el proceso de inscripción siga siendo en línea, conservando para ello la asignación de días y horas de atención.

3.- ¿ Qué características debiera tener el proceso y el sistema de inscripciones en la facultad ?

No se debería depender de DGAE para obtener la información de las actas, sería útil un mecanismo de procesamiento propio, aunque posteriormente lo validara DGAE. Además, el sistema debiera ser más completo beneficiando con ello a las personas que tienen que dar la cara al alumno.

Sólo deben inscribirse los alumnos que no hayan cometido faltas a algún reglamento universitario, v.gr. alumnos suspendidos o deudores de material.

Debiera verificar en tiempo real la historia académica del alumno para evitar preprocesos.

4.- ¿ Qué características relevantes introduciría en el proceso de inscripciones de la facultad que posean en otras dependencias ?

No tengo punto de comparación.

5.- ¿ Qué beneficios obtendría usted directamente de un nuevo sistema para el proceso de inscripciones en la facultad ?

Mayor flexibilidad y rapidez, así como contar con un sistema amigable y completo.

6.- ¿ Que características debiera contemplar un sistema de inscripciones para soportar el proceso en el mediano y largo plazo ?

Estar soportado por una verdadera base de datos, y derivado del dinamismo de la tecnología de cómputo, el tiempo de vida que le estimo es de cinco años.

7.- ¿ Cuál es el tiempo promedio de espera en filas ?

Para la mayoría de los alumnos el tiempo en ventanillas es de diez minutos, pero tienen que llegar de dos a tres horas antes.

8.- ¿ Qué porcentaje de las solicitudes se satisface ?

Aproximadamente al 30% de los alumnos se les inscribe en su primera opción.

Alumno de tercer semestre.

1.- ¿ Cuál es la principal problemática que representa actualmente el proceso de inscripciones en la facultad ?

No respeta el bloque de atención, inscribir 300 alumnos en una hora son demasiados y el sistema se cae con una frecuencia de, por lo menos, una vez por día.

2.- ¿ Qué aspectos conservaría del proceso y del sistema de inscripciones actual ?

La inscripción en línea.

3.- ¿ Qué características debiera tener el proceso y el sistema de inscripciones en la facultad ?

Ampliar a dos semanas el periodo de inscripción y a hora y media el tiempo de atención del bloque. Otra medida sería repartir fichas para atención conforme se vaya llegando.

4.- ¿ Qué características relevantes introduciría en el proceso de inscripciones de la facultad que posean en otras dependencias ?

No conozco ningún otro proceso.

5.- ¿ Qué beneficios obtendría usted directamente de un nuevo sistema para el proceso de inscripciones en la facultad ?

Menos presión durante el periodo de inscripciones.

6.- ¿ Que características debiera contemplar un sistema de inscripciones para soportar el proceso en el mediano y largo plazo ?

Ser flexible a los cambios que se presenten en la legislación universitaria.

7.- ¿ Cuál es el tiempo promedio de espera en filas ?

Tres horas.

8.- ¿ Qué porcentaje de sus solicitudes se satisface ?

El 25 %, lo que representa dos de ocho materias en la primera opción.

Alumno de quinto semestre.

1.- ¿ Cuál es la principal problemática que representa actualmente el proceso de inscripciones en la facultad ?

Es lento y el sistema se satura, la tira de materias se traspapela con facilidad y la entregan de 20 a 60 minutos después de concluida la inscripción.

2.- ¿ Qué aspectos conservaría del proceso y del sistema de inscripciones actual ?

La inscripción en línea.

3.- ¿ Qué características debiera tener el proceso y el sistema de inscripciones en la facultad ?

Atender a 100 personas por hora y mantener el criterio de bloque de atención por promedio y créditos.

4.- ¿ Qué características relevantes introduciría en el proceso de inscripciones de la facultad que posean en otras dependencias ?

No conozco otro.

5.- ¿ Qué beneficios obtendría usted directamente de un nuevo sistema para el proceso de inscripciones en la facultad ?

Una mejor asignación de profesores.

6.- ¿ Que características debiera contemplar un sistema de inscripciones para soportar el proceso en el mediano y largo plazo ?

Crecer con la matricula.

7.- ¿ Cuál es el tiempo promedio de espera en filas ?

Tres horas.

8.- ¿ Qué porcentaje de sus solicitudes se satisface ?

En mi caso el 100%.

Alumno de séptimo semestre.

1.- ¿ Cuál es la principal problemática que representa actualmente el proceso de inscripciones en la facultad ?

El 20 % de los profesores no entregan actas a tiempo.

2.- ¿ Qué aspectos conservaría del proceso y del sistema de inscripciones actual ?

El criterio de asignación de bloques basado en el promedio y en los créditos.

3.- ¿ Qué características debiera tener el proceso y el sistema de inscripciones en la facultad ?

No tardar en entregar las tiras de materias más de 30 minutos, y tener la opción de inscripción individual y por bloque de profesores.

4.- ¿ Qué características relevantes introduciría en el proceso de inscripciones de la facultad que posean en otras dependencias ?

No conozco ninguno.

5.- ¿ Qué beneficios obtendría usted directamente de un nuevo sistema para el proceso de inscripciones en la facultad ?

Que me asignen a los profesores que quiero.

6.- ¿ Que características debiera contemplar un sistema de inscripciones para soportar el proceso en el mediano y largo plazo ?

Considerar que cada año ingresan más alumnos a la facultad.

7.- ¿ Cuál es el tiempo promedio de espera en filas ?

Tres horas.

8.- ¿ Qué porcentaje de sus solicitudes se satisface ?

El 100 %.

Usuario del sistema de inscripciones actual.

1.- ¿ Cuál es la principal problemática que representa actualmente el proceso de inscripciones en la facultad ?

Se tiene que atender a demasiados alumnos para el tamaño de los bloques de atención, además el sistema se cae con frecuencia.

2.- ¿ Qué aspectos conservaría del proceso y del sistema de inscripciones actual ?

Que los alumnos puedan saber en qué grupos quedaron al momento.

3.- ¿ Qué características debiera tener el proceso y el sistema de inscripciones en la facultad ?

Ser ágil y confiable. Se debe idear un medio para que los alumnos no se tengan que formar varias horas antes de que les corresponda a su bloque de atención.

4.- ¿ Qué características relevantes introduciría en el proceso de inscripciones de la facultad que posean en otras dependencias ?

No conozco otro sistema de inscripciones.

5.- ¿ Qué beneficios obtendría usted directamente de un nuevo sistema para el proceso de inscripciones en la facultad ?

Que se me asignen a los profesores que solicito como primera opción y no tener que esperar tanto tiempo para inscribirme.

6.- ¿ Que características debiera contemplar un sistema de inscripciones para soportar el proceso en el mediano y largo plazo ?

Que los mecanismos de atención se vayan ajustando según el tamaño de la facultad.

7.- ¿ Cuál es el tiempo promedio de espera en filas ?

De dos a tres horas.

8.- ¿ Qué porcentaje de las solicitudes se satisface ?

El 100 %.

Personal que le da soporte al sistema de inscripciones actual.

1.- ¿ Cuál es la principal problemática que representa actualmente el proceso de inscripciones en la facultad ?

La carencia de información actualizada de cupos en grupos, que ocasiona que en un número muy importante de casos el alumno tenga que rehacer su horario en la misma ventanilla, con los cuellos de botella que esto implica.

Por otro lado, se pierden demasiadas tiras de materias al momento de mandarlas imprimir.

2.- ¿ Qué aspectos conservaría del proceso y del sistema de inscripciones actual ?

La inscripción en línea, tomando como parámetros objetivos el armado de bloques de atención, el promedio y el avance en créditos. Además los tiempos de respuesta son aceptables.

3.- ¿ Qué características debiera tener el proceso y el sistema de inscripciones en la facultad ?

Que al momento de llegar a las ventanillas el sistema pueda sugerir, dependiendo de los datos del alumno, un bloque completo de materias que tengan horario disponible.

Así mismo, es muy deseable que el proceso de impresión de tiras de materias sea más eficiente y el tiempo de espera sea uniforme.

4.- ¿ Qué características relevantes introduciría en el proceso de inscripciones de la facultad que posean en otras dependencias ?

Analizar cómo se podría instrumentar las inscripciones fuera de línea.

5.- ¿ Qué beneficios obtendría usted directamente de un nuevo sistema para el proceso de inscripciones en la facultad ?

Que los alumnos tendrían una inscripción más rápida y sencilla. Por la parte de soporte al sistema se lograría una mayor facilidad de uso y mantenimiento.

6.- ¿ Que características debiera contemplar un sistema de inscripciones para soportar el proceso en el mediano y largo plazo ?

Permitir adaptarse con relativa facilidad a un cambio de plan de estudios o de reglamento de inscripciones.

7.- ¿ Cuál es el tiempo promedio de espera en filas ?

De una a dos horas.

8.- ¿ Qué porcentaje de las solicitudes se satisface ?

El 40 % en la primera vuelta.

