UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

" ACCESO A INFORMACION DISTRIBUIDA PARA EL CONTROL DE BECARIOS "

FACULTAD DE CIENCIAS
UNAM

T E S I S

QUE PARA OBTENER EL TITULO DE:

ACTUARIA

P R E S E N T A :

CLAUDIA MARIA TERESA AGUILA MARTINEZ

DIRECTOR DE TESIS:
M. en C. GUADALUPE IBARGÜENGOITIA GONZALEZ

MEXICO, D. F.

1998

VNIVERSIDAD NACIONAL
AVFNºMA DE
MEXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
P r e s e n t e

Comunicamos a usted que hemos revisado el trabajo de Tesis: "ACCESO A INFORMACION DISTRIBUIDA PARA EL CONTROL DE BECARIOS."

realizado por CLAUDIA MARIA TERESA AGUILA MARTINEZ.

con número de cuenta 8624538-0 , pasante de la carrera de ACTUARIA.

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis
Propietario M. en C. GUADALUPE IBARGUENGOITIA GONZALEZ.

Propietario DR. ENRIQUE RUIZ VELASCO SANCHEZ.

Propietario M. en C. AARON APARICIO HERNANDEZ.

Suplente MAT. MA. de los ANGELES BAUTISTA ZAMUDIO.

Suplente MAT. ANA LUISA SOLIS GONZALEZ COSIO.

Consejo Departamental de Matemáticas
M. en A.P. MA. PILAR ALONSO REYES.

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE

Dedico esté trabajo a:

A mis padres, Macedonia y Adrian quienes supieron darme las bases de la vida.

A mis hermanos, Marco Antonio, Gloria y Adrian por su ayuda incondicional.

A mi esposo, Marco Antonio Rosas por su comprensión y paciencia.

Dedico este trabajo a todos aquellos que me alentaron a lo largo de mi carrera.

La vida es una oportunidad, aprovechala:

La vida es belleza, admírala.

La vida es dicha, saboréala.

La vida es un sueño, hazlo realidad.

La vida es un reto, afróntalo.

La vida es la VIDA,idefiéndela!

Madre Teresa de Calcuta

AGRADECIMIENTOS.

Mi especial agradecimiento a:

Mat. José Antonio López Saucedo y a su equipo de trabajo por el apoyo que en su oportunidad me brindaron.

Prof. Juan Manuel Rodriguez y a su personal por la colaboración para la realización de este sistema.

M. en C. Lupita Ibargüengoitia por su valiosa enseñanza.

Mis padres, hermanos y esposo, por su ayuda, comprensión y paciencia.

Todas aquellas personas que de una u otra forma me impulsaron y cooperaron para que terminara este trabajo.

A Dios por darme la oportunidad de continuar con vida.

Claudia
México, D.F.

ÍNDICE

INTRODUCCIÓN

INTRODUCCIÓN

En todas las empresas, instituciones y comercios se archiva información de distinta índole y se tiene una contínua consulta de ella, por lo que al desear determinados datos resulta complicado tener que consultarlos y seleccionar archivos o expedientes. Una alternativa para manejar información importante son las Bases de Dátos.

Para cualquier diciplina o actividad en general y en lo particular para la actuaría, una Base de Datos es de gran utilidad debido a la información registrada, por ejemplo de ahí se pueden obtener las estadísticas necesarias para algún estudio específico (proporcionando las frecuencias deseadas, según sea el caso, suceso o personas).

La ingeniería del software es una diciplina que integra los métodos, procedimientos y herramientas. Los cuales han sido aplicados con éxito; dando como resultado una mejor calidad del software. El producto final resulta ser "amigable a los humanos", esto marca la diferencia con otros productos con funciones similares.

En el presente trabajo se hace referencia al entorno de sistemas computacionales con los cuales se puede desarrollar un sistema de información.

Mencionaremos el modelo relacional como uno de los más utilizados en las redes de computación, además las características de los manejadores de las Bases de Datos, en particular de INFORMIX-SQL.