Apéndice B

Código de la generación de la base de datos

```
CREATE RULE co_calificacion
        AS minima<=maxima

CREATE RULE co_hora
        AS inicio<termino

CREATE RULE ru_calificacion
        AS @calificacion between 0 and 10

CREATE RULE ru_fanterior
        AS @fecha < getdate()

CREATE RULE ru_no_negativo
        AS @numero >= 0

CREATE RULE ru_rfc
        AS (@rfc like "[A-Z][AEIOU][A-Z][A-Z][0-9][0-9][0-1][0-
9][0-3][0-9]%") and (convert (tinyint,substring(@rfc,7,2))
between 1 and 12) and (convert (tinyint,substring(@rfc,9,2))
between 1 and 31)

CREATE DEFAULT de_hoy
        AS getdate()

exec sp_addtype dt_apellido, "varchar(20)", "NOT NULL"

exec sp_addtype dt_asignatura, "numeric(4)", "NOT NULL"

exec sp_addtype dt_calificacion, "numeric(4,2)", "NULL"
exec sp_bindrule ru_calificacion, dt_calificacion

exec sp_addtype dt_calle, "varchar(40)", "NULL"

exec sp_addtype dt_codi_postal, "numeric(5)", "NULL"

exec sp_addtype dt_colonia, "varchar(30)", "NULL"

exec sp_addtype dt_creditos, "smallint", "NULL"
exec sp_bindrule ru_no_negativo, dt_creditos

exec sp_addtype dt_descripcion, "text", "NULL"

exec sp_addtype dt_fecha, "smalldatetime", "NULL"

exec sp_addtype dt_nombre, "varchar(25)", "NOT NULL"

exec sp_addtype dt_num_cuenta, "numeric(8)", "NOT NULL"

exec sp_addtype dt_rfc, "char(13)", "NOT NULL"
exec sp_bindrule ru_rfc, dt_rfc

exec sp_addtype dt_telefono, "numeric(12)", "NULL"

CREATE TABLE acta (
    folio           numeric(7) NOT NULL,
    grupo           numeric(16) NOT NULL,
    emision         dt_fecha,
    complementaria  bit,
 CONSTRAINT in_acta_principal    PRIMARY KEY (folio)
)

exec sp_primarykey acta,
    folio

exec sp_bindefault de_hoy, 'acta.emision'
grant all on acta to desarrollo
grant select on acta to consulta


CREATE TABLE alumno (
    cuenta          dt_num_cuenta,
    edo_civil       numeric(4) NULL,
    bachillerato    numeric(4) NULL,
    municipio       numeric(8) NULL,
    paterno         dt_apellido NULL,
    materno         dt_apellido NULL,
    nombre          dt_nombre,
    nombre_corto    varchar(32) NOT NULL,
    sexo            bit,
    nacimiento      dt_fecha,
    calle           dt_calle,
    colonia         dt_colonia,
    cp              dt_codi_postal,
    telefono        dt_telefono,
    extranjero      bit,
    bach_ext        bit,
    nip             numeric(6) NULL,
    email           varchar(30) NULL,
    prom_bach       dt_calificacion,
    folio           char(5) NULL,
    activo          bit DEFAULT "true",
 CONSTRAINT in_alumno_principal    PRIMARY KEY
(cuenta),
    CONSTRAINT co_apellidos
        CHECK (datalength(isnull(paterno,
"")+isnull(materno,""))>0)
)

CREATE INDEX in_alumno_nombre ON alumno
(
    paterno,
    materno,
    nombre
)

exec sp_primarykey alumno,
    cuenta

exec sp_bindrule ru_fanterior, 'alumno.nacimiento'
grant all on alumno to desarrollo
grant select on alumno to consulta


CREATE TABLE alumno_asignatura (
    cuenta          dt_num_cuenta,
    programa        numeric(8) NOT NULL,
    asignatura      numeric(8) NOT NULL,
    antecedentes    tinyint NULL,
 CONSTRAINT in_alumno_asig_principal    PRIMARY KEY
(cuenta, programa, asignatura)
)

exec sp_primarykey alumno_asignatura,
    cuenta,
    programa,
    asignatura

exec sp_bindrule ru_no_negativo,
'alumno_asignatura.antecedentes'
grant all on alumno_asignatura to desarrollo
grant select on alumno_asignatura to consulta


CREATE TABLE alumno_asignatura_modalidad (
    cuenta          dt_num_cuenta,
    programa        numeric(8) NOT NULL,
    modalidad       numeric(4) NOT NULL,
    asignatura      numeric(8) NOT NULL,
    inscripciones   tinyint NULL,
 CONSTRAINT in_alum_asig_mod_principal    PRIMARY
KEY (cuenta, programa, modalidad, asignatura)
)
```

```
exec sp_primarykey alumno_asignatura_modalidad,
    cuenta,
    programa,
    modalidad,
    asignatura

exec sp_bindrule ru_no_negativo,
'alumno_asignatura_modalidad.inscripciones'
grant all on alumno_asignatura_modalidad to desarrollo
grant select on alumno_asignatura_modalidad to consulta


CREATE TABLE alumno_bloque (
    cuenta          dt_num_cuenta,
    programa        numeric(8) NOT NULL,
    bloque          numeric(8) NOT NULL,
    avance          dt_creditos,
 CONSTRAINT in_alumno_bloque_principal    PRIMARY
KEY (cuenta, programa, bloque)
)

exec sp_primarykey alumno_bloque,
    cuenta,
    programa,
    bloque
grant all on alumno_bloque to desarrollo
grant select on alumno_bloque to consulta


CREATE TABLE alumno_horario (
    cuenta          dt_num_cuenta,
    programa        numeric(8) NOT NULL,
    horario         numeric(8) NOT NULL,
    tiempo          dt_fecha,
    movimientos     tinyint NULL,
    atenciones      tinyint NULL,
 CONSTRAINT in_alumno_horario_principal    PRIMARY
KEY (cuenta, programa, horario)
)

exec sp_primarykey alumno_horario,
    cuenta,
    programa,
    horario

exec sp_bindrule ru_no_negativo,
'alumno_horario.movimientos'
exec sp_bindrule ru_no_negativo, 'alumno_horario.atenciones'
grant all on alumno_horario to desarrollo
grant select on alumno_horario to consulta


CREATE TABLE alumno_plan (
    cuenta          dt_num_cuenta,
    programa        numeric(8) NOT NULL,
    inicio          numeric(8) NOT NULL,
    termino         numeric(8) NULL,
    ingreso         numeric(4) NULL,
    estado          numeric(4) NULL,
    turno           numeric(4) NULL,
    obligatorio     dt_creditos,
    optativo        dt_creditos,
    asignaturas     tinyint NULL,
 CONSTRAINT in_alumno_plan_principal    PRIMARY KEY
(cuenta, programa)
)

exec sp_primarykey alumno_plan,
    cuenta,
    programa
exec sp_bindrule ru_no_negativo, 'alumno_plan.asignaturas'
grant all on alumno_plan to desarrollo
grant select on alumno_plan to consulta


CREATE TABLE asignatura (
    clave           numeric(8) IDENTITY,
    clave_corta     numeric(4) NULL,
    nombre          varchar(50) NOT NULL,
    nombre_corto    varchar(28) NOT NULL,
    descripcion     dt_descripcion,
    temario         dt_descripcion,
    activo          bit DEFAULT "true",
 CONSTRAINT in_asignatura_principal    PRIMARY KEY
(clave),
    CONSTRAINT in_asignatura_nombre
    UNIQUE (
        nombre
    ),
    CONSTRAINT in_asignatura_nombre_corto
    UNIQUE (
        nombre_corto
    )
)

exec sp_primarykey asignatura,
    clave
grant all on asignatura to desarrollo
grant select on asignatura to consulta


CREATE TABLE asignatura_bloque (
    bloque          numeric(8) NOT NULL,
    asignatura      numeric(8) NOT NULL,
 CONSTRAINT in_asignatura_bloque_principal    PRIMARY
KEY (bloque, asignatura)
)

exec sp_primarykey asignatura_bloque,
    bloque,
    asignatura
grant all on asignatura_bloque to desarrollo
grant select on asignatura_bloque to consulta


CREATE TABLE asignatura_plan (
    clave           numeric(8) IDENTITY,
    asignatura      numeric(8) NOT NULL,
    programa        numeric(8) NOT NULL,
    tipo            numeric(4) NOT NULL,
    creditos        dt_creditos,
    periodo         tinyint NULL,
    antecedentes    tinyint NULL,
    acreditar       bit,
    optativa        tinyint NULL,
    activo          bit DEFAULT "true",
 CONSTRAINT in_asignatura_plan_principal    PRIMARY
KEY (clave)
)

exec sp_primarykey asignatura_plan,
    clave

exec sp_bindrule ru_no_negativo,
'asignatura_plan.antecedentes'
grant all on asignatura_plan to desarrollo
grant select on asignatura_plan to consulta


CREATE TABLE aula (
```

```
    clave           numeric(8) IDENTITY,
    nombre          dt_nombre,
    descripcion     dt_descripcion,
    capacidad       tinyint NULL,
    ubicacion       dt_descripcion,
    activo          bit DEFAULT "true",
CONSTRAINT in_aula_principal    PRIMARY KEY (clave),
    CONSTRAINT in_aula_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey aula,
    clave
grant all on aula to desarrollo
grant select on aula to consulta


CREATE TABLE bachillerato (
    clave           numeric(4) IDENTITY,
    nombre          dt_nombre,
    activo          bit DEFAULT "true",
 CONSTRAINT in_bachillerato_principal    PRIMARY KEY
(clave),
    CONSTRAINT in_bachillerato_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey bachillerato,
    clave
grant all on bachillerato to desarrollo
grant select on bachillerato to consulta


CREATE TABLE bloque (
    clave           numeric(8) IDENTITY,
    programa        numeric(8) NOT NULL,
    nombre          dt_nombre,
    descripcion     dt_descripcion,
    creditos        dt_creditos,
 CONSTRAINT in_bloque_principal    PRIMARY KEY (clave),
    CONSTRAINT in_bloque_nombre_plan
    UNIQUE (
        nombre,
        programa
    )
)

exec sp_primarykey bloque,
    clave
grant all on bloque to desarrollo
grant select on bloque to consulta


CREATE TABLE calificacion (
    clave           numeric(4) IDENTITY,
    representacion  char(2) NOT NULL,
    valor           dt_calificacion,
    aprobatoria     bit,
    activo          bit DEFAULT "true",
 CONSTRAINT in_calificacion_principal    PRIMARY KEY
(clave)
)

CREATE INDEX in_calificacion_representacion ON calificacion
(
    representacion
)

CREATE INDEX in_calificacion_valor ON calificacion
(
    valor
)

exec sp_primarykey calificacion,
    clave
grant all on calificacion to desarrollo
grant select on calificacion to consulta


CREATE TABLE campus (
    clave           numeric(4) IDENTITY,
    municipio       numeric(8) NULL,
    nombre          dt_nombre,
    nom_corto       numeric(3) DEFAULT 0 NOT NULL,
    calle           dt_calle,
    colonia         dt_colonia,
    cp              dt_codi_postal,
    telefono        dt_telefono,
    descripcion     dt_descripcion,
    activo          bit DEFAULT "true",
 CONSTRAINT in_campus_principal    PRIMARY KEY
(clave),
    CONSTRAINT in_campus_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey campus,
    clave

exec sp_bindrule ru_no_negativo, 'campus.nom_corto'
grant all on campus to desarrollo
grant select on campus to consulta


CREATE TABLE campus_carrera (
    campus          numeric(4) NOT NULL,
    carrera         numeric(4) NOT NULL,
 CONSTRAINT in_campus_carrera_principal    PRIMARY
KEY (campus, carrera)
)

exec sp_primarykey campus_carrera,
    campus,
    carrera
grant all on campus_carrera to desarrollo
grant select on campus_carrera to consulta


CREATE TABLE carrera (
    clave           numeric(4) IDENTITY,
    nombre          varchar(36) NOT NULL,
    nombre_corto    numeric(2) DEFAULT 0 NOT NULL,
    descripcion     dt_descripcion,
    activo          bit DEFAULT "true",
 CONSTRAINT in_carrera_principal    PRIMARY KEY (clave),
    CONSTRAINT in_carrera_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey carrera,
    clave

exec sp_bindrule ru_no_negativo, 'carrera.nombre_corto'
grant all on carrera to desarrollo
```

```
grant select on carrera to consulta


 CREATE TABLE cateria (
      clave              numeric(4) IDENTITY,
      nombre               dt_nombre,
      activo             bit DEFAULT "true",
  CONSTRAINT in_cateria_principal      PRIMARY KEY (clave),
      CONSTRAINT in_cateria_nombre
      UNIQUE (
           nombre
      )
 )

 exec sp_primarykey cateria,
      clave
 grant all on catena to desarrollo
grant select on cateria to consulta


 CREATE TABLE civil (
      clave              numeric(4) IDENTITY,
      nombre               dt_nombre,
  CONSTRAINT in_civil_principal      PRIMARY KEY
NONCLUSTERED (clave),
      CONSTRAINT in_civil_nombre
      UNIQUE CLUSTERED (
           nombre
      )
 )

 exec sp_primarykey civil,
      clave
 grant all on civil to desarrollo
grant select on civil to consulta


 CREATE TABLE dgae_abc_grupo (
      plantel            numeric(3) NULL,
      asignatura           numeric(4) NULL,
      grupo              char(4) NULL,
      cupo               numeric(4) NULL,
      profesor             numeric(1) NULL,
      nombre_profesor      char(32) NULL,
      rfc              char(13) NULL,
      movimiento           char(2) NULL,
      filler             char(3) NULL
 )

 grant all on dgae_abc_grupo to desarrollo
grant select on dgae_abc_grupo to consulta


 CREATE TABLE dgae_abc_ha (
      plantel            numeric(3) NULL,
      num_cuenta           numeric(8) NULL,
      asignatura           numeric(4) NULL,
      periodo            numeric(3) NULL,
      tipo               char(1) NULL,
      calificacion         char(2) NULL,
      folio              numeric(7) NULL,
      grupo              char(4) NULL,
      movimiento           char(2) NULL
 )

 grant all on dgae_abc_ha to desarrollo
grant select on dgae_abc_ha to consulta


 CREATE TABLE dgae_abc_inscripcion (
      num_cuenta           numeric(8) NULL,
      plantel            numeric(3) NULL,
      asignatura           numeric(4) NULL,
      grupo_baja           char(4) NULL,
      grupo_alta           char(4) NULL,
      filler             char(1) NULL
 )

 grant all on dgae_abc_inscripcion to desarrollo
grant select on dgae_abc_inscripcion to consulta


 CREATE TABLE dgae_actas_emitidas (
      folio              numeric(7) NULL,
      plantel            numeric(3) NULL,
      Asignatura           numeric(4) NULL,
      examen             numeric(1) NULL,
      sem_escolar          numeric(1) NULL,
      año_escolar          numeric(2) NULL,
      actas              numeric(2) NULL,
      alumnos              numeric(4) NULL,
      fecha_emision        numeric(6) NULL,
      grupo              char(4) NULL,
      num_cuenta           numeric(8) NULL,
      calificacion         numeric(1) NULL,
      acta               char(1) NULL,
      filler             char(4) NULL
 )

 grant all on dgae_actas_emitidas to desarrollo
grant select on dgae_actas_emitidas to consulta


 CREATE TABLE dgae_alumnos (
      alumno             char(32) NULL,
      num_cuenta           numeric(8) NULL,
      plantel            numeric(3) NULL,
      carrera            numeric(2) NULL,
      primer_ingreso       numeric(2) NULL,
      nacionalidad         numeric(1) NULL,
      ingreso            numeric(2) NULL,
      exalumno             numeric(2) NULL,
      sexo               char(1) NULL,
      nacimiento           numeric(6) NULL,
      movimiento           numeric(6) NULL,
      inscrito           numeric(1) NULL
 )

 grant all on dgae_alumnos to desarrollo
grant select on dgae_alumnos to consulta


 CREATE TABLE dgae_asignaturas (
      asignatura           char(28) NULL,
      plantel            numeric(3) NULL,
      clave              numeric(4) NULL,
      creditos             numeric(2) NULL,
      semestre             numeric(2) NULL,
      nivel              char(1) NULL,
      filler             char(2) NULL
 )

 grant all on dgae_asignaturas to desarrollo
grant select on dgae_asignaturas to consulta


 CREATE TABLE dgae_b_alumno (
      plantel            numeric(3) NULL,
      num_cuenta           numeric(8) NULL,
      carrera            numeric(2) NULL,
      baja               char(2) NULL
 )
```

```
 grant all on dgae_b_alumno to desarrollo
grant select on dgae_b_alumno to consulta


 CREATE TABLE dgae_b_inscripcion (
      plantel          numeric(3) NULL,
      carrera          numeric(2) NULL,
      num_cuenta       numeric(8) NULL,
      filler           char(2) NULL
)

 grant all on dgae_b_inscripcion to desarrollo
grant select on dgae_b_inscripcion to consulta


 CREATE TABLE dgae_c_carrera (
      plantel          numeric(3) NULL,
      num_cuenta       numeric(8) NULL,
      origen           numeric(2) NULL,
      destino          numeric(2) NULL
)

 grant all on dgae_c_carrera to desarrollo
grant select on dgae_c_carrera to consulta


 CREATE TABLE dgae_c_unidad_academica (
      num_cuenta       numeric(8) NULL,
      plantel_origen   numeric(3) NULL,
      carrera_origen   numeric(2) NULL,
      plantel_destino  numeric(3) NULL,
      carrera_destino  numeric(2) NULL
)

 grant all on dgae_c_unidad_academica to desarrollo
grant select on dgae_c_unidad_academica to consulta


 CREATE TABLE dgae_carreras (
      plantel          numeric(3) NULL,
      carrera          numeric(2) NULL,
      nombre           char(36) NULL,
      nivel            char(1) NULL,
      programa         numeric(1) NULL,
      duracion         numeric(2) NULL,
      obligatorios     numeric(3) NULL,
      optativos        numeric(3) NULL,
      filler           char(5) NULL
)

 grant all on dgae_carreras to desarrollo
grant select on dgae_carreras to consulta


 CREATE TABLE dgae_ha (
      num_cuenta       numeric(8) NULL,
      plantel          numeric(3) NULL,
      asignatura       numeric(4) NULL,
      año_semestre     numeric(3) NULL,
      calificacion     char(2) NULL,
      grupo            char(4) NULL,
      acta             numeric(7) NULL,
      examen           char(2) NULL
)

 grant all on dgae_ha to desarrollo
grant select on dgae_ha to consulta


 CREATE TABLE dgae_inscripcion_e (
      Numero_de_Cuenta    tinyint NULL,
      Clave_de_Plantel    tinyint NULL,
      Clave_de_Asignatura tinyint NULL,
      Clave_de_Grupo      tinyint NULL,
      filler              char(5) NULL
)

 grant all on dgae_inscripcion_e to desarrollo
grant select on dgae_inscripcion_e to consulta


 CREATE TABLE dgae_inscripcion_s (
      num_cuenta       numeric(8) NULL,
      plantel          numeric(3) NULL,
      asignatura       numeric(4) NULL,
      grupo            char(4) NULL,
      filler           char(5) NULL
)

 grant all on dgae_inscripcion_s to desarrollo
grant select on dgae_inscripcion_s to consulta


 CREATE TABLE dgae_r_ha (
      num_cuenta       numeric(8) NULL,
      MB               tinyint NULL,
      B                tinyint NULL,
      S                tinyint NULL,
      n_6              tinyint NULL,
      n_7              tinyint NULL,
      n_8              tinyint NULL,
      n_9              tinyint NULL,
      n_10             tinyint NULL,
      NA               tinyint NULL,
      NP               tinyint NULL,
      Revalidadas      tinyint NULL,
      Acreditadas      tinyint NULL,
      Aprobadas_en_Ordinario tinyint NULL,
      Aprobadas_en_Extraordinario tinyint NULL,
      Reprobadas_en_Ordinario tinyint NULL,
      Reprobadas_en_Extraordinario tinyint NULL,
      obligatorios_acumulados numeric(3) NULL,
      optativos_acumulados numeric(3) NULL,
      periodo_inicial  numeric(3) NULL,
      periodo_ultimo   numeric(3) NULL,
      Covalidadas      tinyint NULL
)

 grant all on dgae_r_ha to desarrollo
grant select on dgae_r_ha to consulta


 CREATE TABLE estado (
      clave            numeric(4) IDENTITY,
      nombre           dt_nombre,
      abreviacion      char(4) NOT NULL,
 CONSTRAINT in_estado_principal    PRIMARY KEY (clave),
      CONSTRAINT in_estado_nombre
      UNIQUE (
           nombre
      ),
      CONSTRAINT in_estado_abreviacion
      UNIQUE (
           abreviacion
      )
)

 exec sp_primarykey estado,
      clave
 grant all on estado to desarrollo
grant select on estado to consulta
```

```
CREATE TABLE estado_alumno (
    clave           numeric(4) IDENTITY,
    nombre          dt_nombre,
    activo          bit DEFAULT "true",
 CONSTRAINT in_estado_alumno_principal    PRIMARY
KEY (clave),
    CONSTRAINT in_estado_alumno_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey estado_alumno,
    clave
grant all on estado_alumno to desarrollo
grant select on estado_alumno to consulta


CREATE TABLE estado_grupo (
    clave           numeric(4) IDENTITY,
    nombre          dt_nombre,
    activo          bit DEFAULT "true",
 CONSTRAINT in_estado_grupo_principal    PRIMARY KEY
(clave),
    CONSTRAINT in_estado_grupo_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey estado_grupo,
    clave
grant all on estado_grupo to desarrollo
grant select on estado_grupo to consulta


CREATE TABLE estado_inscripcion (
    clave           numeric(4) IDENTITY,
    nombre          dt_nombre,
    activo          bit DEFAULT "true",
 CONSTRAINT in_edo_inscripcion_principal    PRIMARY
KEY (clave),
    CONSTRAINT in_estado_inscripcion_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey estado_inscripcion,
    clave
grant all on estado_inscripcion to desarrollo
grant select on estado_inscripcion to consulta


CREATE TABLE grado (
    clave           numeric(4) IDENTITY,
    nombre          dt_nombre,
    descripcion     dt_descripcion,
 CONSTRAINT in_grado_principal    PRIMARY KEY
NONCLUSTERED (clave),
    CONSTRAINT in_grado_nombre
    UNIQUE CLUSTERED (
        nombre
    )
)

exec sp_primarykey grado,
    clave
grant all on grado to desarrollo
grant select on grado to consulta


CREATE TABLE grupo (
    clave           numeric(16) IDENTITY,
    asignatura      numeric(8) NOT NULL,
    periodo         numeric(8) NOT NULL,
    modalidad       numeric(4) NOT NULL,
    nom_corto       char(4) NOT NULL,
    estado          numeric(4) NULL,
    turno           numeric(4) NULL,
    cupo            tinyint NULL,
    ocupados        tinyint NULL,
 CONSTRAINT in_grupo_principal    PRIMARY KEY (clave),
    CONSTRAINT co_cupo
        CHECK (ocupados<=cupo)
)

exec sp_primarykey grupo,
    clave
grant all on grupo to desarrollo
grant select on grupo to consulta


CREATE TABLE horario (
    grupo           numeric(16) NOT NULL,
    aula            numeric(8) NOT NULL,
    lunes           bit,
    martes          bit,
    miercoles       bit,
    jueves          bit,
    viernes         bit,
    sabado          bit,
    inicio          dt_fecha,
    termino         dt_fecha,
 CONSTRAINT in_horario_principal    PRIMARY KEY (grupo,
aula)
)

exec sp_primarykey horario,
    grupo,
    aula
grant all on horario to desarrollo
grant select on horario to consulta


CREATE TABLE horario_atencion (
    clave           numeric(8) IDENTITY,
    modalidad       numeric(4) NOT NULL,
    periodo         numeric(8) NOT NULL,
    turno           numeric(4) NULL,
    inicio          dt_fecha,
    termino         dt_fecha,
    tiempo          dt_fecha,
    movimientos     tinyint NULL,
    atenciones      tinyint NULL,
    altas           bit DEFAULT "true",
    bajas           bit DEFAULT "true",
    cambios         bit DEFAULT "true",
    consultas       bit DEFAULT "true",
    activo          bit DEFAULT "true",
 CONSTRAINT in_horario_principal    PRIMARY KEY
NONCLUSTERED (clave)
)

exec sp_primarykey horario_atencion,
    clave

exec sp_bindrule ru_no_negativo,
'horario_atencion.movimientos'
```

```
exec sp_bindrule ru_no_negativo,
'horario_atencion.atenciones'
grant all on horario_atencion to desarrollo
grant select on horario_atencion to consulta


CREATE TABLE ingreso (
    clave           numeric(4) IDENTITY,
    nombre          dt_nombre,
    activo          bit DEFAULT "true",
 CONSTRAINT in_ingreso_principal    PRIMARY KEY
(clave),
    CONSTRAINT in_ingreso_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey ingreso,
    clave
grant all on ingreso to desarrollo
grant select on ingreso to consulta


CREATE TABLE inscripcion (
    programa        numeric(8) NOT NULL,
    cuenta          dt_num_cuenta,
    grupo           numeric(16) NOT NULL,
    estado          numeric(4) NULL,
    calificacion    numeric(4) NULL,
    fecha           dt_fecha,
 CONSTRAINT in_inscripcion_principal    PRIMARY KEY
(programa, cuenta, grupo)
)

exec sp_primarykey inscripcion,
    programa,
    cuenta,
    grupo

exec sp_bindefault de_hoy, 'inscripcion.fecha'
grant all on inscripcion to desarrollo
grant select on inscripcion to consulta


CREATE TABLE modalidad (
    modalidad       numeric(4) IDENTITY,
    nombre          dt_nombre,
    nombre_corto    char(2) DEFAULT "  " NOT NULL,
    inscripciones   tinyint NULL,
    maxima          dt_calificacion,
    minima          dt_calificacion,
    activo          bit DEFAULT "true",
 CONSTRAINT in_modalidad_principal    PRIMARY KEY
(modalidad),
    CONSTRAINT in_modalidad_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey modalidad,
    modalidad

exec sp_bindrule ru_no_negativo, 'modalidad.inscripciones'
grant all on modalidad to desarrollo
grant select on modalidad to consulta


CREATE TABLE modalidad_tipo (
    tipo            numeric(4) NOT NULL,
    modalidad       numeric(4) NOT NULL,
    inscripciones   tinyint NULL,
    maxima          dt_calificacion,
    minima          dt_calificacion,
 CONSTRAINT in_modalidad_tipo_principal    PRIMARY
KEY (tipo, modalidad)
)

exec sp_primarykey modalidad_tipo,
    tipo,
    modalidad

exec sp_bindrule ru_no_negativo,
'modalidad_tipo.inscripciones'
grant all on modalidad_tipo to desarrollo
grant select on modalidad_tipo to consulta


CREATE TABLE municipio (
    clave           numeric(8) IDENTITY,
    nombre          dt_nombre,
    estado          numeric(4) NOT NULL,
 CONSTRAINT in_municipio_principal    PRIMARY KEY
(clave)
)

CREATE INDEX in_municipio_nombre ON municipio
(
    nombre
)

exec sp_primarykey municipio,
    clave
grant all on municipio to desarrollo
grant select on municipio to consulta


CREATE TABLE nivel (
    clave           numeric(4) IDENTITY,
    nombre          dt_nombre,
    descripcion     dt_descripcion,
    activo          bit DEFAULT "true",
 CONSTRAINT in_nivel_principal    PRIMARY KEY (clave),
    CONSTRAINT in_nivel_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey nivel,
    clave
grant all on nivel to desarrollo
grant select on nivel to consulta


CREATE TABLE obs_prof (
    clave           numeric(4) IDENTITY,
    profesor        numeric(8) NOT NULL,
    descripcion     varchar(250) NULL,
    fecha           dt_fecha,
    autor           varchar(50) NULL,
    titulo          varchar(50) NOT NULL,
 CONSTRAINT in_obs_prof_principal    PRIMARY KEY
(clave)
)

exec sp_primarykey obs_prof,
    clave
grant all on obs_prof to desarrollo
grant select on obs_prof to consulta
```

```
CREATE TABLE observacion (
    clave           numeric(16) IDENTITY,
    cuenta          dt_num_cuenta,
    titulo          varchar(50) NOT NULL,
    autor           varchar(50) NULL,
    fecha           dt_fecha,
    cateria         numeric(4) NOT NULL,
    descripcion     varchar(255) NULL,
 CONSTRAINT in_observacion_principal    PRIMARY KEY
NONCLUSTERED (clave)
)

exec sp_primarykey observacion,
    clave

exec sp_bindefault de_hoy, 'observacion.fecha'
grant all on observacion to desarrollo
grant select on observacion to consulta


CREATE TABLE parametros (
    clave           numeric(4) NOT NULL,
    nombre          dt_nombre,
    booleano        bit,
    numerico        int NULL,
    cadena          varchar(255) NULL,
    activo          bit DEFAULT "true",
 CONSTRAINT XPKparametros    PRIMARY KEY (clave)
)

exec sp_primarykey parametros,
    clave
grant all on parametros to desarrollo
grant select on parametros to consulta


CREATE TABLE periodo (
    clave           numeric(8) IDENTITY,
    nombre          dt_nombre,
    nombre_corto    numeric(5) NOT NULL,
    inicio          dt_fecha,
    termino         dt_fecha,
    activo          bit DEFAULT "true",
 CONSTRAINT in_periodo_principal    PRIMARY KEY
(clave),
    CONSTRAINT in_periodo_nombre
    UNIQUE (
        nombre
    )
)

exec sp_primarykey periodo,
    clave
grant all on periodo to desarrollo
grant select on periodo to consulta


CREATE TABLE profesor (
    clave           numeric(8) NOT NULL,
    rfc             dt_rfc,
    grado           numeric(4) NULL,
    edo_civil       numeric(4) NULL,
    tipo            numeric(4) NULL,
    trab_municipio  numeric(8) NULL,
    nombre          dt_nombre,
    paterno         dt_apellido NULL,
    materno         dt_apellido NULL,
    nombre_corto    varchar(32) NOT NULL,
    extranjero      b t,
    sexo            bit,
    nacimiento      dt_fecha,
    cedula          numeric(9) NULL,
    descripcion     dt_descripcion,
    calle           dt_calle,
    colonia         dt_colonia,
    cp              dt_codi_postal,
    telefono        dt_telefono,
    trab_calle      dt_calle,
    trab_colonia    dt_colonia,
    trab_cp         dt_codi_postal,
    trab_telefono   dt_telefono,
    email           varchar(30) NULL,
    ingreso         dt_fecha,
    municipio       numeric(8) NULL,
    activo          bit DEFAULT "true",
 CONSTRAINT in_profesor_principal    PRIMARY KEY
(clave),
    CONSTRAINT co_nacimiento
        CHECK (nacimiento<ingreso),
    CONSTRAINT in_profesor_rfc
    UNIQUE (
        rfc
    )
)

exec sp_primarykey profesor,
    clave

exec sp_bindrule ru_fanterior, 'profesor.nacimiento'
grant all on profesor to desarrollo
grant select on profesor to consulta


CREATE TABLE profesor_grupo (
    grupo           numeric(16) NOT NULL,
    profesor        numeric(8) NOT NULL,
    inicio          dt_fecha,
    termino         dt_fecha,
    numero          tinyint NULL,
 CONSTRAINT in_profesor_grupo_principal    PRIMARY
KEY (grupo, profesor)
)

exec sp_primarykey profesor_grupo,
    grupo,
    profesor
grant all on profesor_grupo to desarrollo
grant select on profesor_grupo to consulta


CREATE TABLE programa (
    clave           numeric(8) IDENTITY,
    nivel           numeric(4) NOT NULL,
    carrera         numeric(4) NOT NULL,
    nombre          dt_nombre,
    descripcion     dt_descripcion,
    inicio          dt_fecha,
    termino         dt_fecha,
    obligatorios    dt_creditos,
    optativos       dt_creditos,
    asig_obl        tinyint NULL,
    periodos        tinyint NULL,
    terminado       bit,
    activo          bit,
 CONSTRAINT in_plan_principal    PRIMARY KEY (clave),
    CONSTRAINT in_plan_nombre
    UNIQUE (
        nombre,
        carrera
    )
)
```

```
exec sp_primarykey programa,
     clave
grant all on programa to desarrollo
grant select on programa to consulta


CREATE TABLE s_asignatura (
      antecedente        numeric(8) NOT NULL,
      consecuente        numeric(8) NOT NULL,
      simultanea         bit,
 CONSTRAINT in_s_asignatura_principal      PRIMARY KEY
(antecedente, consecuente),
      CONSTRAINT co_ser_asignatura
           CHECK (antecedente<>consecuente)
)

exec sp_primarykey s_asignatura,
     antecedente,
     consecuente
grant all on s_asignatura to desarrollo
grant select on s_asignatura to consulta


CREATE TABLE s_bloque (
      antecedente        numeric(8) NOT NULL,
      consecuente        numeric(8) NOT NULL,
      creditos           dt_creditos,
 CONSTRAINT in_s_bloque_principal      PRIMARY KEY
(antecedente, consecuente),
      CONSTRAINT co_seriacionbloque
           CHECK ((a_plan=c_plan) and (a_bloque<>c_bloque))
)

exec sp_primarykey s_bloque,
     antecedente,
     consecuente
grant all on s_bloque to desarrollo
grant select on s_bloque to consulta


CREATE TABLE tipo_asignatura (
      clave              numeric(4) IDENTITY,
      nombre             dt_nombre,
      activo             bit DEFAULT "true",
 CONSTRAINT in_tipo_asignatura_principal      PRIMARY
KEY (clave),
      CONSTRAINT in_tipo_asignatura_nombre
      UNIQUE (
           nombre
      )
)

exec sp_primarykey tipo_asignatura,
     clave
grant all on tipo_asignatura to desarrollo
grant select on tipo_asignatura to consulta


CREATE TABLE tipo_profesor (
      clave              numeric(4) IDENTITY,
      nombre             dt_nombre,
      descripcion        dt_descripcion,
      activo             bit DEFAULT "true",
 CONSTRAINT in_tipo_profesor_principal      PRIMARY KEY
(clave),
      CONSTRAINT in_tipo_profesor_nombre
      UNIQUE (
           nombre
      )
)

exec sp_primarykey tipo_profesor,
     clave
grant all on tipo_profesor to desarrollo
grant select on tipo_profesor to consulta


CREATE TABLE turno (
      clave              numeric(4) IDENTITY,
      nombre             dt_nombre,
      inicio             dt_fecha,
      termino            dt_fecha,
      activo             bit DEFAULT "true",
 CONSTRAINT in_turno_principal      PRIMARY KEY (clave),
      CONSTRAINT in_turno_nombre
      UNIQUE (
           nombre
      )
)

exec sp_primarykey turno,
     clave
grant all on turno to desarrollo
grant select on turno to consulta


ALTER TABLE acta
     ADD  CONSTRAINT es_registrado_en FOREIGN KEY
(grupo)
          REFERENCES grupo


exec sp_foreignkey acta, grupo,
     grupo

ALTER TABLE alumno
     ADD  CONSTRAINT posee_cierto FOREIGN KEY
(edo_civil)
          REFERENCES civil


ALTER TABLE alumno
     ADD  CONSTRAINT vive_en FOREIGN KEY  (municipio)
          REFERENCES municipio


ALTER TABLE alumno
     ADD  CONSTRAINT estudio_en FOREIGN KEY
(bachillerato)
          REFERENCES bachillerato


exec sp_foreignkey alumno, civil,
     edo_civil

exec sp_foreignkey alumno, municipio,
     municipio

exec sp_foreignkey alumno, bachillerato,
     bachillerato

ALTER TABLE alumno_asignatura
     ADD  CONSTRAINT en_cierto FOREIGN KEY  (cuenta,
programa)
          REFERENCES alumno_plan


ALTER TABLE alumno_asignatura
     ADD  CONSTRAINT tiene_para_cada_alumno_un
FOREIGN KEY  (asignatura)
          REFERENCES asignatura_plan
```

```
exec sp_foreignkey alumno_asignatura, alumno_plan,
    cuenta,
    programa

exec sp_foreignkey alumno_asignatura, asignatura_plan,
    asignatura

ALTER TABLE alumno_asignatura_modalidad
    ADD CONSTRAINT tiene_un_estado_para FOREIGN
KEY (cuenta, programa, asignatura)
        REFERENCES alumno_asignatura


ALTER TABLE alumno_asignatura_modalidad
    ADD CONSTRAINT lleva_cierto_numero_de_inscrip
FOREIGN KEY (modalidad)
        REFERENCES modalidad


exec sp_foreignkey alumno_asignatura_modalidad,
alumno_asignatura,
    cuenta,
    programa,
    asignatura

exec sp_foreignkey alumno_asignatura_modalidad, modalidad,
    modalidad

ALTER TABLE alumno_bloque
    ADD CONSTRAINT esta_en_cierto FOREIGN KEY
(cuenta, programa)
        REFERENCES alumno_plan


ALTER TABLE alumno_bloque
    ADD CONSTRAINT para_cada_alumno_tiene FOREIGN
KEY (bloque)
        REFERENCES bloque


exec sp_foreignkey alumno_bloque, alumno_plan,
    cuenta,
    programa

exec sp_foreignkey alumno_bloque, bloque,
    bloque

ALTER TABLE alumno_horario
    ADD CONSTRAINT hace_inscripciones_en FOREIGN
KEY (cuenta, programa)
        REFERENCES alumno_plan


ALTER TABLE alumno_horario
    ADD CONSTRAINT se_atiende_a FOREIGN KEY
(horario)
        REFERENCES horario_atencion


exec sp_foreignkey alumno_horario, alumno_plan,
    cuenta,
    programa

exec sp_foreignkey alumno_horario, horario_atencion,
    horario

ALTER TABLE alumno_plan
    ADD CONSTRAINT termino_de_cursar_el FOREIGN KEY
(termino)
        REFERENCES periodo


ALTER TABLE alumno_plan
    ADD CONSTRAINT empezo_a_cursar_el FOREIGN KEY
(inicio)
        REFERENCES periodo


ALTER TABLE alumno_plan
    ADD CONSTRAINT ingreso_por FOREIGN KEY
(ingreso)
        REFERENCES ingreso


ALTER TABLE alumno_plan
    ADD CONSTRAINT asiste_en_un FOREIGN KEY (turno)
        REFERENCES turno


ALTER TABLE alumno_plan
    ADD CONSTRAINT queda_clasificado_en_un FOREIGN
KEY (estado)
        REFERENCES estado_alumno


ALTER TABLE alumno_plan
    ADD CONSTRAINT tiene FOREIGN KEY (programa)
        REFERENCES programa


ALTER TABLE alumno_plan
    ADD CONSTRAINT esta_cursando FOREIGN KEY
(cuenta)
        REFERENCES alumno

exec sp_foreignkey alumno_plan, periodo,
    termino

exec sp_foreignkey alumno_plan, periodo,
    inicio

exec sp_foreignkey alumno_plan, ingreso,
    ingreso

exec sp_foreignkey alumno_plan, turno,
    turno

exec sp_foreignkey alumno_plan, estado_alumno,
    estado

exec sp_foreignkey alumno_plan, programa,
    programa

exec sp_foreignkey alumno_plan, alumno,
    cuenta

ALTER TABLE asignatura_bloque
    ADD CONSTRAINT forma_parte_de FOREIGN KEY
(asignatura)
        REFERENCES asignatura_plan


ALTER TABLE asignatura_bloque
    ADD CONSTRAINT esta_formado_por FOREIGN KEY
(bloque)
        REFERENCES bloque

exec sp_foreignkey asignatura_bloque, asignatura_plan,
```

```
    asignatura

exec sp_foreignkey asignatura_bloque, bloque,
    bloque

ALTER TABLE asignatura_plan
    ADD CONSTRAINT de_cierto FOREIGN KEY (tipo)
        REFERENCES tipo_asignatura


ALTER TABLE asignatura_plan
    ADD CONSTRAINT adquiere_caracteristicas_difer
FOREIGN KEY (asignatura)
        REFERENCES asignatura


ALTER TABLE asignatura_plan
    ADD CONSTRAINT define_varias FOREIGN KEY
(programa)
        REFERENCES programa


exec sp_foreignkey asignatura_plan, tipo_asignatura,
    tipo

exec sp_foreignkey asignatura_plan, asignatura,
    asignatura

exec sp_foreignkey asignatura_plan, programa,
    programa

ALTER TABLE bloque
    ADD CONSTRAINT consta_de FOREIGN KEY
(programa)
        REFERENCES programa


exec sp_foreignkey bloque, programa,
    programa

ALTER TABLE campus
    ADD CONSTRAINT se_encuentra_ubicado_en FOREIGN
KEY (municipio)
        REFERENCES municipio


exec sp_foreignkey campus, municipio,
    municipio

ALTER TABLE campus_carrera
    ADD CONSTRAINT imparte_varias FOREIGN KEY
(campus)
        REFERENCES campus


ALTER TABLE campus_carrera
    ADD CONSTRAINT se_imparte_en_un FOREIGN KEY
(carrera)
        REFERENCES carrera


exec sp_foreignkey campus_carrera, campus,
    campus

exec sp_foreignkey campus_carrera, carrera,
    carrera

ALTER TABLE grupo
    ADD CONSTRAINT se_encuentra_en_un FOREIGN KEY
(estado)
        REFERENCES estado_grupo

ALTER TABLE grupo
    ADD CONSTRAINT se_imparte_en FOREIGN KEY
(asignatura)
        REFERENCES asignatura_plan


ALTER TABLE grupo
    ADD CONSTRAINT le_corresponde_cierto FOREIGN
KEY (turno)
        REFERENCES turno


ALTER TABLE grupo
    ADD CONSTRAINT es_de_cierto FOREIGN KEY
(modalidad)
        REFERENCES modalidad


ALTER TABLE grupo
    ADD CONSTRAINT se_imparte_en_cierto FOREIGN KEY
(periodo)
        REFERENCES periodo


exec sp_foreignkey grupo, estado_grupo,
    estado

exec sp_foreignkey grupo, asignatura_plan,
    asignatura

exec sp_foreignkey grupo, turno,
    turno

exec sp_foreignkey grupo, modalidad,
    modalidad

exec sp_foreignkey grupo, periodo,
    periodo

ALTER TABLE horario
    ADD CONSTRAINT se_ocupa_por FOREIGN KEY (aula)
        REFERENCES aula


ALTER TABLE horario
    ADD CONSTRAINT toma_las_clases_en FOREIGN KEY
(grupo)
        REFERENCES grupo


exec sp_foreignkey horario, aula,
    aula

exec sp_foreignkey horario, grupo,
    grupo

ALTER TABLE horario_atencion
    ADD CONSTRAINT se_le_atiende_en_algunos FOREIGN
KEY (turno)
        REFERENCES turno


ALTER TABLE horario_atencion
    ADD CONSTRAINT hay FOREIGN KEY (periodo)
        REFERENCES periodo


ALTER TABLE horario_atencion
```

```
    ADD CONSTRAINT se_divide_en FOREIGN KEY
(modalidad)
        REFERENCES modalidad


exec sp_foreignkey horario_atencion, turno,
    turno

exec sp_foreignkey horario_atencion, periodo,
    periodo

exec sp_foreignkey horario_atencion, modalidad,
    modalidad

ALTER TABLE inscripcion
    ADD CONSTRAINT efectua FOREIGN KEY (cuenta,
programa)
        REFERENCES alumno_plan


ALTER TABLE inscripcion
    ADD CONSTRAINT se_encuentra_en FOREIGN KEY
(estado)
        REFERENCES estado_inscripcion


ALTER TABLE inscripcion
    ADD CONSTRAINT se_le_registra FOREIGN KEY
(grupo)
        REFERENCES grupo


ALTER TABLE inscripcion
    ADD CONSTRAINT se_obtiene_una FOREIGN KEY
(calificacion)
        REFERENCES calificacion


exec sp_foreignkey inscripcion, alumno_plan,
    cuenta,
    programa

exec sp_foreignkey inscripcion, estado_inscripcion,
    estado

exec sp_foreignkey inscripcion, grupo,
    grupo

exec sp_foreignkey inscripcion, calificacion,
    calificacion

ALTER TABLE modalidad_tipo
    ADD CONSTRAINT contiene_una_o_mas FOREIGN KEY
(tipo)
        REFERENCES tipo_asignatura


ALTER TABLE modalidad_tipo
    ADD CONSTRAINT puede_tener_reglas_especiales_
FOREIGN KEY (modalidad)
        REFERENCES modalidad


exec sp_foreignkey modalidad_tipo, tipo_asignatura,
    tipo

exec sp_foreignkey modalidad_tipo, modalidad,
    modalidad

ALTER TABLE municipio
    ADD CONSTRAINT esta_dividido_en FOREIGN KEY
(estado)
        REFERENCES estado


exec sp_foreignkey municipio, estado,
    estado

ALTER TABLE obs_prof
    ADD CONSTRAINT se_le_hacen FOREIGN KEY
(profesor)
        REFERENCES profesor


exec sp_foreignkey obs_prof, profesor,
    profesor

ALTER TABLE observacion
    ADD CONSTRAINT tiene_una FOREIGN KEY (cateria)
        REFERENCES cateria


ALTER TABLE observacion
    ADD CONSTRAINT se_le_realizan FOREIGN KEY
(cuenta)
        REFERENCES alumno


exec sp_foreignkey observacion, cateria,
    cateria

exec sp_foreignkey observacion, alumno,
    cuenta

ALTER TABLE profesor
    ADD CONSTRAINT es_alcanzado_por FOREIGN KEY
(grado)
        REFERENCES grado


ALTER TABLE profesor
    ADD CONSTRAINT se_encuentra_en_cierto FOREIGN
KEY (edo_civil)
        REFERENCES civil


ALTER TABLE profesor
    ADD CONSTRAINT vive_en_un FOREIGN KEY
(municipio)
        REFERENCES municipio


ALTER TABLE profesor
    ADD CONSTRAINT trabaja_en_un FOREIGN KEY
(trab_municipio)
        REFERENCES municipio


ALTER TABLE profesor
    ADD CONSTRAINT corresponde_a_cierto FOREIGN KEY
(tipo)
        REFERENCES tipo_profesor


exec sp_foreignkey profesor, grado,
    grado

exec sp_foreignkey profesor, civil,
    edo_civil

exec sp_foreignkey profesor, municipio,
```

```
    municipio

exec sp_foreignkey profesor, municipio,
    trab_municipio

exec sp_foreignkey profesor, tipo_profesor,
    tipo

ALTER TABLE profesor_grupo
    ADD CONSTRAINT da_clases_a_un FOREIGN KEY
(profesor)
        REFERENCES profesor


ALTER TABLE profesor_grupo
    ADD CONSTRAINT le_imparte_la_clase_un FOREIGN
KEY (grupo)
        REFERENCES grupo


exec sp_foreignkey profesor_grupo, profesor,
    profesor

exec sp_foreignkey profesor_grupo, grupo,
    grupo

ALTER TABLE programa
    ADD CONSTRAINT de_educacion_del FOREIGN KEY
(nivel)
        REFERENCES nivel


ALTER TABLE programa
    ADD CONSTRAINT consta_de_varios FOREIGN KEY
(carrera)
        REFERENCES carrera


exec sp_foreignkey programa, nivel,
    nivel

exec sp_foreignkey programa, carrera,
    carrera

ALTER TABLE s_asignatura
    ADD CONSTRAINT es_antecedente_de FOREIGN KEY
(antecedente)
        REFERENCES asignatura_plan


ALTER TABLE s_asignatura
    ADD CONSTRAINT es_consecuente_de FOREIGN KEY
(consecuente)
        REFERENCES asignatura_plan


exec sp_foreignkey s_asignatura, asignatura_plan,
    antecedente

exec sp_foreignkey s_asignatura, asignatura_plan,
    consecuente

ALTER TABLE s_bloque
    ADD CONSTRAINT antecedente_de FOREIGN KEY
(antecedente)
        REFERENCES bloque


ALTER TABLE s_bloque
    ADD CONSTRAINT consecuente_de FOREIGN KEY
(consecuente)
        REFERENCES bloque

exec sp_foreignkey s_bloque, bloque,
    antecedente

exec sp_foreignkey s_bloque, bloque,
    consecuente



create trigger tI_acta on acta for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* INSERT trigger on acta */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:04 1997 */
  /* grupo es registrado en acta ON CHILD INSERT RESTRICT
*/
  if
    /* %ChildFK(" or",update) */
    update(grupo)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,grupo
        where
          /* %JoinFKPK(inserted,grupo) */
          inserted.grupo = grupo.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "acta" because "grupo"
does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_acta on acta for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* UPDATE trigger on acta */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insfolio numeric(7),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* grupo es registrado en acta ON CHILD UPDATE RESTRICT
*/
  if
    /* %ChildFK(" or",update) */
    update(grupo)
  begin
```

```
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,grupo
     where
      /* %JoinFKPK(inserted,grupo) */
      inserted.grupo = grupo.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "acta" because "grupo"
does not exist.'
    to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_alumno on alumno for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* DELETE trigger on alumno */
begin
  declare  @errno   int,
        @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
   /* alumno se le realizan observacion ON PARENT DELETE
CASCADE */
   delete observacion
    from observacion,deleted
    where
     /*  %JoinFKPK(observacion,deleted," = "," and") */
     observacion.cuenta = deleted.cuenta
   /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
   /* alumno esta cursando alumno_plan ON PARENT DELETE
RESTRICT */
   if exists (
    select * from deleted,alumno_plan
    where
     /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
     alumno_plan.cuenta = deleted.cuenta
   )
   begin
    select @errno  = 30001,
        @errmsg = 'Cannot DELETE "alumno" because
"alumno_plan" exists.'
    to error
   end

   /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tI_alumno on alumno for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* INSERT trigger on alumno */
begin
  declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @errno   int,
        @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* civil esta en cierto alumno ON CHILD INSERT SET NULL */
  if
   /* %ChildFK(" or",update) */
   update(edo_civil)
  begin
   update alumno
    set
     /* %SetFK(alumno,NULL) */
     alumno.edo_civil = NULL
    from alumno,inserted
    where
     /* %JoinPKPK(alumno,inserted," = "," and") */
     alumno.cuenta = inserted.cuenta and
     not exists (
      select * from civil
      where
       /* %JoinFKPK(inserted,civil," = "," and") */
       inserted.edo_civil = civil.clave
     )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* municipio vive en alumno ON CHILD INSERT SET NULL */
  if
   /* %ChildFK(" or",update) */
   update(municipio)
  begin
   update alumno
    set
     /* %SetFK(alumno,NULL) */
     alumno.municipio = NULL
    from alumno,inserted
    where
     /* %JoinPKPK(alumno,inserted," = "," and") */
     alumno.cuenta = inserted.cuenta and
     not exists (
      select * from municipio
      where
       /* %JoinFKPK(inserted,municipio," = "," and") */
       inserted.municipio = municipio.clave
     )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* bachillerato estudio en alumno ON CHILD INSERT SET
NULL */
  if
   /* %ChildFK(" or",update) */
   update(bachillerato)
  begin
   update alumno
    set
     /* %SetFK(alumno,NULL) */
     alumno.bachillerato = NULL
    from alumno,inserted
    where
     /* %JoinPKPK(alumno,inserted," = "," and") */
     alumno.cuenta = inserted.cuenta and
     not exists (
      select * from bachillerato
      where
       /* %JoinFKPK(inserted,bachillerato," = "," and") */
       inserted.bachillerato = bachillerato.clave
     )
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
```

```
end

create trigger tU_alumno on alumno for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* UPDATE trigger on alumno */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @inscuenta dt_num_cuenta,
           @errno  int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno se le realizan observacion ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(cuenta)
  begin
    if @numrows = 1
    begin
      select @inscuenta = inserted.cuenta
        from inserted
      update observacion
      set
        /*  %JoinFKPK(observacion,@ins," = ",",") */
        observacion.cuenta = @inscuenta
      from observacion,inserted,deleted
      where
        /*  %JoinFKPK(observacion,deleted," = "," and") */
        observacion.cuenta = deleted.cuenta
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "alumno" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno esta cursando alumno_plan ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(cuenta)
  begin
    if @numrows = 1
    begin
      select @inscuenta = inserted.cuenta
        from inserted
      update alumno_plan
      set
        /*  %JoinFKPK(alumno_plan,@ins," = ",",") */
        alumno_plan.cuenta = @inscuenta
      from alumno_plan,inserted,deleted
      where
        /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
        alumno_plan.cuenta = deleted.cuenta
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "alumno" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* civil esta en cierto alumno ON CHILD UPDATE SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(edo_civil)
  begin
    update alumno
      set
        /* %SetFK(alumno,NULL) */
        alumno.edo_civil = NULL
      from alumno,inserted
      where
        /* %JoinPKPK(alumno,inserted," = "," and") */
        alumno.cuenta = inserted.cuenta and
        not exists (
          select * from civil
          where
            /* %JoinFKPK(inserted,civil," = "," and") */
            inserted.edo_civil = civil.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* municipio vive en alumno ON CHILD UPDATE SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(municipio)
  begin
    update alumno
      set
        /* %SetFK(alumno,NULL) */
        alumno.municipio = NULL
      from alumno,inserted
      where
        /* %JoinPKPK(alumno,inserted," = "," and") */
        alumno.cuenta = inserted.cuenta and
        not exists (
          select * from municipio
          where
            /* %JoinFKPK(inserted,municipio," = "," and") */
            inserted.municipio = municipio.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* bachillerato estudio en alumno ON CHILD UPDATE SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(bachillerato)
  begin
    update alumno
      set
        /* %SetFK(alumno,NULL) */
        alumno.bachillerato = NULL
      from alumno,inserted
      where
        /* %JoinPKPK(alumno,inserted," = "," and") */
        alumno.cuenta = inserted.cuenta and
        not exists (
          select * from bachillerato
          where
            /* %JoinFKPK(inserted,bachillerato," = "," and") */
            inserted.bachillerato = bachillerato.clave
        )
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end
```

```
create trigger tD_alumno_asignatura on alumno_asignatura for
DELETE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* DELETE trigger on alumno_asignatura */
begin
  declare  @errno  int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
    /* alumno_asignatura tiene un estado para cada
alumno_asignatura_modalidad ON PARENT DELETE
CASCADE */
    delete alumno_asignatura_modalidad
      from alumno_asignatura_modalidad,deleted
      where
        /*  %JoinFKPK(alumno_asignatura_modalidad,deleted," =
"," and") */
        alumno_asignatura_modalidad.cuenta = deleted.cuenta
and
        alumno_asignatura_modalidad.programa =
deleted.programa and
        alumno_asignatura_modalidad.asignatura =
deleted.asignatura

    /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tI_alumno_asignatura on alumno_asignatura for
INSERT as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* INSERT trigger on alumno_asignatura */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno  int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_plan esta en cierto alumno_asignatura ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cuenta) or
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno_plan
        where
          /* %JoinFKPK(inserted,alumno_plan) */
          inserted.cuenta = alumno_plan.cuenta and
          inserted.programa = alumno_plan.programa
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "alumno_asignatura"
because "alumno_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* asignatura_plan para cada alumno tiene un
alumno_asignatura ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(asignatura)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura_plan
        where
          /* %JoinFKPK(inserted,asignatura_plan) */
          inserted.asignatura = asignatura_plan.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "alumno_asignatura"
because "asignatura_plan" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_alumno_asignatura on alumno_asignatura for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* UPDATE trigger on alumno_asignatura */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @inscuenta dt_num_cuenta,
           @insprograma numeric(8),
           @insasignatura numeric(8),
           @errno  int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_asignatura tiene un estado para cada
alumno_asignatura_modalidad ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(cuenta) or
    update(programa) or
    update(asignatura)
  begin
    if @numrows = 1
    begin
      select @inscuenta = inserted.cuenta,
             @insprograma = inserted.programa,
             @insasignatura = inserted.asignatura
        from inserted
      update alumno_asignatura_modalidad
      set
        /*  %JoinFKPK(alumno_asignatura_modalidad,@ins," =
",",") */
        alumno_asignatura_modalidad.cuenta = @inscuenta,
        alumno_asignatura_modalidad.programa = @insprograma,
        alumno_asignatura_modalidad.asignatura =
@insasignatura
      from alumno_asignatura_modalidad,inserted,deleted
      where
        /*  %JoinFKPK(alumno_asignatura_modalidad,deleted," =
"," and") */
```

```
      alumno_asignatura_modalidad.cuenta = deleted.cuenta
and
      alumno_asignatura_modalidad.programa =
deleted.programa and
      alumno_asignatura_modalidad.asignatura =
deleted.asignatura
    end
    else
    begin
      select @ermo = 30006,
          @errmsg = 'Cannot cascade "alumno_asignatura"