Como una aplicación de la automatización de información, se desarrolla un sistema para el control de becarios, el cual sigue un proceso de acuerdo al modelo relacional. Siguiendo las fases de vida de un sistema y buscando que sea óptimo. Utilizando los paradigmas de la ingeniería del software.

Es así que en el capítulo 1, se mencionan los conceptos básicos de la computación e ingeniería del software.

En el capítulo 2, se explican los Sistemas Manejadores de Bases de Datos, además se desglosan los modelos de bases de datos en específico el relacional y lo concerniente al supervisor del software.

El capítulo 3, presenta la aplicación de todo lo anterior en un sistema único que se diseñó con el fin de controlar los datos requeridos para los becarios. Así como su desarrollo y especificaciones de elaboración, además de la explicación de uso.

1

SISTEMAS COMPUTACIONALES

1.1 LA COMPUTACIÓN.

La evolución tecnológica, cuyos orígenes se remontan a los albores de la historia humana, ha conocido, a partir de la década de los cuarentas; un impulso extraordinario gracias al diseño y la popularidad progresiva de unas máquinas llamadas *computadoras* u *ordenadores*, según el área de influencia anglosajona o francesa respectivamente.

La *computadora* ha sido definida como una máquina capaz de realizar y controlar a gran velocidad cálculos y procesos complicados que requieren una toma rápida de decisiones.

La *computación* ó *informática* es un conjunto de conocimientos científicos y técnicos que hacen posible el tratamiento automatizado de la información por medio de computadoras.

Una *computadora* es una máquina capaz de trabajar con símbolos, independientemente del significado de estos, pues puede manejar números, textos, archivos, imágenes y sonidos. siempre que se le codifique según un formato interno con el que la máquina sea capaz de trabajar. Una computadora actual emplea la electricidad como medio para representar la información, mediante una codificación digital binaria.

Una computadora es una herramienta utilizada por los humanos en el proceso de solución de problemas. La computadora extiende las habilidades humanas debido a las características siguientes: Velocidad de operación, capacidad de memoria, puntualidad y costo-efectividad en muchas aplicaciones.

Sin un *software*, una computadora es básicamente un conjunto de metal(*hardware*) inservible. El *software*, es un conjunto de programas utilizables en una clase de computadoras, junto con la documentación asociada a la computadora o los programas, tales como manuales, diagramas, instrucciones de funcionamiento. El más importante de todos los programas es el *sistema operativo* el cual controla todos los recursos de la computadora y de la base para que los programas de aplicación puedan ser escritos.

1.2 HARDWARE.

HARDWARE: Palabra inglesa que significa "material de ferretería", se emplea para designar a la parte física de la computadora, esto es, el conjunto de circuitos electrónicos y dispositivos mecánicos que, actuando conjuntamente bajo la dirección del *software*, realiza el manejo y almacenamiento de la información [1].

Un sistema de computo incluye un número de dispositivos separados funcionalmente que constituyen su *hardware*: El procesador central, el sistema de memoria y los mecanismos de entrada y salida. La unidad central es el conjunto de circuitos que gobiernan el funcionamiento de toda la computadora y el lugar donde se realizan las operaciones sobre los datos a procesar. Los dispositivos periféricos(impresora, ratón, monitor, teclado, terminales, disco duro y flexible.) se encargan de recoger los datos, almacenarlos y suministrar los resultados al usuario o a otras máquinas.

La unidad central de la computadora consta de tres tipos de componentes: El procesador, la memoria central y los circuitos de interfaz.

El procesador es el conjunto de circuitos que controlan el funcionamiento de la computadora y realiza las operaciones con los datos.

La memoria central, está compuesta por un conjunto de circuitos que adoptan unos valores de tensión equivalentes a la representación interna de los datos y las instrucciones que el procesador está utilizando en un momento dado.

Los circuitos de interfaz, o simplemente interfaz, son los circuitos que permiten al procesador tomar o suministrar información a los dispositivos periféricos, dicho de otra manera constituyen el medio físico y lógico común y necesario de dos sistemas para intercambiar comunicación.