 UPDATE because more than one row has been affected.'
      raiserror @ermo @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_plan esta en cierto alumno_asignatura ON CHILD
 UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cuenta) or
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno_plan
        where
          /* %JoinFKPK(inserted,alumno_plan) */
          inserted.cuenta = alumno_plan.cuenta and
          inserted.programa = alumno_plan.programa
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @ermo  = 30007,
          @errmsg = 'Cannot UPDATE "alumno_asignatura"
 because "alumno_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* asignatura_plan para cada alumno tiene un
 alumno_asignatura ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(asignatura)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura_plan
        where
          /* %JoinFKPK(inserted,asignatura_plan) */
          inserted.asignatura = asignatura_plan.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @ermo  = 30007,
          @errmsg = 'Cannot UPDATE "alumno_asignatura"
 because "asignatura_plan" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
  error:
    raiserror @ermo @errmsg
    rollback transaction
end

create trigger ti_alumno_asignatura_modalidad on
alumno_asignatura_modalidad for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* INSERT trigger on alumno_asignatura_modalidad */
begin
  declare  @numrows int,
         @nullcnt int,
         @validcnt int,
         @ermo  int,
         @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_asignatura tiene un estado para cada
 alumno_asignatura_modalidad ON CHILD INSERT RESTRICT
 */
  if
    /* %ChildFK(" or",update) */
    update(cuenta) or
    update(programa) or
    update(asignatura)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno_asignatura
        where
          /* %JoinFKPK(inserted,alumno_asignatura) */
          inserted.cuenta = alumno_asignatura.cuenta and
          inserted.programa = alumno_asignatura.programa and
          inserted.asignatura = alumno_asignatura.asignatura
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @ermo  = 30002,
          @errmsg = 'Cannot INSERT
 "alumno_asignatura_modalidad" because "alumno_asignatura"
 does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* modalidad lleva cierto numero de inscripciones en este tipo
 alumno_asignatura_modalidad ON CHILD INSERT RESTRICT
 */
  if
    /* %ChildFK(" or",update) */
    update(modalidad)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,modalidad
        where
          /* %JoinFKPK(inserted,modalidad) */
          inserted.modalidad = modalidad.modalidad
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @ermo  = 30002,
          @errmsg = 'Cannot INSERT
 "alumno_asignatura_modalidad" because "modalidad" does not
 exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
```

```
 return
error:
   raiserror @ermo @errmsg
   rollback transaction
end

create trigger tU_alumno_asignatura_modalidad on
alumno_asignatura_modalidad for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* UPDATE trigger on alumno_asignatura_modalidad */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @inscuenta dt_num_cuenta,
           @insprograma numeric(8),
           @insmodalidad numeric(4),
           @insasignatura numeric(8),
           @ermo   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_asignatura tiene un estado para cada
alumno_asignatura_modalidad ON CHILD UPDATE RESTRICT
*/
  if
    /* %ChildFK(" or",update) */
    update(cuenta) or
    update(programa) or
    update(asignatura)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno_asignatura
        where
          /* %JoinFKPK(inserted,alumno_asignatura) */
          inserted.cuenta = alumno_asignatura.cuenta and
          inserted.programa = alumno_asignatura.programa and
          inserted.asignatura = alumno_asignatura.asignatura
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @ermo  = 30007,
             @errmsg = 'Cannot UPDATE
"alumno_asignatura_modalidad" because "alumno_asignatura"
does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* modalidad lleva cierto numero de inscripciones en este tipo
alumno_asignatura_modalidad ON CHILD UPDATE RESTRICT
*/
  if
    /* %ChildFK(" or",update) */
    update(modalidad)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,modalidad
        where
          /* %JoinFKPK(inserted,modalidad) */
          inserted.modalidad = modalidad.modalidad
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @ermo  = 30007,
             @errmsg = 'Cannot UPDATE
"alumno_asignatura_modalidad" because "modalidad" does not
exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
   raiserror @ermo @errmsg
   rollback transaction
end

create trigger tI_alumno_bloque on alumno_bloque for INSERT
as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* INSERT trigger on alumno_bloque */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @ermo   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_plan esta en cierto alumno_bloque ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cuenta) or
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno_plan
        where
          /* %JoinFKPK(inserted,alumno_plan) */
          inserted.cuenta = alumno_plan.cuenta and
          inserted.programa = alumno_plan.programa
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @ermo  = 30002,
             @errmsg = 'Cannot INSERT "alumno_bloque" because
"alumno_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* bloque para cada alumno tiene alumno_bloque ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(bloque)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,bloque
        where
          /* %JoinFKPK(inserted,bloque) */
          inserted.bloque = bloque.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @ermo  = 30002,
```

```
        @errmsg = 'Cannot INSERT "alumno_bloque" because
"bloque" does not exist.'
    to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_alumno_bloque on alumno_bloque for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* UPDATE trigger on alumno_bloque */
begin
  declare  @numrows int,
         @nullcnt int,
         @validcnt int,
         @inscuenta dt_num_cuenta,
         @insprograma numeric(8),
         @insbloque numeric(8),
         @errno   int,
         @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_plan esta en cierto alumno_bloque ON CHILD
UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cuenta) or
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno_plan
        where
          /* %JoinFKPK(inserted,alumno_plan) */
          inserted.cuenta = alumno_plan.cuenta and
          inserted.programa = alumno_plan.programa
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
           @errmsg = 'Cannot UPDATE "alumno_bloque"
because "alumno_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* bloque para cada alumno tiene alumno_bloque ON CHILD
UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(bloque)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,bloque
        where
          /* %JoinFKPK(inserted,bloque) */
          inserted.bloque = bloque.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
           @errmsg = 'Cannot UPDATE "alumno_bloque"
because "bloque" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tI_alumno_horario on alumno_horario for INSERT
as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* INSERT trigger on alumno_horario */
begin
  declare  @numrows int,
         @nullcnt int,
         @validcnt int,
         @errno   int,
         @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_plan realiza inscripciones en alumno_horario ON
CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cuenta) or
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno_plan
        where
          /* %JoinFKPK(inserted,alumno_plan) */
          inserted.cuenta = alumno_plan.cuenta and
          inserted.programa = alumno_plan.programa
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
           @errmsg = 'Cannot INSERT "alumno_horario" because
"alumno_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* horario_atencion se atiende a alumno_horario ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(horario)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,horario_atencion
        where
          /* %JoinFKPK(inserted,horario_atencion) */
          inserted.horario = horario_atencion.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
```

```
        @errmsg = 'Cannot INSERT "alumno_horario" because
"horario_atencion" does not exist.'
    to error
   end
 end

 /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_alumno_horario on alumno_horario for
UPDATE as
/* ERwin Builtin Thu Aug 14 21 30:05 1997 */
/* UPDATE trigger on alumno_horario */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @inscuenta dt_num_cuenta,
        @insprograma numeric(8),
        @inshorario numeric(8),
        @errno   int,
        @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
 /* alumno_plan realiza inscripciones en alumno_horario ON
CHILD UPDATE RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(cuenta) or
   update(programa)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,alumno_plan
     where
      /* %JoinFKPK(inserted,alumno_plan) */
      inserted.cuenta = alumno_plan.cuenta and
      inserted.programa = alumno_plan.programa
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "alumno_horario"
 because "alumno_plan" does not exist.'
    to error
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* horario_atencion se atiende a alumno_horario ON CHILD
 UPDATE RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(horario)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,horario_atencion
     where
      /* %JoinFKPK(inserted,horario_atencion) */
      inserted.horario = horario_atencion.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "alumno_horario"
because "horario_atencion" does not exist.'
    to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_alumno_plan on alumno_plan for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* DELETE trigger on alumno_plan */
begin
 declare  @errno  int,
        @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
   /* alumno_plan esta en cierto alumno_bloque ON PARENT
DELETE CASCADE */
   delete alumno_bloque
    from alumno_bloque,deleted
    where
     /* %JoinFKPK(alumno_bloque,deleted," = "," and") */
     alumno_bloque.cuenta = deleted.cuenta and
     alumno_bloque.programa = deleted.programa
   /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
   /* alumno_plan esta en cierto alumno_asignatura ON
PARENT DELETE CASCADE */
   delete alumno_asignatura
    from alumno_asignatura,deleted
    where
     /* %JoinFKPK(alumno_asignatura,deleted," = "," and") */
     alumno_asignatura.cuenta = deleted.cuenta and
     alumno_asignatura.programa = deleted.programa
   /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
   /* alumno_plan realiza inscripcion ON PARENT DELETE
RESTRICT */
   if exists (
    select * from deleted,inscripcion
    where
     /* %JoinFKPK(inscripcion,deleted," = "," and") */
     inscripcion.cuenta = deleted.cuenta and
     inscripcion.programa = deleted.programa
   )
   begin
    select @errno  = 30001,
        @errmsg = 'Cannot DELETE "alumno_plan" because
"inscripcion" exists.'
    to error
   end
   /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
   /* alumno_plan realiza inscripciones en alumno_horario ON
PARENT DELETE RESTRICT */
   if exists (
    select * from deleted,alumno_horario
    where
     /* %JoinFKPK(alumno_horario,deleted," = "," and") */
     alumno_horario.cuenta = deleted.cuenta and
     alumno_horario.programa = deleted.programa
   )
   begin
    select @errno  = 30001,
        @errmsg = 'Cannot DELETE "alumno_plan" because
"alumno_horario" exists.'
    to error
   end
```

```
    /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tI_alumno_plan on alumno_plan for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* INSERT trigger on alumno_plan */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* periodo termino de cursar el alumno_plan ON CHILD
INSERT SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(termino)
  begin
    update alumno_plan
      set
        /* %SetFK(alumno_plan,NULL) */
        alumno_plan.termino = NULL
      from alumno_plan,inserted
      where
        /* %JoinPKPK(alumno_plan,inserted," = "," and") */
        alumno_plan.cuenta = inserted.cuenta and
        alumno_plan.programa = inserted.programa and
        not exists (
          select * from periodo
          where
            /* %JoinFKPK(inserted,periodo," = "," and") */
            inserted.termino = periodo.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* periodo empezo a cursar el alumno_plan ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(inicio)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,periodo
        where
          /* %JoinFKPK(inserted,periodo) */
          inserted.inicio = periodo.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "alumno_plan" because
"periodo" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* ingreso ingreso por alumno_plan ON CHILD INSERT SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(ingreso)
  begin
    update alumno_plan
      set
        /* %SetFK(alumno_plan,NULL) */
        alumno_plan.ingreso = NULL
      from alumno_plan,inserted
      where
        /* %JoinPKPK(alumno_plan,inserted," = "," and") */
        alumno_plan.cuenta = inserted.cuenta and
        alumno_plan.programa = inserted.programa and
        not exists (
          select * from ingreso
          where
            /* %JoinFKPK(inserted,ingreso," = "," and") */
            inserted.ingreso = ingreso.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* turno asiste en un alumno_plan ON CHILD INSERT SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(turno)
  begin
    update alumno_plan
      set
        /* %SetFK(alumno_plan,NULL) */
        alumno_plan.turno = NULL
      from alumno_plan,inserted
      where
        /* %JoinPKPK(alumno_plan,inserted," = "," and") */
        alumno_plan.cuenta = inserted.cuenta and
        alumno_plan.programa = inserted.programa and
        not exists (
          select * from turno
          where
            /* %JoinFKPK(inserted,turno," = "," and") */
            inserted.turno = turno.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* estado_alumno se encuentra en un alumno_plan ON CHILD
INSERT SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(estado)
  begin
    update alumno_plan
      set
        /* %SetFK(alumno_plan,NULL) */
        alumno_plan.estado = NULL
      from alumno_plan,inserted
      where
        /* %JoinPKPK(alumno_plan,inserted," = "," and") */
        alumno_plan.cuenta = inserted.cuenta and
        alumno_plan.programa = inserted.programa and
        not exists (
          select * from estado_alumno
          where
            /* %JoinFKPK(inserted,estado_alumno," = "," and") */
            inserted.estado = estado_alumno.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* programa tiene alumno_plan ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
```

```
    from inserted,programa
      where
       /* %JoinFKPK(inserted,programa) */
       inserted.programa = programa.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30002,
         @errmsg = 'Cannot INSERT "alumno_plan" because
"programa" does not exist.'
    to error
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno esta cursando alumno_plan ON CHILD INSERT
RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(cuenta)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,alumno
     where
      /* %JoinFKPK(inserted,alumno) */
      inserted.cuenta = alumno.cuenta
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30002,
         @errmsg = 'Cannot INSERT "alumno_plan" because
"alumno" does not exist.'
    to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_alumno_plan on alumno_plan for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:05 1997 */
/* UPDATE trigger on alumno_plan */
begin
 declare  @numrows int,
         @nullcnt int,
         @validcnt int,
         @inscuenta dt_num_cuenta,
         @insprograma numeric(8),
         @errno   int,
         @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
 /* alumno_plan esta en cierto alumno_bloque ON PARENT
UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(cuenta) or
  update(programa)
 begin
  if @numrows = 1
  begin
   select @inscuenta = inserted.cuenta,
        @insprograma = inserted.programa
    from inserted
   update alumno_bloque
   set
    /* %JoinFKPK(alumno_bloque,@ins," = ",",") */
    alumno_bloque.cuenta = @inscuenta,
    alumno_bloque.programa = @insprograma
   from alumno_bloque,inserted,deleted
   where
    /* %JoinFKPK(alumno_bloque,deleted," = "," and") */
    alumno_bloque.cuenta = deleted.cuenta and
    alumno_bloque.programa = deleted.programa
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "alumno_plan" UPDATE
because more than one row has been affected.'
   raiserror @errno @errmsg
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
 /* alumno_plan esta en cierto alumno_asignatura ON PARENT
UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(cuenta) or
  update(programa)
 begin
  if @numrows = 1
  begin
   select @inscuenta = inserted.cuenta,
        @insprograma = inserted.programa
    from inserted
   update alumno_asignatura
   set
    /* %JoinFKPK(alumno_asignatura,@ins," = ",",") */
    alumno_asignatura.cuenta = @inscuenta,
    alumno_asignatura.programa = @insprograma
   from alumno_asignatura,inserted,deleted
   where
    /* %JoinFKPK(alumno_asignatura,deleted," = "," and") */
    alumno_asignatura.cuenta = deleted.cuenta and
    alumno_asignatura.programa = deleted.programa
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "alumno_plan" UPDATE
because more than one row has been affected.'
   raiserror @errno @errmsg
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
 /* alumno_plan realiza inscripcion ON PARENT UPDATE
CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(cuenta) or
  update(programa)
 begin
  if @numrows = 1
  begin
   select @inscuenta = inserted.cuenta,
        @insprograma = inserted.programa
    from inserted
   update inscripcion
   set
    /* %JoinFKPK(inscripcion,@ins," = ",",") */
    inscripcion.cuenta = @inscuenta,
    inscripcion.programa = @insprograma
   from inscripcion,inserted,deleted
```

```
    where
     /*  %JoinFKPK(inscripcion,deleted," = "," and") */
     inscripcion.cuenta = deleted.cuenta and
     inscripcion.programa = deleted.programa
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "alumno_plan" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* alumno_plan realiza inscripciones en alumno_horario ON
PARENT UPDATE CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(cuenta) or
   update(programa)
  begin
   if @numrows = 1
   begin
    select @inscuenta = inserted.cuenta,
        @insprograma = inserted.programa
     from inserted
    update alumno_horario
    set
     /*  %JoinFKPK(alumno_horario,@ins," = ",",") */
     alumno_horario.cuenta = @inscuenta,
     alumno_horario.programa = @insprograma
    from alumno_horario,inserted,deleted
    where
     /*  %JoinFKPK(alumno_horario,deleted," = "," and") */
     alumno_horario.cuenta = deleted.cuenta and
     alumno_horario.programa = deleted.programa
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "alumno_plan" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* periodo termino de cursar el alumno_plan ON CHILD
UPDATE SET NULL */
  if
   /* %ChildFK(" or",update) */
   update(termino)
  begin
   update alumno_plan
    set
     /* %SetFK(alumno_plan,NULL) */
     alumno_plan.termino = NULL
    from alumno_plan,inserted
    where
     /* %JoinPKPK(alumno_plan,inserted," = "," and") */
     alumno_plan.cuenta = inserted.cuenta and
     alumno_plan.programa = inserted.programa and
     not exists (
      select * from periodo
      where
       /* %JoinFKPK(inserted,periodo," = "," and") */
       inserted.termino = periodo.clave
     )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* periodo empezo a cursar el alumno_plan ON CHILD
UPDATE RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(inicio)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,periodo
     where
      /* %JoinFKPK(inserted,periodo) */
      inserted.inicio = periodo.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "alumno_plan" because
"periodo" does not exist.'
    to error
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* ingreso ingreso por alumno_plan ON CHILD UPDATE SET
NULL */
  if
   /* %ChildFK(" or",update) */
   update(ingreso)
  begin
   update alumno_plan
    set
     /* %SetFK(alumno_plan,NULL) */
     alumno_plan.ingreso = NULL
    from alumno_plan,inserted
    where
     /* %JoinPKPK(alumno_plan,inserted," = "," and") */
     alumno_plan.cuenta = inserted.cuenta and
     alumno_plan.programa = inserted.programa and
     not exists (
      select * from ingreso
      where
       /* %JoinFKPK(inserted,ingreso," = "," and") */
       inserted.ingreso = ingreso.clave
     )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* turno asiste en un alumno_plan ON CHILD UPDATE SET
NULL */
  if
   /* %ChildFK(" or",update) */
   update(turno)
  begin
   update alumno_plan
    set
     /* %SetFK(alumno_plan,NULL) */
     alumno_plan.turno = NULL
    from alumno_plan,inserted
    where
     /* %JoinPKPK(alumno_plan,inserted," = "," and") */
     alumno_plan.cuenta = inserted.cuenta and
     alumno_plan.programa = inserted.programa and
     not exists (
      select * from turno
      where
       /* %JoinFKPK(inserted,turno," = "," and") */
       inserted.turno = turno.clave
     )
  end
  /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
  /* estado_alumno se encuentra en un alumno_plan ON CHILD
UPDATE SET NULL */
  if
```

```
  /* %ChildFK(" or",update) */
  update(estado)
 begin
  update alumno_plan
   set
    /* %SetFK(alumno_plan,NULL) */
    alumno_plan.estado = NULL
   from alumno_plan,inserted
   where
    /* %JoinPKPK(alumno_plan,inserted," = "," and") */
    alumno_plan.cuenta = inserted.cuenta and
    alumno_plan.programa = inserted.programa and
    not exists (
     select * from estado_alumno
     where
      /* %JoinFKPK(inserted,estado_alumno," = "," and") */
      inserted.estado = estado_alumno.clave
    )
 end
 /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
 /* programa tiene alumno_plan ON CHILD UPDATE
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(programa)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,programa
    where
     /* %JoinFKPK(inserted,programa) */
     inserted.programa = programa.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'Cannot UPDATE "alumno_plan" because
"programa" does not exist.'
    to error
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:05 1997 */
 /* alumno esta cursando alumno_plan ON CHILD UPDATE
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(cuenta)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,alumno
    where
     /* %JoinFKPK(inserted,alumno) */
     inserted.cuenta = alumno.cuenta
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'Cannot UPDATE "alumno_plan" because
"alumno" does not exist.'
    to error
  end
 end

 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tD_asignatura on asignatura for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on asignatura */
begin
 declare  @errno  int,
       @errmsg  varchar(255)
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* asignatura adquiere caracteristicas diferentes en cada
asignatura_plan ON PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,asignatura_plan
   where
    /* %JoinFKPK(asignatura_plan,deleted," = "," and") */
    asignatura_plan.asignatura = deleted.clave
  )
  begin
   select @errno  = 30001,
       @errmsg = 'Cannot DELETE "asignatura" because
"asignatura_plan" exists.'
    to error
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tU_asignatura on asignatura for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on asignatura */
begin
 declare  @numrows int,
       @nullcnt int,
       @validcnt int,
       @insclave numeric(8),
       @errno  int,
       @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* asignatura adquiere caracteristicas diferentes en cada
asignatura_plan ON PARENT UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update asignatura_plan
   set
    /* %JoinFKPK(asignatura_plan,@ins," = ",",") */
    asignatura_plan.asignatura = @insclave
   from asignatura_plan,inserted,deleted
   where
    /* %JoinFKPK(asignatura_plan,deleted," = "," and") */
    asignatura_plan.asignatura = deleted.clave
  end
  else
  begin
   select @errno = 30006,
       @errmsg = 'Cannot cascade "asignatura" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
  end
```

```
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tI_asignatura_bloque on asignatura_bloque for
INSERT as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* INSERT trigger on asignatura_bloque */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* asignatura_plan forma parte de asignatura_bloque ON
CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(asignatura)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura_plan
        where
          /* %JoinFKPK(inserted,asignatura_plan) */
          inserted.asignatura = asignatura_plan.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "asignatura_bloque"
because "asignatura_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* bloque esta formado por asignatura_bloque ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(bloque)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,bloque
        where
          /* %JoinFKPK(inserted,bloque) */
          inserted.bloque = bloque.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "asignatura_bloque"
because "bloque" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_asignatura_bloque on asignatura_bloque for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on asignatura_bloque */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insbloque numeric(8),
           @insasignatura numeric(8),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* asignatura_plan forma parte de asignatura_bloque ON
CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(asignatura)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura_plan
        where
          /* %JoinFKPK(inserted,asignatura_plan) */
          inserted.asignatura = asignatura_plan.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "asignatura_bloque"
because "asignatura_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* bloque esta formado por asignatura_bloque ON CHILD
UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(bloque)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,bloque
        where
          /* %JoinFKPK(inserted,bloque) */
          inserted.bloque = bloque.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "asignatura_bloque"
because "bloque" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
   raiserror @errno @errmsg
```

```
    rollback transaction
end

create trigger tD_asignatura_plan on asignatura_plan for
DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on asignatura_plan */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* asignatura_plan forma parte de asignatura_bloque ON
PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,asignatura_bloque
      where
        /*  %JoinFKPK(asignatura_bloque,deleted," = "," and") */
        asignatura_bloque.asignatura = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "asignatura_plan"
because "asignatura_bloque" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* asignatura_plan para cada alumno tiene un
alumno_asignatura ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,alumno_asignatura
      where
        /*  %JoinFKPK(alumno_asignatura,deleted," = "," and") */
        alumno_asignatura.asignatura = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "asignatura_plan"
because "alumno_asignatura" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* asignatura_plan se imparte en grupo ON PARENT
DELETE RESTRICT */
    if exists (
      select * from deleted,grupo
      where
        /*  %JoinFKPK(grupo,deleted," = "," and") */
        grupo.asignatura = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "asignatura_plan"
because "grupo" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* asignatura_plan es antecedente de s_asignatura ON
PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,s_asignatura
      where
        /*  %JoinFKPK(s_asignatura,deleted," = "," and") */
        s_asignatura.antecedente = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "asignatura_plan"
because "s_asignatura" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* asignatura_plan es consecuente de s_asignatura ON
PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,s_asignatura
      where
        /*  %JoinFKPK(s_asignatura,deleted," = "," and") */
        s_asignatura.consecuente = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "asignatura_plan"
because "s_asignatura" exists.'
      to error
    end

    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tI_asignatura_plan on asignatura_plan for
INSERT as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* INSERT trigger on asignatura_plan */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* tipo_asignatura es de cierto asignatura_plan ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(tipo)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,tipo_asignatura
        where
          /* %JoinFKPK(inserted,tipo_asignatura) */
          inserted.tipo = tipo_asignatura.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "asignatura_plan" because
"tipo_asignatura" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* asignatura adquiere caracteristicas diferentes en cada
asignatura_plan ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(asignatura)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura
        where
          /* %JoinFKPK(inserted,asignatura) */
          inserted.asignatura = asignatura.clave
```

```
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "asignatura_plan" because
"asignatura" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* programa define varias asignatura_plan ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,programa
        where
          /* %JoinFKPK(inserted,programa) */
          inserted.programa = programa.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "asignatura_plan" because
"programa" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_asignatura_plan on asignatura_plan for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on asignatura_plan */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(8),
           @errno  int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* asignatura_plan forma parte de asignatura_bloque ON
PARENT UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update asignatura_bloque
      set
        /*  %JoinFKPK(asignatura_bloque,@ins," = ",",") */
        asignatura_bloque.asignatura = @insclave
      from asignatura_bloque,inserted,deleted
      where
        /*  %JoinFKPK(asignatura_bloque,deleted," = "," and") */
        asignatura_bloque.asignatura = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "asignatura_plan"
UPDATE because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* asignatura_plan para cada alumno tiene un
alumno_asignatura ON PARENT UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update alumno_asignatura
      set
        /*  %JoinFKPK(alumno_asignatura,@ins," = ",",") */
        alumno_asignatura.asignatura = @insclave
      from alumno_asignatura,inserted,deleted
      where
        /*  %JoinFKPK(alumno_asignatura,deleted," = "," and") */
        alumno_asignatura.asignatura = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "asignatura_plan"
UPDATE because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* asignatura_plan se imparte en grupo ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update grupo
      set
        /*  %JoinFKPK(grupo,@ins," = ",",") */
        grupo.asignatura = @insclave
      from grupo,inserted,deleted
      where
        /*  %JoinFKPK(grupo,deleted," = "," and") */
        grupo.asignatura = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "asignatura_plan"
UPDATE because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* asignatura_plan es antecedente de s_asignatura ON
PARENT UPDATE CASCADE */
```

```
 if
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update s_asignatura
   set
    /*  %JoinFKPK(s_asignatura,@ins," = ",",") */
    s_asignatura.antecedente = @insclave
   from s_asignatura,inserted,deleted
   where
    /*  %JoinFKPK(s_asignatura,deleted," = " and") */
    s_asignatura.antecedente = deleted.clave
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "asignatura_plan"
UPDATE because more than one row has been affected.'
   raiserror @errno @errmsg
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* asignatura_plan es consecuente de s_asignatura ON
PARENT UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update s_asignatura
   set
    /*  %JoinFKPK(s_asignatura,@ins," = ",",") */
    s_asignatura.consecuente = @insclave
   from s_asignatura,inserted,deleted
   where
    /*  %JoinFKPK(s_asignatura,deleted," = " and") */
    s_asignatura.consecuente = deleted.clave
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "asignatura_plan"
UPDATE because more than one row has been affected.'
   raiserror @errno @errmsg
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* tipo_asignatura es de cierto asignatura_plan ON CHILD
UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(tipo)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,tipo_asignatura
    where
     /* %JoinFKPK(inserted,tipo_asignatura) */
     inserted.tipo = tipo_asignatura.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "asignatura_plan"
because "tipo_asignatura" does not exist.'
    to error
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* asignatura adquiere caracteristicas diferentes en cada
asignatura_plan ON CHILD UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(asignatura)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,asignatura
    where
     /* %JoinFKPK(inserted,asignatura) */
     inserted.asignatura = asignatura.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "asignatura_plan"
because "asignatura" does not exist.'
    to error
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* programa define varias asignatura_plan ON CHILD
UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(programa)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,programa
    where
     /* %JoinFKPK(inserted,programa) */
     inserted.programa = programa.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "asignatura_plan"
because "programa" does not exist.'
    to error
  end
 end

 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tD_aula on aula for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on aula */
begin
 declare  @errno   int,
        @errmsg  varchar(255)
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* aula se ocupa por horario ON PARENT DELETE
RESTRICT */
```

```
    if exists (
      select * from deleted,horario
      where
        /*  %JoinFKPK(horario,deleted," = "," and") */
        horario.aula = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "aula" because "horario"
exists.'
      to error
    end

    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_aula on aula for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on aula */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(8),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* aula se ocupa por horario ON PARENT UPDATE CASCADE
*/
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update horario
      set
        /*  %JoinFKPK(horario,@ins," = ",",") */
        horario.aula = @insclave
      from horario,inserted,deleted
      where
        /*  %JoinFKPK(horario,deleted," = "," and") */
        horario.aula = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "aula" UPDATE because
more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tD_bachillerato on bachillerato for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on bachillerato */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* bachillerato estudio en alumno ON PARENT DELETE
RESTRICT */
  if exists (
    select * from deleted,alumno
    where
      /*  %JoinFKPK(alumno,deleted," = "," and") */
      alumno.bachillerato = deleted.clave
  )
  begin
    select @errno  = 30001,
           @errmsg = 'Cannot DELETE "bachillerato" because
"alumno" exists.'
    to error
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tU_bachillerato on bachillerato for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on bachillerato */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* bachillerato estudio en alumno ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update alumno
      set
        /*  %JoinFKPK(alumno,@ins," = ",",") */
        alumno.bachillerato = @insclave
      from alumno,inserted,deleted
      where
        /*  %JoinFKPK(alumno,deleted," = "," and") */
        alumno.bachillerato = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "bachillerato" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end
```

```
create trigger tD_bloque on bloque for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on bloque */
begin
  declare  @errno  int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* bloque para cada alumno tiene alumno_bloque ON
PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,alumno_bloque
      where
        /*  %JoinFKPK(alumno_bloque,deleted," = "," and") */
        alumno_bloque.bloque = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "bloque" because
"alumno_bloque" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* bloque es antecedente de s_bloque ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,s_bloque
      where
        /*  %JoinFKPK(s_bloque,deleted," = "," and") */
        s_bloque.antecedente = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "bloque" because
"s_bloque" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* bloque es consecuente de s_bloque ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,s_bloque
      where
        /*  %JoinFKPK(s_bloque,deleted," = "," and") */
        s_bloque.consecuente = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "bloque" because
"s_bloque" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* bloque esta formado por asignatura_bloque ON PARENT
DELETE RESTRICT */
    if exists (
      select * from deleted,asignatura_bloque
      where
        /*  %JoinFKPK(asignatura_bloque,deleted," = "," and") */
        asignatura_bloque.bloque = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "bloque" because
"asignatura_bloque" exists.'
      to error
    end

    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tI_bloque on bloque for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* INSERT trigger on bloque */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno  int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* programa consta de bloque ON CHILD INSERT RESTRICT
*/
  if
    /* %ChildFK(" or",update) */
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,programa
        where
          /* %JoinFKPK(inserted,programa) */
          inserted.programa = programa.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "bloque" because
"programa" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_bloque on bloque for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on bloque */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(8),
           @errno  int,
           @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* bloque para cada alumno tiene alumno_bloque ON PARENT
UPDATE CASCADE */
 if
   /* %ParentPK(" or",update) */
   update(clave)
 begin
   if @numrows = 1
   begin
     select @insclave = inserted.clave
       from inserted
     update alumno_bloque
     set
       /*  %JoinFKPK(alumno_bloque,@ins," = ",",") */
```

```
      alumno_bloque.bloque = @insclave
    from alumno_bloque,inserted,deleted
    where
     /*  %JoinFKPK(alumno_bloque,deleted," = "," and") */
     alumno_bloque.bloque = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "bloque" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* bloque es antecedente de s_bloque ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update s_bloque
    set
     /*  %JoinFKPK(s_bloque,@ins," = ",",") */
     s_bloque.antecedente = @insclave
    from s_bloque,inserted,deleted
    where
     /*  %JoinFKPK(s_bloque,deleted," = "," and") */
     s_bloque.antecedente = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "bloque" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* bloque es consecuente de s_bloque ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update s_bloque
    set
     /*  %JoinFKPK(s_bloque,@ins," = ",",") */
     s_bloque.consecuente = @insclave
    from s_bloque,inserted,deleted
    where
     /*  %JoinFKPK(s_bloque,deleted," = "," and") */
     s_bloque.consecuente = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "bloque" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* bloque esta formado por asignatura_bloque ON PARENT
UPDATE CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update asignatura_bloque
    set
     /*  %JoinFKPK(asignatura_bloque,@ins," = ",",") */
     asignatura_bloque.bloque = @insclave
    from asignatura_bloque,inserted,deleted
    where
     /*  %JoinFKPK(asignatura_bloque,deleted," = "," and") */
     asignatura_bloque.bloque = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "bloque" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* programa consta de bloque ON CHILD UPDATE RESTRICT
*/
  if
   /* %ChildFK(" or",update) */
   update(programa)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,programa
     where
      /* %JoinFKPK(inserted,programa) */
      inserted.programa = programa.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30007,
         @errmsg = 'Cannot UPDATE "bloque" because
"programa" does not exist.'
    to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_calificacion on calificacion for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on calificacion */
begin
  declare  @errno   int,
         @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   /* calificacion se obtiene una inscripcion ON PARENT
DELETE RESTRICT */
   if exists (
    select * from deleted,inscripcion
    where
```

```
    /*  %JoinFKPK(inscripcion,deleted," = "," and") */
    inscripcion.calificacion = deleted.clave
  )
  begin
   select @errno  = 30001,
        @errmsg = 'Cannot DELETE "calificacion" because
"inscripcion" exists.'
    to error
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tU_calificacion on calificacion for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on calificacion */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @insclave numeric(4),
        @errno   int,
        @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* calificacion se obtiene una inscripcion ON PARENT
UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update inscripcion
   set
    /*  %JoinFKPK(inscripcion,@ins," = ",",") */
    inscripcion.calificacion = @insclave
   from inscripcion,inserted,deleted
   where
    /*  %JoinFKPK(inscripcion,deleted," = "," and") */
    inscripcion.calificacion = deleted.clave
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "calificacion" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
  end
 end

 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tD_campus on campus for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on campus */
begin
 declare  @errno   int,
        @errmsg  varchar(255)
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* campus imparte varias campus_carrera ON PARENT
DELETE RESTRICT */
  if exists (
   select * from deleted,campus_carrera
   where
    /*  %JoinFKPK(campus_carrera,deleted," = "," and") */
    campus_carrera.campus = deleted.clave
  )
  begin
   select @errno  = 30001,
        @errmsg = 'Cannot DELETE "campus" because
"campus_carrera" exists.'
    to error
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tI_campus on campus for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* INSERT trigger on campus */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @errno   int,
        @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* municipio se encuentra en un campus ON CHILD INSERT
SET NULL */
 if
  /* %ChildFK(" or",update) */
  update(municipio)
 begin
  update campus
   set
    /* %SetFK(campus,NULL) */
    campus.municipio = NULL
   from campus,inserted
   where
    /* %JoinPKPK(campus,inserted," = "," and") */
    campus.clave = inserted.clave and
    not exists (
     select * from municipio
     where
      /* %JoinFKPK(inserted,municipio," = "," and") */
      inserted.municipio = municipio.clave
    )
 end

 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tU_campus on campus for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on campus */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @insclave numeric(4),
        @errno   int,
```

```
        @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* campus imparte varias campus_carrera ON PARENT
UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update campus_carrera
      set
        /*  %JoinFKPK(campus_carrera,@ins," = ",",") */
        campus_carrera.campus = @insclave
      from campus_carrera,inserted,deleted
      where
        /*  %JoinFKPK(campus_carrera,deleted," = "," and") */
        campus_carrera.campus = deleted.clave
    end
    else
    begin
      select @errno = 30006,
          @errmsg = 'Cannot cascade "campus" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* municipio se encuentra en un campus ON CHILD UPDATE
SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(municipio)
  begin
    update campus
      set
        /* %SetFK(campus,NULL) */
        campus.municipio = NULL
      from campus,inserted
      where
        /* %JoinPKPK(campus,inserted," = "," and") */
        campus.clave = inserted.clave and
        not exists (
          select * from municipio
          where
            /* %JoinFKPK(inserted,municipio," = "," and") */
            inserted.municipio = municipio.clave
        )
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tI_campus_carrera on campus_carrera for
INSERT as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* INSERT trigger on campus_carrera */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* campus imparte varias campus_carrera ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(campus)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,campus
        where
          /* %JoinFKPK(inserted,campus) */
          inserted.campus = campus.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
          @errmsg = 'Cannot INSERT "campus_carrera"
because "campus" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* carrera se imparte en un campus_carrera ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(carrera)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,carrera
        where
          /* %JoinFKPK(inserted,carrera) */
          inserted.carrera = carrera.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
          @errmsg = 'Cannot INSERT "campus_carrera"
because "carrera" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_campus_carrera on campus_carrera for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on campus_carrera */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @inscampus numeric(4),
           @inscarrera numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
```

```
  /* campus imparte varias campus_carrera ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(campus)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,campus
        where
          /* %JoinFKPK(inserted,campus) */
          inserted.campus = campus.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'Cannot UPDATE "campus_carrera"
because "campus" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* carrera se imparte en un campus_carrera ON CHILD
UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(carrera)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,carrera
        where
          /* %JoinFKPK(inserted,carrera) */
          inserted.carrera = carrera.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'Cannot UPDATE "campus_carrera"
because "carrera" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tD_carrera on carrera for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on carrera */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* carrera consta de varios programa ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,programa
      where
        /*  %JoinFKPK(programa,deleted," = "," and") */
        programa.carrera = deleted.clave
    )
    begin
      select @errno  = 30001,
            @errmsg = 'Cannot DELETE "carrera" because
"programa" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* carrera se imparte en un campus_carrera ON PARENT
DELETE RESTRICT */
    if exists (
      select * from deleted,campus_carrera
      where
        /*  %JoinFKPK(campus_carrera,deleted," = "," and") */
        campus_carrera.carrera = deleted.clave
    )
    begin
      select @errno  = 30001,
            @errmsg = 'Cannot DELETE "carrera" because
"campus_carrera" exists.'
      to error
    end

    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_carrera on carrera for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on carrera */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* carrera consta de varios programa ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update programa
      set
        /*  %JoinFKPK(programa,@ins," = ",",") */
        programa.carrera = @insclave
      from programa,inserted,deleted
      where
        /*  %JoinFKPK(programa,deleted," = "," and") */
        programa.carrera = deleted.clave
    end
    else
    begin
      select @errno = 30006,
            @errmsg = 'Cannot cascade "carrera" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* carrera se imparte en un campus_carrera ON PARENT
UPDATE CASCADE */
  if
```

```
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update campus_carrera
   set
    /*  %JoinFKPK(campus_carrera,@ins," = ",",") */
    campus_carrera.carrera = @insclave
   from campus_carrera,inserted,deleted
   where
    /*  %JoinFKPK(campus_carrera,deleted," = "," and") */
    campus_carrera.carrera = deleted.clave
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "carrera" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
  end
 end

 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tD_cateria on cateria for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on cateria */
begin
 declare  @errno   int,
        @errmsg  varchar(255)
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* cateria tiene una observacion ON PARENT DELETE
RESTRICT */
  if exists (
   select * from deleted,observacion
   where
    /*  %JoinFKPK(observacion,deleted," = "," and") */
    observacion.cateria = deleted.clave
  )
  begin
   select @errno  = 30001,
        @errmsg = 'Cannot DELETE "cateria" because
"observacion" exists.'
    to error
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tU_cateria on cateria for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on cateria */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @insclave numeric(4),
        @errno   int,
        @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* cateria tiene una observacion ON PARENT UPDATE
CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update observacion
   set
    /*  %JoinFKPK(observacion,@ins," = ",",") */
    observacion.cateria = @insclave
   from observacion,inserted,deleted
   where
    /*  %JoinFKPK(observacion,deleted," = "," and") */
    observacion.cateria = deleted.clave
  end
  else
  begin
   select @errno = 30006,
        @errmsg = 'Cannot cascade "cateria" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
  end
 end

 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tD_civil on civil for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on civil */
begin
 declare  @errno   int,
        @errmsg  varchar(255)
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* civil esta en cierto alumno ON PARENT DELETE SET
NULL */
  update alumno
   set
    /* %SetFK(alumno,NULL) */
    alumno.edo_civil = NULL
   from alumno,deleted
   where
    /* %JoinFKPK(alumno,deleted," = "," and") */
    alumno.edo_civil = deleted.clave
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* civil se encuentra en cierto profesor ON PARENT DELETE