1.3 SOFTWARE.

SOFTWARE: Es un conjunto de programas, códigos y convenciones necesarias para la realización de una tarea por el mecanismo de la computadora. El cual puede ser de aplicación o de utilidad. El *software* de aplicación es aquel que sirve para una tarea determinada, sea educativa, científica, cálculos númericos, almacenamiento y extracción de información, manejo de conjunto de datos, procesamiento de la palabra, comunicación remota entre los usuarios. El *software* de utilidad es aquel que tiene como finalidad la ayuda a la creación de otros programas, como en el caso de los lenguajes de programación o de los sistemas operativos [1].

Cuando la librería de programas de una computadora contiene el *software* necesario para realizar una cierta tarea, decimos que la aplicación está implementada en tal computadora. Es decir que cuenta con un conjunto de información organizada como referencia.

Las computadoras actuales almacenan el *software* en el mismo espacio de memoria que recibe también a los datos, sin ninguna distinción ni separación física entre ambos. Es el propio *software* el que se encarga de que la computadora busque cada tipo de información en el lugar adecuado.

Dentro del *software* podemos distinguir entre algoritmos y estructuras de datos. Los algoritmos son los conjuntos de instruciones que nos permiten realizar un trabajo determinado; su uso en las computadoras exige su escritura en forma de programas. Las estrcturas de datos, son agrupaciones estándar de datos que nos ofrecen los lenguajes de programación, con el fin de facilitar la referencia a una información determinada dentro de un programa.

Programa: Es un conjunto de instrucciones que, ejecutadas en un cierto orden, indican a la computadora las operaciones que deben realizar con los datos para obtener el resultado deseado.

El *Sistema Operativo* tiene como principal tarea llevar un registro de la utilización de los recursos, dar paso a las solicitudes de recursos, llevar la cuenta de su uso y medir entre las solicitudes en conflicto de los distintos programas y usuarios. Proporciona al usuario un interfaz.

Dentro de los sistemas operativos los más importantes y usados son MS-DOS y UNIX. En este caso vamos a referirnos al sistema operativo UNIX por las ventajas con que cuenta para la realización del proyecto.

1.4 CARACTERÍSTICAS DE UNIX.

El sistema operativo UNIX tiene importantes características en la computación como son: las habilidades de multiusuario, multitarea e independencia de dispositivos de la computadora [2].

HABILIDADES MULTIUSUARIO.

A diferencia de otros sistemas tales como MS-DOS y OS/2, UNIX permite que más de una persona use la computadora al mismo tiempo. Cada uno de los usuarios pueden desarrollar una tarea completamente diferente, donde aparentemente tienen la atención individual de la computadora.

Aunque la computadora parece estar dedicando todo el tiempo a cada usuario, en realidad sólo pocos milisegundos de tiempo dedica a cada uno y rápidamente activa la atención de un usuario a otro. El proceso es tan rápido que el usuario, no puede darse cuenta que en realidad UNIX lo abandona por un corto tiempo y luego regresa su atención.

Un sistema operativo multiusuario tal como UNIX ofrece varias ventajas sobre un sistema uni-usuario (computadoras personales):

- Costo por usuario.

En un sistema multiusuario cada usuario sólo necesita una terminal, la cual cuesta menos que una computadora. Estas terminales están conectadas a una sola computadora. Aunque la computadora UNIX es más cara que una simple PC, está ofrece menos costo por persona.

- Administración central.

Las compañías, instituciones, empresas y comercios; al emplear un sistema operativo UNIX pueden emplear menos personal para realizar tareas rutinarias como son: respaldos y mantenimiento, las cuales pueden ser realizadas desde una locación central.

-Recursos compartidos.

En un sistema multiusuario se pueden compartir información e interactuar con otros usuarios. Como enviar mensajes a través de correo electrónico, intercambiar datos y compartir dispositivos como modems, impresoras, faxes, cintas y otros equipos.

El número de usuarios que soporta un sistema varia de acuerdo a la infraestructura del sistema.

HABILIDADES MULTITAREA.

UNIX puede realizar más de una tarea a la vez con cada usuario, a ésto se le llama Multitarea y puede ser de la siguiente forma:

- Procesos de fondo.