SET NULL */
  update profesor
   set
    /* %SetFK(profesor,NULL) */
    profesor.edo_civil = NULL
   from profesor,deleted
   where
    /* %JoinFKPK(profesor,deleted," = "," and") */
    profesor.edo_civil = deleted.clave

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
```

```
end

create trigger tU_civil on civil for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on civil */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* civil esta en cierto alumno ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update alumno
      set
        /*  %JoinFKPK(alumno,@ins," = ",",") */
        alumno.edo_civil = @insclave
      from alumno,inserted,deleted
      where
        /*  %JoinFKPK(alumno,deleted," = "," and") */
        alumno.edo_civil = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "civil" UPDATE because
more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* civil se encuentra en cierto profesor ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update profesor
      set
        /*  %JoinFKPK(profesor,@ins," = ",",") */
        profesor.edo_civil = @insclave
      from profesor,inserted,deleted
      where
        /*  %JoinFKPK(profesor,deleted," = "," and") */
        profesor.edo_civil = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "civil" UPDATE because
more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tD_estado on estado for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on estado */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* estado esta dividido en municipio ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,municipio
      where
        /*  %JoinFKPK(municipio,deleted," = "," and") */
        municipio.estado = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "estado" because
"municipio" exists.'
      to error
    end

    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_estado on estado for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on estado */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* estado esta dividido en municipio ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update municipio
      set
        /*  %JoinFKPK(municipio,@ins," = ",",") */
        municipio.estado = @insclave
      from municipio,inserted,deleted
      where
        /*  %JoinFKPK(municipio,deleted," = "," and") */
        municipio.estado = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "estado" UPDATE
because more than one row has been affected.'
```

```
    raiserror @errno @errmsg
   end
 end

 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_estado_alumno on estado_alumno for
DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on estado_alumno */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   /* estado_alumno se encuentra en un alumno_plan ON
PARENT DELETE SET NULL */
   update alumno_plan
    set
     /* %SetFK(alumno_plan,NULL) */
     alumno_plan.estado = NULL
    from alumno_plan,deleted
    where
     /* %JoinFKPK(alumno_plan,deleted," = "," and") */
     alumno_plan.estado = deleted.clave

   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_estado_alumno on estado_alumno for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on estado_alumno */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* estado_alumno se encuentra en un alumno_plan ON
 PARENT UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
       from inserted
      update alumno_plan
      set
       /*  %JoinFKPK(alumno_plan,@ins," = ",",") */
       alumno_plan.estado = @insclave
      from alumno_plan,inserted,deleted
      where
       /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
       alumno_plan.estado = deleted.clave
    end
    else
    begin
      select @errno = 30006,
           @errmsg = 'Cannot cascade "estado_alumno" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_estado_grupo on estado_grupo for DELETE
as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on estado_grupo */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   /* estado_grupo se encuentra en un grupo ON PARENT
DELETE SET NULL */
   update grupo
    set
     /* %SetFK(grupo,NULL) */
     grupo.estado = NULL
    from grupo,deleted
    where
     /* %JoinFKPK(grupo,deleted," = "," and") */
     grupo.estado = deleted.clave

   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_estado_grupo on estado_grupo for UPDATE
as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on estado_grupo */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  /* estado_grupo se encuentra en un grupo ON PARENT
 UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
       from inserted
      update grupo
      set
       /*  %JoinFKPK(grupo,@ins," = ",",") */
       grupo.estado = @insclave
      from grupo,inserted,deleted
      where
       /*  %JoinFKPK(grupo,deleted," = "," and") */
       grupo.estado = deleted.clave
```

```
    end
    else
    begin
     select @errno = 30006,
          @errmsg = 'Cannot cascade "estado_grupo" UPDATE
because more than one row has been affected.'
     raiserror @errno @errmsg
    end
   end

   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_estado_inscripcion on estado_inscripcion for
DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on estado_inscripcion */
begin
  declare @errno  int,
          @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* estado_inscripcion se encuentra en un inscripcion ON
PARENT DELETE SET NULL */
    update inscripcion
     set
      /* %SetFK(inscripcion,NULL) */
      inscripcion.estado = NULL
     from inscripcion,deleted
     where
      /* %JoinFKPK(inscripcion,deleted," = "," and") */
      inscripcion.estado = deleted.clave

   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_estado_inscripcion on estado_inscripcion for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on estado_inscripcion */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* estado_inscripcion se encuentra en un inscripcion ON
PARENT UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update inscripcion
   set
    /* %JoinFKPK(inscripcion,@ins," = ",",") */
    inscripcion.estado = @insclave
   from inscripcion,inserted,deleted
    where
     /* %JoinFKPK(inscripcion,deleted," = "," and") */
     inscripcion.estado = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "estado_inscripcion"
UPDATE because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_grado on grado for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on grado */
begin
  declare  @errno  int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
    /* grado es alcanzado por profesor ON PARENT DELETE
SET NULL */
    update profesor
     set
      /* %SetFK(profesor,NULL) */
      profesor.grado = NULL
     from profesor,deleted
     where
      /* %JoinFKPK(profesor,deleted," = "," and") */
      profesor.grado = deleted.clave

   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_grado on grado for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* UPDATE trigger on grado */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
 /* grado es alcanzado por profesor ON PARENT UPDATE
CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update profesor
   set
    /* %JoinFKPK(profesor,@ins," = ",",") */
    profesor.grado = @insclave
```

```
    from profesor,inserted,deleted
    where
     /*  %JoinFKPK(profesor,deleted," = "," and") */
     profesor.grado = deleted.clave
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "grado" UPDATE because
more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_grupo on grupo for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:06 1997 */
/* DELETE trigger on grupo */
begin
  declare  @errno   int,
        @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   /* grupo es registrado en acta ON PARENT DELETE
CASCADE */
   delete acta
    from acta,deleted
    where
     /*  %JoinFKPK(acta,deleted," = "," and") */
     acta.grupo = deleted.clave
   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   /* grupo toma las clases en horario ON PARENT DELETE
RESTRICT */
   if exists (
    select * from deleted,horario
    where
     /*  %JoinFKPK(horario,deleted," = "," and") */
     horario.grupo = deleted.clave
   )
   begin
    select @errno  = 30001,
        @errmsg = 'Cannot DELETE "grupo" because "horario"
exists.'
    to error
   end
   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   /* grupo le imparte la clase un profesor_grupo ON PARENT
DELETE RESTRICT */
   if exists (
    select * from deleted,profesor_grupo
    where
     /*  %JoinFKPK(profesor_grupo,deleted," = "," and") */
     profesor_grupo.grupo = deleted.clave
   )
   begin
    select @errno  = 30001,
        @errmsg = 'Cannot DELETE "grupo" because
"profesor_grupo" exists.'
    to error
   end
   /* ERwin Builtin Thu Aug 14 21:30:06 1997 */
   /* grupo se le registra inscripcion ON PARENT DELETE
RESTRICT */
   if exists (
    select * from deleted,inscripcion
    where
     /*  %JoinFKPK(inscripcion,deleted," = "," and") */
     inscripcion.grupo = deleted.clave
   )
   begin
    select @errno  = 30001,
        @errmsg = 'Cannot DELETE "grupo" because
"inscripcion" exists.'
    to error
   end

   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tI_grupo on grupo for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* INSERT trigger on grupo */
begin
  declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @errno   int,
        @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* estado_grupo se encuentra en un grupo ON CHILD INSERT
SET NULL */
  if
   /* %ChildFK(" or",update) */
   update(estado)
  begin
   update grupo
    set
     /* %SetFK(grupo,NULL) */
     grupo.estado = NULL
    from grupo,inserted
    where
     /* %JoinPKPK(grupo,inserted," = "," and") */
     grupo.clave = inserted.clave and
     not exists (
      select * from estado_grupo
      where
       /* %JoinFKPK(inserted,estado_grupo," = "," and") */
       inserted.estado = estado_grupo.clave
     )
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* asignatura_plan se imparte en grupo ON CHILD INSERT
RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(asignatura)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,asignatura_plan
     where
      /* %JoinFKPK(inserted,asignatura_plan) */
      inserted.asignatura = asignatura_plan.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30002,
        @errmsg = 'Cannot INSERT "grupo" because
"asignatura_plan" does not exist.'
    to error
```

```
    end
   end
   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   /* turno le corresponde cierto grupo ON CHILD INSERT SET
NULL */
   if
    /* %ChildFK(" or",update) */
    update(turno)
   begin
    update grupo
     set
      /* %SetFK(grupo,NULL) */
      grupo.turno = NULL
     from grupo,inserted
     where
      /* %JoinPKPK(grupo,inserted," = "," and") */
      grupo.clave = inserted.clave and
      not exists (
       select * from turno
       where
        /* %JoinFKPK(inserted,turno," = "," and") */
        inserted.turno = turno.clave
      )
   end
   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   /* modalidad es de cierto grupo ON CHILD INSERT
RESTRICT */
   if
    /* %ChildFK(" or",update) */
    update(modalidad)
   begin
    select @nullcnt = 0
    select @validcnt = count(*)
     from inserted,modalidad
      where
       /* %JoinFKPK(inserted,modalidad) */
       inserted.modalidad = modalidad.modalidad
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
     select @errno  = 30002,
          @errmsg = 'Cannot INSERT "grupo" because
"modalidad" does not exist.'
     to error
    end
   end
   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   /* periodo se imparte en cierto grupo ON CHILD INSERT
RESTRICT */
   if
    /* %ChildFK(" or",update) */
    update(periodo)
   begin
    select @nullcnt = 0
    select @validcnt = count(*)
     from inserted,periodo
      where
       /* %JoinFKPK(inserted,periodo) */
       inserted.periodo = periodo.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
     select @errno  = 30002,
          @errmsg = 'Cannot INSERT "grupo" because "periodo"
does not exist.'
     to error
    end
   end

   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_grupo on grupo for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on grupo */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(16),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* grupo es registrado en acta ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update acta
    set
     /* %JoinFKPK(acta,@ins," = ",",") */
     acta.grupo = @insclave
    from acta,inserted,deleted
    where
     /* %JoinFKPK(acta,deleted," = "," and") */
     acta.grupo = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "grupo" UPDATE because
more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* grupo toma las clases en horario ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update horario
    set
     /* %JoinFKPK(horario,@ins," = ",",") */
     horario.grupo = @insclave
    from horario,inserted,deleted
    where
     /* %JoinFKPK(horario,deleted," = "," and") */
     horario.grupo = deleted.clave
   end
   else
   begin
    select @errno = 30006,
```

```
        @errmsg = 'Cannot cascade "grupo" UPDATE because
more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* grupo le imparte la clase un profesor_grupo ON PARENT
UPDATE CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update profesor_grupo
    set
     /* %JoinFKPK(profesor_grupo,@ins," = ",",") */
     profesor_grupo.grupo = @insclave
    from profesor_grupo,inserted,deleted
    where
     /* %JoinFKPK(profesor_grupo,deleted," = "," and") */
     profesor_grupo.grupo = deleted.clave
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "grupo" UPDATE because
more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* grupo se le registra inscripcion ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update inscripcion
    set
     /* %JoinFKPK(inscripcion,@ins," = ",",") */
     inscripcion.grupo = @insclave
    from inscripcion,inserted,deleted
    where
     /* %JoinFKPK(inscripcion,deleted," = "," and") */
     inscripcion.grupo = deleted.clave
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "grupo" UPDATE because
more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* estado_grupo se encuentra en un grupo ON CHILD
UPDATE SET NULL */
  if
   /* %ChildFK(" or",update) */
   update(estado)
  begin
   update grupo
   set
    /* %SetFK(grupo,NULL) */
    grupo.estado = NULL
   from grupo,inserted
   where
    /* %JoinPKPK(grupo,inserted," = "," and") */
    grupo.clave = inserted.clave and
    not exists (
     select * from estado_grupo
     where
      /* %JoinFKPK(inserted,estado_grupo," = "," and") */
      inserted.estado = estado_grupo.clave
    )
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* asignatura_plan se imparte en grupo ON CHILD UPDATE
RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(asignatura)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,asignatura_plan
     where
      /* %JoinFKPK(inserted,asignatura_plan) */
      inserted.asignatura = asignatura_plan.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "grupo" because
"asignatura_plan" does not exist.'
    to error
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* turno le corresponde cierto grupo ON CHILD UPDATE SET
NULL */
  if
   /* %ChildFK(" or",update) */
   update(turno)
  begin
   update grupo
    set
     /* %SetFK(grupo,NULL) */
     grupo.turno = NULL
    from grupo,inserted
    where
     /* %JoinPKPK(grupo,inserted," = "," and") */
     grupo.clave = inserted.clave and
     not exists (
      select * from turno
      where
       /* %JoinFKPK(inserted,turno," = "," and") */
       inserted.turno = turno.clave
     )
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* modalidad es de cierto grupo ON CHILD UPDATE
RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(modalidad)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,modalidad
     where
      /* %JoinFKPK(inserted,modalidad) */
      inserted.modalidad = modalidad.modalidad
```

```
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
     select @errno  = 30007,
          @errmsg = 'Cannot UPDATE "grupo" because
"modalidad" does not exist.'
     to error
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* periodo se imparte en cierto grupo ON CHILD UPDATE
RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(periodo)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,periodo
     where
      /* %JoinFKPK(inserted,periodo) */
      inserted.periodo = periodo.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
     select @errno  = 30007,
          @errmsg = 'Cannot UPDATE "grupo" because
"periodo" does not exist.'
     to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tI_horario on horario for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* INSERT trigger on horario */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* aula se ocupa por horario ON CHILD INSERT RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(aula)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,aula
     where
      /* %JoinFKPK(inserted,aula) */
      inserted.aula = aula.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
     select @errno  = 30002,
          @errmsg = 'Cannot INSERT "horario" because "aula"
does not exist.'
     to error
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* grupo toma las clases en horario ON CHILD INSERT
RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(grupo)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,grupo
     where
      /* %JoinFKPK(inserted,grupo) */
      inserted.grupo = grupo.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
     select @errno  = 30002,
          @errmsg = 'Cannot INSERT "horario" because "grupo"
does not exist.'
     to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_horario on horario for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on horario */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insgrupo numeric(16),
           @insaula numeric(8),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* aula se ocupa por horario ON CHILD UPDATE RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(aula)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,aula
     where
      /* %JoinFKPK(inserted,aula) */
      inserted.aula = aula.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
     select @errno  = 30007,
          @errmsg = 'Cannot UPDATE "horario" because "aula"
does not exist.'
     to error
   end
```

```
 end
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* grupo toma las clases en horario ON CHILD UPDATE
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(grupo)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,grupo
    where
     /* %JoinFKPK(inserted,grupo) */
     inserted.grupo = grupo.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'Cannot UPDATE "horario" because "grupo"
does not exist.'
    to error
  end
 end

 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tD_horario_atencion on horario_atencion for
DELETE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* DELETE trigger on horario_atencion */
begin
 declare  @errno  int,
       @errmsg  varchar(255)
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* horario_atencion se atiende a alumno_horario ON
PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,alumno_horario
   where
    /*  %JoinFKPK(alumno_horario,deleted," = "," and") */
    alumno_horario.horario = deleted.clave
  )
  begin
   select @errno  = 30001,
       @errmsg = 'Cannot DELETE "horario_atencion"
because "alumno_horario" exists.'
    to error
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tI_horario_atencion on horario_atencion for
INSERT as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* INSERT trigger on horario_atencion */
begin
 declare  @numrows int,
       @nullcnt int,
       @validcnt int,
       @errno  int,
       @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* turno se le atiende en algunos horario_atencion ON CHILD
INSERT SET NULL */
 if
  /* %ChildFK(" or",update) */
  update(turno)
 begin
  update horario_atencion
   set
    /* %SetFK(horario_atencion,NULL) */
    horario_atencion.turno = NULL
   from horario_atencion,inserted
   where
    /* %JoinPKPK(horario_atencion,inserted," = "," and") */
    horario_atencion.clave = inserted.clave and
    not exists (
     select * from turno
     where
      /* %JoinFKPK(inserted,turno," = "," and") */
      inserted.turno = turno.clave
    )
 end
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* periodo hay horario_atencion ON CHILD INSERT
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(periodo)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,periodo
    where
     /* %JoinFKPK(inserted,periodo) */
     inserted.periodo = periodo.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
       @errmsg = 'Cannot INSERT "horario_atencion"
because "periodo" does not exist.'
    to error
  end
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* modalidad se divide en horario_atencion ON CHILD INSERT
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(modalidad)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,modalidad
    where
     /* %JoinFKPK(inserted,modalidad) */
     inserted.modalidad = modalidad.modalidad
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
       @errmsg = 'Cannot INSERT "horario_atencion"
because "modalidad" does not exist.'
    to error
```

```
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_horario_atencion on horario_atencion for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on horario_atencion */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(8),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* horario_atencion se atiende a alumno_horario ON PARENT
UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update alumno_horario
      set
        /*  %JoinFKPK(alumno_horario,@ins," = ",",") */
        alumno_horario.horario = @insclave
      from alumno_horario,inserted,deleted
      where
        /*  %JoinFKPK(alumno_horario,deleted," = "," and") */
        alumno_horario.horario = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "horario_atencion"
UPDATE because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* turno se le atiende en algunos horario_atencion ON CHILD
UPDATE SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(turno)
  begin
    update horario_atencion
      set
        /* %SetFK(horario_atencion,NULL) */
        horario_atencion.turno = NULL
      from horario_atencion,inserted
      where
        /* %JoinPKPK(horario_atencion,inserted," = "," and") */
        horario_atencion.clave = inserted.clave and
        not exists (
          select * from turno
          where
            /* %JoinFKPK(inserted,turno," = "," and") */
            inserted.turno = turno.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* periodo hay horario_atencion ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(periodo)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,periodo
        where
          /* %JoinFKPK(inserted,periodo) */
          inserted.periodo = periodo.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "horario_atencion"
because "periodo" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* modalidad se divide en horario_atencion ON CHILD
UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(modalidad)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,modalidad
        where
          /* %JoinFKPK(inserted,modalidad) */
          inserted.modalidad = modalidad.modalidad
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "horario_atencion"
because "modalidad" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tD_ingreso on ingreso for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* DELETE trigger on ingreso */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
    /* ingreso ingreso por alumno_plan ON PARENT DELETE
SET NULL */
    update alumno_plan
      set
        /* %SetFK(alumno_plan,NULL) */
        alumno_plan.ingreso = NULL
      from alumno_plan,deleted
```

```
    where
     /* %JoinFKPK(alumno_plan,deleted," = "," and") */
     alumno_plan.ingreso = deleted.clave

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tU_ingreso on ingreso for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on ingreso */
begin
 declare  @numrows int,
          @nullcnt int,
          @validcnt int,
          @insclave numeric(4),
          @errno   int,
          @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* ingreso ingreso por alumno_plan ON PARENT UPDATE
CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(clave)
 begin
  if @numrows = 1
  begin
   select @insclave = inserted.clave
    from inserted
   update alumno_plan
   set
    /*  %JoinFKPK(alumno_plan,@ins," = ",",") */
    alumno_plan.ingreso = @insclave
   from alumno_plan,inserted,deleted
   where
    /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
    alumno_plan.ingreso = deleted.clave
  end
  else
  begin
   select @errno = 30006,
          @errmsg = 'Cannot cascade "ingreso" UPDATE
because more than one row has been affected.'
   raiserror @errno @errmsg
  end
 end

 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tI_inscripcion on inscripcion for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* INSERT trigger on inscripcion */
begin
 declare  @numrows int,
          @nullcnt int,
          @validcnt int,
          @errno   int,
          @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* alumno_plan realiza inscripcion ON CHILD INSERT
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(cuenta) or
  update(programa)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,alumno_plan
    where
     /* %JoinFKPK(inserted,alumno_plan) */
     inserted.cuenta = alumno_plan.cuenta and
     inserted.programa = alumno_plan.programa
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
          @errmsg = 'Cannot INSERT "inscripcion" because
"alumno_plan" does not exist.'
   to error
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* estado_inscripcion se encuentra en un inscripcion ON
CHILD INSERT SET NULL */
 if
  /* %ChildFK(" or",update) */
  update(estado)
 begin
  update inscripcion
   set
    /* %SetFK(inscripcion,NULL) */
    inscripcion.estado = NULL
   from inscripcion,inserted
   where
    /* %JoinPKPK(inscripcion,inserted," = "," and") */
    inscripcion.programa = inserted.programa and
    inscripcion.cuenta = inserted.cuenta and
    inscripcion.grupo = inserted.grupo and
    not exists (
     select * from estado_inscripcion
     where
      /* %JoinFKPK(inserted,estado_inscripcion," = "," and") */
      inserted.estado = estado_inscripcion.clave
    )
 end
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* grupo se le registra inscripcion ON CHILD INSERT
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(grupo)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,grupo
    where
     /* %JoinFKPK(inserted,grupo) */
     inserted.grupo = grupo.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
          @errmsg = 'Cannot INSERT "inscripcion" because
"grupo" does not exist.'
   to error
  end
 end
```

```
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* calificacion se obtiene una inscripcion ON CHILD INSERT
 SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(calificacion)
  begin
    update inscripcion
      set
        /* %SetFK(inscripcion,NULL) */
        inscripcion.calificacion = NULL
      from inscripcion,inserted
      where
        /* %JoinPKPK(inscripcion,inserted," = "," and") */
        inscripcion.programa = inserted.programa and
        inscripcion.cuenta = inserted.cuenta and
        inscripcion.grupo = inserted.grupo and
        not exists (
          select * from calificacion
          where
            /* %JoinFKPK(inserted,calificacion," = "," and") */
            inserted.calificacion = calificacion.clave
        )
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_inscripcion on inscripcion for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on inscripcion */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insprograma numeric(8),
           @inscuenta dt_num_cuenta,
           @insgrupo numeric(16),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* alumno_plan realiza inscripcion ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cuenta) or
    update(programa)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno_plan
        where
          /* %JoinFKPK(inserted,alumno_plan) */
          inserted.cuenta = alumno_plan.cuenta and
          inserted.programa = alumno_plan.programa
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "inscripcion" because
"alumno_plan" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* estado_inscripcion se encuentra en un inscripcion ON
CHILD UPDATE SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(estado)
  begin
    update inscripcion
      set
        /* %SetFK(inscripcion,NULL) */
        inscripcion.estado = NULL
      from inscripcion,inserted
      where
        /* %JoinPKPK(inscripcion,inserted," = "," and") */
        inscripcion.programa = inserted.programa and
        inscripcion.cuenta = inserted.cuenta and
        inscripcion.grupo = inserted.grupo and
        not exists (
          select * from estado_inscripcion
          where
            /* %JoinFKPK(inserted,estado_inscripcion," = "," and") */
            inserted.estado = estado_inscripcion.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* grupo se le registra inscripcion ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(grupo)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,grupo
        where
          /* %JoinFKPK(inserted,grupo) */
          inserted.grupo = grupo.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "inscripcion" because
"grupo" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* calificacion se obtiene una inscripcion ON CHILD UPDATE
SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(calificacion)
  begin
    update inscripcion
      set
        /* %SetFK(inscripcion,NULL) */
        inscripcion.calificacion = NULL
      from inscripcion,inserted
      where
        /* %JoinPKPK(inscripcion,inserted," = "," and") */
        inscripcion.programa = inserted.programa and
        inscripcion.cuenta = inserted.cuenta and
        inscripcion.grupo = inserted.grupo and
        not exists (
          select * from calificacion
          where
            /* %JoinFKPK(inserted,calificacion," = "," and") */
            inserted.calificacion = calificacion.clave
        )
```

```
 end

 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_modalidad on modalidad for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* DELETE trigger on modalidad */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
    /* modalidad puede tener reglas especiales para cierto
modalidad_tipo ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,modalidad_tipo
      where
        /*  %JoinFKPK(modalidad_tipo,deleted," = "," and") */
        modalidad_tipo.modalidad = deleted.modalidad
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "modalidad" because
"modalidad_tipo" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
    /* modalidad lleva cierto numero de inscripciones en este tipo
alumno_asignatura_modalidad ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,alumno_asignatura_modalidad
      where
        /*  %JoinFKPK(alumno_asignatura_modalidad,deleted," =
"," and") */
        alumno_asignatura_modalidad.modalidad =
deleted.modalidad
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "modalidad" because
"alumno_asignatura_modalidad" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
    /* modalidad se divide en horario_atencion ON PARENT
DELETE CASCADE */
    delete horario_atencion
      from horario_atencion,deleted
      where
        /*  %JoinFKPK(horario_atencion,deleted," = "," and") */
        horario_atencion.modalidad = deleted.modalidad
    /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
    /* modalidad es de cierto grupo ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,grupo
      where
        /*  %JoinFKPK(grupo,deleted," = "," and") */
        grupo.modalidad = deleted.modalidad
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "modalidad" because
"grupo" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_modalidad on modalidad for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on modalidad */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insmodalidad numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* modalidad puede tener reglas especiales para cierto
modalidad_tipo ON PARENT UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(modalidad)
  begin
    if @numrows = 1
    begin
      select @insmodalidad = inserted.modalidad
        from inserted
      update modalidad_tipo
      set
        /*  %JoinFKPK(modalidad_tipo,@ins," = ","") */
        modalidad_tipo.modalidad = @insmodalidad
      from modalidad_tipo,inserted,deleted
      where
        /*  %JoinFKPK(modalidad_tipo,deleted," = "," and") */
        modalidad_tipo.modalidad = deleted.modalidad
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "modalidad" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* modalidad lleva cierto numero de inscripciones en este tipo
alumno_asignatura_modalidad ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(modalidad)
  begin
    if @numrows = 1
    begin
      select @insmodalidad = inserted.modalidad
        from inserted
      update alumno_asignatura_modalidad
      set
        /*  %JoinFKPK(alumno_asignatura_modalidad,@ins," =
","") */
        alumno_asignatura_modalidad.modalidad =
@insmodalidad
      from alumno_asignatura_modalidad,inserted,deleted
      where
        /*  %JoinFKPK(alumno_asignatura_modalidad,deleted," =
"," and") */
        alumno_asignatura_modalidad.modalidad =
deleted.modalidad
```

```
  end
  else
  begin
   select @ermo = 30006,
        @errmsg = 'Cannot cascade "modalidad" UPDATE
because more than one row has been affected.'
    raiserror @ermo @errmsg
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* modalidad se divide en horario_atencion ON PARENT
UPDATE CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(modalidad)
 begin
  if @numrows = 1
  begin
   select @insmodalidad = inserted.modalidad
    from inserted
   update horario_atencion
   set
    /* %JoinFKPK(horario_atencion,@ins," = ",",") */
    horario_atencion.modalidad = @insmodalidad
   from horario_atencion,inserted,deleted
   where
    /* %JoinFKPK(horario_atencion,deleted," = "," and") */
    horario_atencion.modalidad = deleted.modalidad
  end
  else
  begin
   select @ermo = 30006,
        @errmsg = 'Cannot cascade "modalidad" UPDATE
because more than one row has been affected.'
    raiserror @ermo @errmsg
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* modalidad es de cierto grupo ON PARENT UPDATE
CASCADE */
 if
  /* %ParentPK(" or",update) */
  update(modalidad)
 begin
  if @numrows = 1
  begin
   select @insmodalidad = inserted.modalidad
    from inserted
   update grupo
   set
    /* %JoinFKPK(grupo,@ins," = ",",") */
    grupo.modalidad = @insmodalidad
   from grupo,inserted,deleted
   where
    /* %JoinFKPK(grupo,deleted," = "," and") */
    grupo.modalidad = deleted.modalidad
  end
  else
  begin
   select @ermo = 30006,
        @errmsg = 'Cannot cascade "modalidad" UPDATE
because more than one row has been affected.'
    raiserror @ermo @errmsg
  end

 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 return
error:
   raiserror @ermo @errmsg
   rollback transaction
end

create trigger tI_modalidad_tipo on modalidad_tipo for INSERT
as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* INSERT trigger on modalidad_tipo */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @ermo   int,
        @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* tipo_asignatura contiene una o mas modalidad_tipo ON
CHILD INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(tipo)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,tipo_asignatura
    where
     /* %JoinFKPK(inserted,tipo_asignatura) */
     inserted.tipo = tipo_asignatura.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @ermo  = 30002,
        @errmsg = 'Cannot INSERT "modalidad_tipo" because
"tipo_asignatura" does not exist.'
    to error
  end
 end
 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 /* modalidad puede tener reglas especiales para cierto
modalidad_tipo ON CHILD INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(modalidad)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,modalidad
    where
     /* %JoinFKPK(inserted,modalidad) */
     inserted.modalidad = modalidad.modalidad
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @ermo  = 30002,
        @errmsg = 'Cannot INSERT "modalidad_tipo" because
"modalidad" does not exist.'
    to error
  end

 /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
 return
error:
   raiserror @ermo @errmsg
   rollback transaction
end

create trigger tU_modalidad_tipo on modalidad_tipo for
UPDATE as
```

```
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on modalidad_tipo */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @instipo numeric(4),
           @insmodalidad numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* tipo_asignatura contiene una o mas modalidad_tipo ON
CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(tipo)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,tipo_asignatura
        where
          /* %JoinFKPK(inserted,tipo_asignatura) */
          inserted.tipo = tipo_asignatura.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'Cannot UPDATE "modalidad_tipo" because
"tipo_asignatura" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* modalidad puede tener reglas especiales para derto
modalidad_tipo ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(modalidad)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,modalidad
        where
          /* %JoinFKPK(inserted,modalidad) */
          inserted.modalidad = modalidad.modalidad
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'Cannot UPDATE "modalidad_tipo" because
"modalidad" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tD_municipio on municipio for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* DELETE trigger on municipio */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* municipio vive en alumno ON PARENT DELETE SET
NULL */
  update alumno
    set
      /* %SetFK(alumno,NULL) */
      alumno.municipio = NULL
    from alumno,deleted
    where
      /* %JoinFKPK(alumno,deleted," = "," and") */
      alumno.municipio = deleted.clave
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* municipio vive en un profesor ON PARENT DELETE SET
NULL */
  update profesor
    set
      /* %SetFK(profesor,NULL) */
      profesor.municipio = NULL
    from profesor,deleted
    where
      /* %JoinFKPK(profesor,deleted," = "," and") */
      profesor.municipio = deleted.clave
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* municipio trabaja en un profesor ON PARENT DELETE
SET NULL */
  update profesor
    set
      /* %SetFK(profesor,NULL) */
      profesor.trab_municipio = NULL
    from profesor,deleted
    where
      /* %JoinFKPK(profesor,deleted," = "," and") */
      profesor.trab_municipio = deleted.clave
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* municipio se encuentra en un campus ON PARENT
DELETE SET NULL */
  update campus
    set
      /* %SetFK(campus,NULL) */
      campus.municipio = NULL
    from campus,deleted
    where
      /* %JoinFKPK(campus,deleted," = "," and") */
      campus.municipio = deleted.clave

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tI_municipio on municipio for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* INSERT trigger on municipio */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* estado esta dividido en municipio ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(estado)
  begin
```

```
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,estado
     where
      /* %JoinFKPK(inserted,estado) */
      inserted.estado = estado.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30002,
         @errmsg = 'Cannot INSERT "municipio" because
"estado" does not exist.'
     to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_municipio on municipio for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on municipio */
begin
  declare  @numrows int,
         @nullcnt int,
         @validcnt int,
         @insclave numeric(8),
         @errno  int,
         @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* municipio vive en alumno ON PARENT UPDATE CASCADE
*/
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update alumno
    set
     /*  %JoinFKPK(alumno,@ins," = ",",") */
     alumno.municipio = @insclave
    from alumno,inserted,deleted
    where
     /*  %JoinFKPK(alumno,deleted," = "," and") */
     alumno.municipio = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "municipio" UPDATE
because more than one row has been affected.'
     raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* municipio vive en un profesor ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update profesor
    set
     /*  %JoinFKPK(profesor,@ins," = ",",") */
     profesor.municipio = @insclave
    from profesor,inserted,deleted
    where
     /*  %JoinFKPK(profesor,deleted," = "," and") */
     profesor.municipio = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "municipio" UPDATE
because more than one row has been affected.'
     raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* municipio trabaja en un profesor ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update profesor
    set
     /*  %JoinFKPK(profesor,@ins," = ",",") */
     profesor.trab_municipio = @insclave
    from profesor,inserted,deleted
    where
     /*  %JoinFKPK(profesor,deleted," = "," and") */
     profesor.trab_municipio = deleted.clave
   end
   else
   begin
    select @errno = 30006,
         @errmsg = 'Cannot cascade "municipio" UPDATE
because more than one row has been affected.'
     raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* municipio se encuentra en un campus ON PARENT
UPDATE CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update campus
    set
     /*  %JoinFKPK(campus,@ins," = ",",") */
     campus.municipio = @insclave
    from campus,inserted,deleted
    where
     /*  %JoinFKPK(campus,deleted," = "," and") */
     campus.municipio = deleted.clave
   end
   else
   begin
```

```
    select @errno = 30006,
        @errmsg = 'Cannot cascade "municipio" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* estado esta dividido en municipio ON CHILD UPDATE
RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(estado)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,estado
     where
      /* %JoinFKPK(inserted,estado) */
      inserted.estado = estado.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30007,
        @errmsg = 'Cannot UPDATE "municipio" because
"estado" does not exist.'
     to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_nivel on nivel for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* DELETE trigger on nivel */
begin
  declare  @errno  int,
        @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   /* nivel de educacion del programa ON PARENT DELETE
RESTRICT */
   if exists (
    select * from deleted,programa
    where
     /* %JoinFKPK(programa,deleted," = "," and") */
     programa.nivel = deleted.clave
   )
   begin
    select @errno  = 30001,
        @errmsg = 'Cannot DELETE "nivel" because
"programa" exists.'
     to error
    end

   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_nivel on nivel for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on nivel */
begin
  declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @insclave numeric(4),
        @errno  int,
        @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* nivel de educacion del programa ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update programa
    set
     /* %JoinFKPK(programa,@ins," = ",",") */
     programa.nivel = @insclave
    from programa,inserted,deleted
    where
     /* %JoinFKPK(programa,deleted," = "," and") */
     programa.nivel = deleted.clave
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "nivel" UPDATE because
more than one row has been affected.'
     raiserror @errno @errmsg
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tI_obs_prof on obs_prof for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* INSERT trigger on obs_prof */
begin
  declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @errno  int,
        @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* profesor se le hacen obs_prof ON CHILD INSERT
RESTRICT */
  if
   /* %ChildFK(" or",update) */
   update(profesor)
  begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,profesor
     where
      /* %JoinFKPK(inserted,profesor) */
      inserted.profesor = profesor.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
```

```
    select @errno  = 30002,
           @errmsg = 'Cannot INSERT "obs_prof" because
"profesor" does not exist.'
    to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_obs_prof on obs_prof for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on obs_prof */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* profesor se le hacen obs_prof ON CHILD UPDATE
 RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(profesor)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,profesor
        where
          /* %JoinFKPK(inserted,profesor) */
          inserted.profesor = profesor.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "obs_prof" because
 "profesor" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tI_observacion on observacion for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* INSERT trigger on observacion */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* cateria tiene una observacion ON CHILD INSERT
 RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cateria)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,cateria
        where
          /* %JoinFKPK(inserted,cateria) */
          inserted.cateria = cateria.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "observacion" because
"cateria" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* alumno se le realizan observacion ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cuenta)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno
        where
          /* %JoinFKPK(inserted,alumno) */
          inserted.cuenta = alumno.cuenta
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'Cannot INSERT "observacion" because
 "alumno" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_observacion on observacion for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* UPDATE trigger on observacion */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(16),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* cateria tiene una observacion ON CHILD UPDATE
 RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cateria)
  begin
    select @nullcnt = 0
```

```
   select @validcnt = count(*)
     from inserted,cateria
       where
         /* %JoinFKPK(inserted,cateria) */
         inserted.cateria = cateria.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
     select @errno  = 30007,
          @errmsg = 'Cannot UPDATE "observacion" because
"cateria" does not exist.'
     to error
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* alumno se le realizan observacion ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(cuenta)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,alumno
        where
          /* %JoinFKPK(inserted,alumno) */
          inserted.cuenta = alumno.cuenta
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
     select @errno  = 30007,
          @errmsg = 'Cannot UPDATE "observacion" because
"alumno" does not exist.'
     to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_periodo on periodo for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:07 1997 */
/* DELETE trigger on periodo */
begin
  declare  @errno  int,
           @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   /* periodo termino de cursar el alumno_plan ON PARENT
DELETE RESTRICT */
   if exists (
     select * from deleted,alumno_plan
     where
       /* %JoinFKPK(alumno_plan,deleted," = "," and") */
       alumno_plan.termino = deleted.clave
   )
   begin
     select @errno  = 30001,
          @errmsg = 'Cannot DELETE "periodo" because
"alumno_plan" exists.'
     to error
   end
   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   /* periodo empezo a cursar el alumno_plan ON PARENT
DELETE RESTRICT */
   if exists (
     select * from deleted,alumno_plan
     where
       /* %JoinFKPK(alumno_plan,deleted," = "," and") */
       alumno_plan.inicio = deleted.clave
   )
   begin
     select @errno  = 30001,
          @errmsg = 'Cannot DELETE "periodo" because
"alumno_plan" exists.'
     to error
   end
   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   /* periodo hay horario_atencion ON PARENT DELETE
CASCADE */
   delete horario_atencion
     from horario_atencion,deleted
     where
       /* %JoinFKPK(horario_atencion,deleted," = "," and") */
       horario_atencion.periodo = deleted.clave
   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   /* periodo se imparte en cierto grupo ON PARENT DELETE
RESTRICT */
   if exists (
     select * from deleted,grupo
     where
       /* %JoinFKPK(grupo,deleted," = "," and") */
       grupo.periodo = deleted.clave
   )
   begin
     select @errno  = 30001,
          @errmsg = 'Cannot DELETE "periodo" because "grupo"
exists.'
     to error
   end

   /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_periodo on periodo for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on periodo */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(8),
           @errno  int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:07 1997 */
  /* periodo termino de cursar el alumno_plan ON PARENT
UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update alumno_plan
      set
        /* %JoinFKPK(alumno_plan,@ins," = ",",") */
        alumno_plan.termino = @insclave
```

```
    from alumno_plan,inserted,deleted
    where
     /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
     alumno_plan.termino = deleted.clave
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "periodo" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* periodo empezo a cursar el alumno_plan ON PARENT
UPDATE CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update alumno_plan
    set
     /*  %JoinFKPK(alumno_plan,@ins," = ",",") */
     alumno_plan.inicio = @insclave
    from alumno_plan,inserted,deleted
    where
     /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
     alumno_plan.inicio = deleted.clave
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "periodo" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* periodo hay horario_atencion ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update horario_atencion
    set
     /*  %JoinFKPK(horario_atencion,@ins," = ",",") */
     horario_atencion.periodo = @insclave
    from horario_atencion,inserted,deleted
    where
     /*  %JoinFKPK(horario_atencion,deleted," = "," and") */
     horario_atencion.periodo = deleted.clave
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "periodo" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* periodo se imparte en cierto grupo ON PARENT UPDATE
CASCADE */
  if
   /* %ParentPK(" or",update) */
   update(clave)
  begin
   if @numrows = 1
   begin
    select @insclave = inserted.clave
     from inserted
    update grupo
    set
     /*  %JoinFKPK(grupo,@ins," = ",",") */
     grupo.periodo = @insclave
    from grupo,inserted,deleted
    where
     /*  %JoinFKPK(grupo,deleted," = "," and") */
     grupo.periodo = deleted.clave
   end
   else
   begin
    select @errno = 30006,
        @errmsg = 'Cannot cascade "periodo" UPDATE
because more than one row has been affected.'
    raiserror @errno @errmsg
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tD_profesor on profesor for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* DELETE trigger on profesor */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
   /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
   /* profesor se le hacen obs_prof ON PARENT DELETE
CASCADE */
   delete obs_prof
    from obs_prof,deleted
    where
     /*  %JoinFKPK(obs_prof,deleted," = "," and") */
     obs_prof.profesor = deleted.clave
   /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
   /* profesor da clases a un profesor_grupo ON PARENT
DELETE RESTRICT */
   if exists (
    select * from deleted,profesor_grupo
    where
     /*  %JoinFKPK(profesor_grupo,deleted," = "," and") */