Se pueden realizar tareas que no requieren interacción con el usuario tales como formatear un disco o clasificar un archivo de datos. Estas tareas reciben parte de la atención de la computadora al no usar el teclado, el usuario puede hacer otras tareas, quedando las otras de fondo.

SISTEMA ABIERTO Y PORTATIL.

Unas de las características más notables de UNIX es su portabilidad, su capacidad para correr en un rango amplio de modelos y marcas de computadoras.

Al principio la mayoría de los sistemas operativos eran propios, o sea eran escritos para una sola marca de computadoras por ejemplo VMS fue diseñado exclusivamente para la DEC (Digital Equipment Corporation) y MVS fue diseñado para correr en máquinas IBM. Antes de UNIX cada computadora tenía su propio sistema operativo y los programas de aplicación fueron escritos específicamente para cada sistema operativo. Hoy los sistemas UNIX corren en una amplia variedad de computadoras construidas por cuenta de los fabricantes.

UNIX cambió el curso de la computación, esto nos lleva a conocer lo que es un sistema abierto, el cual proporciona grandes beneficios, tales como poder actualizar o instalar un nuevo sistema UNIX sin preocuparse por la marca, número de computadoras o costo. Los usuarios fácilmente pueden convertir datos y *software* al nuevo sistema operativo.

1.5 CONCEPTOS DEL SISTEMA OPERATIVO UNIX.

Existen conceptos fundamentales en el sistema operativo UNIX que son importantes conocer para entender su diseño [3].

PROCESOS: Es un concepto clave, básicamente se trata de un programa de ejecución, el cual maneja las llamadas del sistema.

ARCHIVOS: Es una secuencia de bytes. (Un byte es un pequeño trozo de información, normalmente compuesto por 8 bits. Para nuestro propósito, un byte es equivalente a un carácter) que almacena cierta información. Un arhivo puede contener una carta, una lista de nombres y direcciones, las instrucciones fuente de un programa, datos para ser usados por un programa e incluso programas en su forma ejecutable, así como otro tipo de material que no esté en forma de texto [3].

SHELL: Intérprete de comandos, es un programa que se encuentra entre el usuario y las facilidades del núcleo que tiene varias ventajas, entre las que tenemos:

* Las abreviaturas de nombres de archivos: Se puede tomar todo un conjunto de nombres de archivos como argumentos de un programa especificando el patrón de los nombres; el shell encontrará los nombres de archivos que igualen dicho patrón.

* Redireccionamiento de entrada-salida: Se puede lograr que la salida de cualquier programa se envie hacia un archivo en lugar de dirigirla hacia una terminal; se puede hacer que la entrada provenga también de un archivo y no de la terminal. La entrada y salida pueden conectarse a otros programas.

* Personalizar el entorno: El usuario puede definir sus propios comandos y abreviaturas.

DIRECTORIOS: Los archivos se agrupan en *directorios*, en forma semejante a como se acomodan los libros en estantes en una biblioteca, por lo que archivos en distintos directorios pueden tener el mismo nombre sin ningún conflicto.

Un directorio puede tener otros *subdirectorios*, así como archivos ordinarios. La manera natural de visualizar esta organización es considerarla como un árbol de directorios y archivos. Es posible desplazarse dentro de este árbol y encontrar cualquier archivo en el sistema comenzando en la raíz y moviéndose a través de las ramas adecuadas. En forma inversa, se puede empezar desde donde uno se encuentra y moverse hacia la raíz.

PROGRAMAS DE UTILIDAD: La interfaz de usuarios de UNIX consiste de un gran número de programas de utilidad estandar [2]. Estos programas estan divididos en seis categorias:

1.- Comandos que manejan directorios y archivos.

2.- Filtros.

3.- Herramientas para compilar y desarrollar programas.

4.- Procesador de textos.

5.- Administración del sistema.

6.- Diversos.

1.6 INGENIERÍA DEL SOFTWARE.

Es el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales [4].

La ingeniería del software surge de la ingeniería de sistemas y software. Comprende un conjunto de tres elementos esenciales: Método, herramientas y procedimiento. Esto facilita el desarrollo del software y proporciona las bases para construir software de alta calidad de una forma productiva.