     profesor_grupo.profesor = deleted.clave
   )
   begin
    select @errno  = 30001,
        @errmsg = 'Cannot DELETE "profesor" because
"profesor_grupo" exists.'
    to error
   end

   /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end
```

```
create trigger tI_profesor on profesor for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* INSERT trigger on profesor */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* grado es alcanzado por profesor ON CHILD INSERT SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(grado)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.grado = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from grado
          where
            /* %JoinFKPK(inserted,grado," = "," and") */
            inserted.grado = grado.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* civil se encuentra en cierto profesor ON CHILD INSERT
SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(edo_civil)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.edo_civil = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from civil
          where
            /* %JoinFKPK(inserted,civil," = "," and") */
            inserted.edo_civil = civil.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* municipio vive en un profesor ON CHILD INSERT SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(municipio)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.municipio = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from municipio
          where
            /* %JoinFKPK(inserted,municipio," = "," and") */
            inserted.municipio = municipio.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* municipio trabaja en un profesor ON CHILD INSERT SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(trab_municipio)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.trab_municipio = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from municipio
          where
            /* %JoinFKPK(inserted,municipio," = "," and") */
            inserted.trab_municipio = municipio.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* tipo_profesor es de cierto profesor ON CHILD INSERT SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(tipo)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.tipo = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from tipo_profesor
          where
            /* %JoinFKPK(inserted,tipo_profesor," = "," and") */
            inserted.tipo = tipo_profesor.clave
        )
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
  raiserror @errno @errmsg
  rollback transaction
end

create trigger tU_profesor on profesor for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on profesor */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(8),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
```

```
  /* profesor se le hacen obs_prof ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update obs_prof
      set
        /*  %JoinFKPK(obs_prof,@ins," = ",",") */
        obs_prof.profesor = @insclave
      from obs_prof,inserted,deleted
      where
        /*  %JoinFKPK(obs_prof,deleted," = "," and") */
        obs_prof.profesor = deleted.clave
    end
    else
    begin
      select @errno = 30006,
          @errmsg = 'Cannot cascade "profesor" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* profesor da clases a un profesor_grupo ON PARENT
UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update profesor_grupo
      set
        /*  %JoinFKPK(profesor_grupo,@ins," = ",",") */
        profesor_grupo.profesor = @insclave
      from profesor_grupo,inserted,deleted
      where
        /*  %JoinFKPK(profesor_grupo,deleted," = "," and") */
        profesor_grupo.profesor = deleted.clave
    end
    else
    begin
      select @errno = 30006,
          @errmsg = 'Cannot cascade "profesor" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* grado es alcanzado por profesor ON CHILD UPDATE SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(grado)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.grado = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from grado
          where
            /* %JoinFKPK(inserted,grado," = "," and") */
            inserted.grado = grado.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* civil se encuentra en cierto profesor ON CHILD UPDATE
SET NULL */
  if
    /* %ChildFK(" or",update) */
    update(edo_civil)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.edo_civil = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from civil
          where
            /* %JoinFKPK(inserted,civil," = "," and") */
            inserted.edo_civil = civil.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* municipio vive en un profesor ON CHILD UPDATE SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(municipio)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.municipio = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from municipio
          where
            /* %JoinFKPK(inserted,municipio," = "," and") */
            inserted.municipio = municipio.clave
        )
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* municipio trabaja en un profesor ON CHILD UPDATE SET
NULL */
  if
    /* %ChildFK(" or",update) */
    update(trab_municipio)
  begin
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.trab_municipio = NULL
      from profesor,inserted
      where
        /* %JoinPKPK(profesor,inserted," = "," and") */
        profesor.clave = inserted.clave and
        not exists (
          select * from municipio
          where
            /* %JoinFKPK(inserted,municipio," = "," and") */
            inserted.trab_municipio = municipio.clave
        )
```

```
 end
 /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
 /* tipo_profesor es de cierto profesor ON CHILD UPDATE SET
NULL */
 if
  /* %ChildFK(" or",update) */
  update(tipo)
 begin
  update profesor
   set
    /* %SetFK(profesor,NULL) */
    profesor.tipo = NULL
   from profesor,inserted
   where
    /* %JoinPKPK(profesor,inserted," = "," and") */
    profesor.clave = inserted.clave and
    not exists (
     select * from tipo_profesor
     where
      /* %JoinFKPK(inserted,tipo_profesor," = "," and") */
      inserted.tipo = tipo_profesor.clave
    )
 end

 /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tI_profesor_grupo on profesor_grupo for INSERT
as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* INSERT trigger on profesor_grupo */
begin
 declare  @numrows int,
          @nullcnt int,
          @validcnt int,
          @errno   int,
          @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
 /* profesor da clases a un profesor_grupo ON CHILD INSERT
RESTRICT */
 if
   /* %ChildFK(" or",update) */
   update(profesor)
 begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,profesor
     where
      /* %JoinFKPK(inserted,profesor) */
      inserted.profesor = profesor.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30002,
           @errmsg = 'Cannot INSERT "profesor_grupo" because
"profesor" does not exist.'
     to error
   end
 end
 /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
 /* grupo le imparte la clase un profesor_grupo ON CHILD
INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(grupo)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,grupo
    where
     /* %JoinFKPK(inserted,grupo) */
     inserted.grupo = grupo.clave
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
          @errmsg = 'Cannot INSERT "profesor_grupo" because
"grupo" does not exist.'
    to error
  end
 end

 /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_profesor_grupo on profesor_grupo for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on profesor_grupo */
begin
 declare  @numrows int,
          @nullcnt int,
          @validcnt int,
          @insgrupo numeric(16),
          @insprofesor numeric(8),
          @errno   int,
          @errmsg  varchar(255)
 select @numrows = @@rowcount
 /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
 /* profesor da clases a un profesor_grupo ON CHILD UPDATE
RESTRICT */
 if
   /* %ChildFK(" or",update) */
   update(profesor)
 begin
   select @nullcnt = 0
   select @validcnt = count(*)
    from inserted,profesor
     where
      /* %JoinFKPK(inserted,profesor) */
      inserted.profesor = profesor.clave
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
    select @errno  = 30007,
           @errmsg = 'Cannot UPDATE "profesor_grupo" because
"profesor" does not exist.'
     to error
   end
 end
 /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
 /* grupo le imparte la clase un profesor_grupo ON CHILD
UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(grupo)
 begin
```

```
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,grupo
        where
          /* %JoinFKPK(inserted,grupo) */
          inserted.grupo = grupo.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'Cannot UPDATE "profesor_grupo" because
"grupo" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tD_programa on programa for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* DELETE trigger on programa */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* programa tiene alumno_plan ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,alumno_plan
      where
        /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
        alumno_plan.programa = deleted.clave
    )
    begin
      select @errno  = 30001,
            @errmsg = 'Cannot DELETE "programa" because
"alumno_plan" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* programa consta de bloque ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,bloque
      where
        /*  %JoinFKPK(bloque,deleted," = "," and") */
        bloque.programa = deleted.clave
    )
    begin
      select @errno  = 30001,
            @errmsg = 'Cannot DELETE "programa" because
"bloque" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* programa define varias asignatura_plan ON PARENT
DELETE RESTRICT */
    if exists (
      select * from deleted,asignatura_plan
      where
        /*  %JoinFKPK(asignatura_plan,deleted," = "," and") */
        asignatura_plan.programa = deleted.clave
    )
    begin
      select @errno  = 30001,
            @errmsg = 'Cannot DELETE "programa" because
"asignatura_plan" exists.'
      to error
    end

    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tI_programa on programa for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* INSERT trigger on programa */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* nivel de educacion del programa ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(nivel)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,nivel
        where
          /* %JoinFKPK(inserted,nivel) */
          inserted.nivel = nivel.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
            @errmsg = 'Cannot INSERT "programa" because
"nivel" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* carrera consta de varios programa ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(carrera)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,carrera
        where
          /* %JoinFKPK(inserted,carrera) */
          inserted.carrera = carrera.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
            @errmsg = 'Cannot INSERT "programa" because
"carrera" does not exist.'
      to error
    end
  end
```

```
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger tU_programa on programa for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on programa */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(8),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* programa tiene alumno_plan ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update alumno_plan
      set
        /*  %JoinFKPK(alumno_plan,@ins," = ",",") */
        alumno_plan.programa = @insclave
      from alumno_plan,inserted,deleted
      where
        /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
        alumno_plan.programa = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "programa" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* programa consta de bloque ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update bloque
      set
        /*  %JoinFKPK(bloque,@ins," = ",",") */
        bloque.programa = @insclave
      from bloque,inserted,deleted
      where
        /*  %JoinFKPK(bloque,deleted," = "," and") */
        bloque.programa = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "programa" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* programa define varias asignatura_plan ON PARENT
UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update asignatura_plan
      set
        /*  %JoinFKPK(asignatura_plan,@ins," = ",",") */
        asignatura_plan.programa = @insclave
      from asignatura_plan,inserted,deleted
      where
        /*  %JoinFKPK(asignatura_plan,deleted," = "," and") */
        asignatura_plan.programa = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "programa" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* nivel de educacion del programa ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(nivel)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,nivel
        where
          /* %JoinFKPK(inserted,nivel) */
          inserted.nivel = nivel.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
             @errmsg = 'Cannot UPDATE "programa" because
"nivel" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* carrera consta de varios programa ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(carrera)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,carrera
        where
          /* %JoinFKPK(inserted,carrera) */
          inserted.carrera = carrera.clave
```

```
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'Cannot UPDATE "programa" because
"carrera" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tI_s_asignatura on s_asignatura for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* INSERT trigger on s_asignatura */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* asignatura_plan es antecedente de s_asignatura ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(antecedente)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura_plan
        where
          /* %JoinFKPK(inserted,asignatura_plan) */
          inserted.antecedente = asignatura_plan.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
            @errmsg = 'Cannot INSERT "s_asignatura" because
"asignatura_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* asignatura_plan es consecuente de s_asignatura ON CHILD
INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(consecuente)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura_plan
        where
          /* %JoinFKPK(inserted,asignatura_plan) */
          inserted.consecuente = asignatura_plan.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
            @errmsg = 'Cannot INSERT "s_asignatura" because
"asignatura_plan" does not exist.'
      to error
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_s_asignatura on s_asignatura for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on s_asignatura */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insantecedente numeric(8),
           @insconsecuente numeric(8),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* asignatura_plan es antecedente de s_asignatura ON CHILD
UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(antecedente)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura_plan
        where
          /* %JoinFKPK(inserted,asignatura_plan) */
          inserted.antecedente = asignatura_plan.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'Cannot UPDATE "s_asignatura" because
"asignatura_plan" does not exist.'
      to error
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* asignatura_plan es consecuente de s_asignatura ON CHILD
UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(consecuente)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,asignatura_plan
        where
          /* %JoinFKPK(inserted,asignatura_plan) */
          inserted.consecuente = asignatura_plan.clave
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'Cannot UPDATE "s_asignatura" because
"asignatura_plan" does not exist.'
```

```
    to error
   end
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end

create trigger ti_s_bloque on s_bloque for INSERT as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* INSERT trigger on s_bloque */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* bloque es antecedente de s_bloque ON CHILD INSERT
 RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(antecedente)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,bloque
        where
          /* %JoinFKPK(inserted,bloque) */
          inserted.antecedente = bloque.clave
     /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

     if @validcnt + @nullcnt != @numrows
     begin
       select @errno  = 30002,
              @errmsg = 'Cannot INSERT "s_bloque" because
 "bloque" does not exist.'
       to error
     end
    end
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* bloque es consecuente de s_bloque ON CHILD INSERT
 RESTRICT */
    if
      /* %ChildFK(" or",update) */
      update(consecuente)
    begin
      select @nullcnt = 0
      select @validcnt = count(*)
        from inserted,bloque
          where
            /* %JoinFKPK(inserted,bloque) */
            inserted.consecuente = bloque.clave
      /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
  from inserted where"," and") */

      if @validcnt + @nullcnt != @numrows
      begin
        select @errno  = 30002,
               @errmsg = 'Cannot INSERT "s_bloque" because
  "bloque" does not exist.'
        to error
      end
     end

     /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
     return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_s_bloque on s_bloque for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on s_bloque */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insantecedente numeric(8),
           @insconsecuente numeric(8),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* bloque es antecedente de s_bloque ON CHILD UPDATE
 RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(antecedente)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,bloque
        where
          /* %JoinFKPK(inserted,bloque) */
          inserted.antecedente = bloque.clave
     /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
 from inserted where"," and") */

     if @validcnt + @nullcnt != @numrows
     begin
       select @errno  = 30007,
              @errmsg = 'Cannot UPDATE "s_bloque" because
 "bloque" does not exist.'
       to error
     end
    end
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* bloque es consecuente de s_bloque ON CHILD UPDATE
 RESTRICT */
    if
      /* %ChildFK(" or",update) */
      update(consecuente)
    begin
      select @nullcnt = 0
      select @validcnt = count(*)
        from inserted,bloque
          where
            /* %JoinFKPK(inserted,bloque) */
            inserted.consecuente = bloque.clave
      /* %NotnullFK(inserted," is null","select @nullcnt = count(*)
  from inserted where"," and") */

      if @validcnt + @nullcnt != @numrows
      begin
        select @errno  = 30007,
               @errmsg = 'Cannot UPDATE "s_bloque" because
  "bloque" does not exist.'
        to error
      end
     end

     /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
     return
error:
     raiserror @errno @errmsg
```

```
   rollback transaction
end

create trigger tD_tipo_asignatura on tipo_asignatura for
DELETE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* DELETE trigger on tipo_asignatura */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* tipo_asignatura contiene una o mas modalidad_tipo ON
PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,modalidad_tipo
      where
        /*  %JoinFKPK(modalidad_tipo,deleted," = " and") */
        modalidad_tipo.tipo = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "tipo_asignatura" because
"modalidad_tipo" exists.'
      to error
    end
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* tipo_asignatura es de cierto asignatura_plan ON PARENT
DELETE RESTRICT */
    if exists (
      select * from deleted,asignatura_plan
      where
        /*  %JoinFKPK(asignatura_plan,deleted," = " and") */
        asignatura_plan.tipo = deleted.clave
    )
    begin
      select @errno  = 30001,
             @errmsg = 'Cannot DELETE "tipo_asignatura" because
"asignatura_plan" exists.'
      to error
    end

    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_tipo_asignatura on tipo_asignatura for
UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on tipo_asignatura */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* tipo_asignatura contiene una o mas modalidad_tipo ON
PARENT UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update modalidad_tipo
      set
        /*  %JoinFKPK(modalidad_tipo,@ins," = ",",") */
        modalidad_tipo.tipo = @insclave
      from modalidad_tipo,inserted,deleted
      where
        /*  %JoinFKPK(modalidad_tipo,deleted," = " and") */
        modalidad_tipo.tipo = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "tipo_asignatura" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* tipo_asignatura es de cierto asignatura_plan ON PARENT
UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update asignatura_plan
      set
        /*  %JoinFKPK(asignatura_plan,@ins," = ",",") */
        asignatura_plan.tipo = @insclave
      from asignatura_plan,inserted,deleted
      where
        /*  %JoinFKPK(asignatura_plan,deleted," = " and") */
        asignatura_plan.tipo = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "tipo_asignatura" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tD_tipo_profesor on tipo_profesor for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* DELETE trigger on tipo_profesor */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* tipo_profesor es de cierto profesor ON PARENT DELETE
SET NULL */
    update profesor
      set
        /* %SetFK(profesor,NULL) */
        profesor.tipo = NULL
      from profesor,deleted
      where
        /* %JoinFKPK(profesor,deleted," = " and") */
        profesor.tipo = deleted.clave
```

```
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_tipo_profesor on tipo_profesor for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on tipo_profesor */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* tipo_profesor es de cierto profesor ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update profesor
      set
        /*  %JoinFKPK(profesor,@ins," = ","") */
        profesor.tipo = @insclave
      from profesor,inserted,deleted
      where
        /*  %JoinFKPK(profesor,deleted," = "," and") */
        profesor.tipo = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "tipo_profesor" UPDATE
because more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tD_turno on turno for DELETE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* DELETE trigger on turno */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* turno asiste en un alumno_plan ON PARENT DELETE
SET NULL */
    update alumno_plan
      set
        /* %SetFK(alumno_plan,NULL) */
        alumno_plan.turno = NULL
      from alumno_plan,deleted
      where
        /* %JoinFKPK(alumno_plan,deleted," = "," and") */
        alumno_plan.turno = deleted.clave
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* turno se le atiende en algunos horario_atencion ON
PARENT DELETE SET NULL */
    update horario_atencion
      set
        /* %SetFK(horario_atencion,NULL) */
        horario_atencion.turno = NULL
      from horario_atencion,deleted
      where
        /* %JoinFKPK(horario_atencion,deleted," = "," and") */
        horario_atencion.turno = deleted.clave
    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    /* turno le corresponde cierto grupo ON PARENT DELETE
SET NULL */
    update grupo
      set
        /* %SetFK(grupo,NULL) */
        grupo.turno = NULL
      from grupo,deleted
      where
        /* %JoinFKPK(grupo,deleted," = "," and") */
        grupo.turno = deleted.clave

    /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end

create trigger tU_turno on turno for UPDATE as
/* ERwin Builtin Thu Aug 14 21:30:08 1997 */
/* UPDATE trigger on turno */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insclave numeric(4),
           @errno   int,
           @errmsg  varchar(255)
  select @numrows = @@rowcount
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* turno asiste en un alumno_plan ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update alumno_plan
      set
        /*  %JoinFKPK(alumno_plan,@ins," = ","") */
        alumno_plan.turno = @insclave
      from alumno_plan,inserted,deleted
      where
        /*  %JoinFKPK(alumno_plan,deleted," = "," and") */
        alumno_plan.turno = deleted.clave
    end
    else
    begin
      select @errno = 30006,
             @errmsg = 'Cannot cascade "turno" UPDATE because
more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
```

```
  /* turno se le atiende en algunos horario_atencion ON
 PARENT UPDATE CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update horario_atencion
      set
        /*  %JoinFKPK(horario_atencion,@ins," = ",",") */
        horario_atencion.turno = @insclave
      from horario_atencion,inserted,deleted
      where
        /*  %JoinFKPK(horario_atencion,deleted," = "," and") */
        horario_atencion.turno = deleted.clave
    end
    else
    begin
      select @errno = 30006,
            @errmsg = 'Cannot cascade "turno" UPDATE because
more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end
  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  /* turno le corresponde cierto grupo ON PARENT UPDATE
CASCADE */
  if
    /* %ParentPK(" or",update) */
    update(clave)
  begin
    if @numrows = 1
    begin
      select @insclave = inserted.clave
        from inserted
      update grupo
      set
        /*  %JoinFKPK(grupo,@ins," = ",",") */
        grupo.turno = @insclave
      from grupo,inserted,deleted
      where
        /*  %JoinFKPK(grupo,deleted," = "," and") */
        grupo.turno = deleted.clave
    end
    else
    begin
      select @errno = 30006,
            @errmsg = 'Cannot cascade "turno" UPDATE because
more than one row has been affected.'
      raiserror @errno @errmsg
    end
  end

  /* ERwin Builtin Thu Aug 14 21:30:08 1997 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end
```