Los métodos nos indican "cómo" construir técnicamente el software, abarca una amplia gama de tareas como: Planificación y estimación de proyectos análisis de los requisitos del sistema y del software, diseño de la estructura de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento. Los métodos de la ingeniería del software introducen frecuentemente una notación especial orientada a un lenguaje o gráfica y un conjunto de criterios para la calidad del sofware.

Las herramientas de la ingeniería del software proporcionan un soporte automático o semiautomático para los métodos, existen herramientas para cada método. Cuando la información creada por una herramienta puede ser usada por otra, se establece un sistema para el soporte del desarrollo del software, llamado ingeniería del software asistida por computadora (del inglés CASE) combina software, hardware y base de datos.

Los procedimientos de la ingeniería del sofware son los que unen a los métodos y herramientas y facilita un desarrollo racional y oportuno del software de computadora. Los procedimientos indican la secuencia en que se aplican los métodos, las entregas que se requieren, los controles que ayudan a asegurar la calidad y coordinar los cambios.

La ingeniería del software está compuesta por fases que comprenden los métodos, herramientas y los procedimientos. Estas fases se ejecutan siguiendo los paradigmas de la ingeniería del software. La elección del paradigma adecuado depende de la naturaleza del proyecto de la aplicación, los métodos y herramientas a usar y los controles y entregas requeridos.

FASES:

Fase de definición: Comienza con la planificación del software. Durante esta etapa se desarrolla una descripción bien definida del software; se analiza el riesgo y los recursos necesarios; se realizan las estimaciones del tiempo y costo.

Después se realiza un análisis y definición de requisitos del software en un esfuerzo conjunto llevado a cabo por el desarrollador del software y el cliente. La especificación de requisitos del software es el documento distribuible que se produce como resultado de esta etapa.

Fase de desarrollo: Traduce un conjunto de requisitos en el elemento operativo del sistema que llamamos software. El diseño del software comienza con la descripción arquitectónica y de datos. Con una estructura modular, se definen las interfaces y se establece la estructura de datos. Se siguen criterios que aseguren la calidad. Produciendo un primer borrador se las especificaciones del diseño, convirtiendose en una parte de la configuración del software.

Una vez terminado el diseño se lleva a cabo la codificación. En cuanto al código, se revisa su estilo y claridad, y se comprueba que haya una correspondencia directa con la descripción detallada del diseño.

Fase de verificación, lanzamiento y mantenimiento: Durante esta fase el ingeniero del software prueba el software a través de una serie de actividades de verificación y validación, para encontrar el mayor número posibles de errores antes

de ponerse en circulación, se verifica el rendimiento funcional de cada módulo para concluir con las pruebas de integración de ellos. Lo prepara para su lanzamiento y lo mantiene a lo largo de toda su vida útil.

1.6.1 PARADIGMAS DE LA INGENIERÍA DEL SOFTWARE.

1.6.1.1 CICLO DE VIDA CLÁSICO DE UN SISTEMA.

El ciclo de vida clásico para la ingeniería del software a veces llamado "modelo en cascada" exige un trato sistemático y secuencial del desarrollo del software, el cual comienza en el nivel del sistema y avanza a través del análisis, diseño, codificación, prueba y mantenimiento.

Ingeniería y análisis del sistema.

Se establecen los requisitos de todos los elementos del sistema y luego se designa un subconjunto de estos requisitos al software; el cual pertenece a un sistema mayor. Este planteamiento del sistema es importante cuando el software se interrrelaciona con otros elementos, tales como el hardware, personas y base de datos. La ingeniería y el análisis del sistema se contemplan los requisitos globales con una pequeña cantidad de análisis y de diseño.

Análisis de los requerimientos del software.

La recopilación de los requisitos se centra e intensifica. Para la construcción de los programas, es necesario comprender los limites del software así como las funciones, el rendimiento y las interfaces requeridas. Los requisitos del sistema, software deben ser documentados y revisados con el cliente.

Diseño.

El diseño del software es un proceso multipaso que considera cuatro atributos del programa: La estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz.

El proceso del diseño traduce los requisitos en una representación del software de tal forma que se obtenga la calidad requerida antes de que comience

la codificación. El diseño se documenta y forma parte de la configuración del software.

Codificación.

La codificación consiste en traducir el diseño en una forma clara para la máquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.

Prueba.

Una vez realizado el código, empieza la prueba del programa, que se centra en la lógica interna del software, garantizando que todas las instrucciones se han probado, asegurando que la entrada definida produce los resultados que se requieren.

Mantenimiento.

El software sufrira cambios después de que se entrega al cliente. Los cambios son debido a posibles errores o por adaptaciones del software a cambios del entorno externo (un nuevo sistema operativo o dispositivos periféricos), o por ampliaciones funcionales o del rendimiento.

Problemas que presenta el ciclo.

Los proyectos reales no siempre siguen el flujo secuencial que propone el modelo, ya que siempre hay iteracciones y esto crea problemas en la aplicación del paradigma.

Se requieren todos los requisitos al principio y tiene dificultades en acomodar posibles incertidumbres que puede existir al comienzo de muchos proyectos.

En las etapas finales del desarrollo del proyecto es donde se detectan posibles errores que pueden ser desastrozos, esto es durante el funcionamiento. Por lo que el cliente debe ser paciente.

1.6.1.2 CONSTRUCCIÓN DE PROTOTIPOS.

Es un paradigma que facilita al programador la creación de un modelo del software a construir. Puede tener cualquiera de las siguientes formas: Un prototipo en papel o un modelo basado en una PC que describa la interacción hombre-máquina; un prototipo que implementa algunos subconjuntos de las funcione requeridas del programa deseado o un programa existente que ejecute parte o toda la función deseada, pero que puede ser mejorado.

Recolección y refinamiento de requisitos.

El técnico y el cliente se reune y definen los objetivos globales para el software, identifica todos los requisitos conocidos y perfilan las áreas donde será necesario una mayor definición.

Diseño rápido.

Representa los aspectos del software visible al usuario (método de entrada y salida), esto nos conduce a la construcción de un prototipo.

Evaluación de un prototipo.

El prototipo es evaluado por el cliente/usuario.

Refinamiento del prototipo.

Refinar los requisitos del software a desarrollar, para que satisfaga las necesidades del cliente.

Ventajas del prototipo.

Facilita una mejor comprensión de lo que se va ha desarrollar, esto proporciona una rápida generación de programas que funcionen.

Problema del paradigma: El prototipo no considera los aspectos de calidad o de mantenimiento del software a largo plazo. Cuando se le informa al cliente

que tiene que ser reconstruido, este solicita que se le apliquen "cuantas mejoras" sean necesarias.

El técnico de desarrollo frecuentemente utiliza un sistema operativo o un lenguaje de programación inapropiados con el fin de obtener un prototipo que funcione rápidamente.

Este paradigma es útil siempre y cuando el técnico y el cliente estén de acuerdo en que el prototipo se construye sólo para definir los requisitos, posteriormente será descartado y se construira el software real, sin perder de vista la calidad y el mantenimiento.

1.6.1.3 EL PARADIGMA EN ESPIRAL.

Ha sido desarrollado para cubrir las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos, añadiendo un nuevo elemento: El análisis de riesgo. Define cuatro actividades principales:

Planificación.

Determinación de objetivos, alternativas y restricciones. Recolección de requisitos y planificación del proyecto inicial. Planificación basada en los comentarios del cliente.

Análisis de riesgo.

Análisis de alternativas e identificación/resolución de riesgos. Análisis de riesgo basados en los requisitos iniciales, reacción del cliente para decidir si se sigue o no.

Ingeniería.

Desarrollo del producto de "siguiente nivel". Prototipo inicial del software, prototipo del siguiente nivel, sistema de ingeniería hacia el producto final.

Ventajas del paradigma.

Valoración de los resultados de ingeniería. Durante la primera vuelta del bucle se definen los objetivos, las alternativas y las restricciones, se analizan e identifican los riesgos. Si el análisis de riesgo indica que hay incertidumbre en los requisitos, se puede usar la creación de prototipos para dar asistencia tanto al técnico como al cliente. En cada bucle la culminación del análisis de riesgo nos indica si se continua o no. El camino de la espiral nos lleva hacia un modelo más completo del sistema operacional.

Problemas del paradigma: Requiere una considerable habilidad para la valoración del riesgo, sino se descubre un riesgo grave, indudablemente surgirá un problema. Pasaran unos cuantos años antes de que se pueda determinar con absoluta certeza la eficacia de este importante nuevo paradigma.

Técnicas de cuarta generación.

Las herramientas del software tienen características del software de alto nivel, también genera automáticamente el código fuente basado en las especificaciones del técnico, lo que nos lleva a construir más rápido el programa.

El paradigma T4G puede incluir todas o algunas de las siguientes herramientas: Lenguaje no procedimental para consulta a base de datos, generación de informes, manipulación de datos, interacción y definición de pantallas, generación de código, facilidades gráficas de alto nivel y facilidades de hoja de cálculo.

Sigue los pasos:

Recolección de requisitos.

Estrategía de diseño.

Se desarrolla software mediante los enfoques convencionales.

Implementación.

Permite centrarse en los resultados deseados, lo que se traduce en un código que produce dichos resultados. Obviamente debe existir una estructura de datos con información relevante y a la cual L4G pueda acceder rápidamente.

Prueba.

Para transformar una implementación L4G en un producto, se debe dirigir una prueba completa.

El uso de L4G indica una reducción de tiempo requerido para la creación del software en aplicaciones pequeñas y de tamaño mediano. Sin embargo para grandes trabajos de desarrollo de software exige el mismo o más tiempo de análisis, diseño y prueba.

Combinación de paradigmas.

Los paradigmas descritos son métodos alternativos para la ingeniería del software. En muchos casos, los paradigmas pueden y deben combinarse en forma que puedan utilizarse las ventajas de cada uno en un único proyecto. En todos los casos:

Recolección preliminar de requisitos: El trabajo comienza con la determinación de los objetivos, alternativas y restricciones.

Si los requisitos son inciertos se puede usar el prototipo para definir mejor los requisitos. Después volver al diseño, codificación y durante la etapa de prueba se puede usar las técnicas de cuarta generación para implementar el sistema durante el paso de codificación del ciclo de vida. También se puede usar L4G junto con el modelo en espiral en los pasos de creación de prototipos o de codificación.

La naturaleza de la aplicación indicará el método a seguir. A través de la combinación de los métodos, el todo puede ser mejor que la suma de las partes.

2

SISTEMAS MANEJADORES DE BASES DE DATOS

2.1 SISTEMA MANEJADORES DE BASES DE DATOS (SMBD).

Un sistema manejador de base de datos, es una colección de programas, los cuales, usan las facilidades del sistema operativo, habilitando a los usuarios para manipular las bases de datos.

Estas herramientas evitan que los usuarios tengan que programar cada vez que necesiten consultar la base de datos, benefician también a los que no tienen conocimientos sobre comandos de UNIX y programación *shell*.

Una base de datos, está formada por ciertas operaciones básicas que son necesarias realizar:

* Crear una estructura donde los datos son almacenados.
* Escribir los datos en esta estructura.
* Modificar datos cuando existan cambios.
* Quitar datos obsoletos.
* Extraer datos en varias combinaciones y formas.
* Asegurar consistencia, integridad y seguridad de datos.

LAS BASES DE DATOS Y SISTEMAS MANEJADORES DE BASES DE DATOS.

Una *Base de Datos* es una colección de datos almacenados. Un SMBD es una colección de datos organizados e interrelacionados para el servicio de una o más aplicaciones en un uso óptimo. La base de datos es diseñada para ser usada por diferentes programas y programadores, en un uso común y controlado, como en el caso de aumentar datos nuevos, modificar y recuperar los ya existentes.

En un principio los datos registrados se procesan en uno o más archivos usados para cada aplicación. El objetivo de la base de datos, es permitir a la misma colección de datos, servir a tantas aplicaciones como sea posible.

La independencia de datos es una de las características más importantes de una SMBD. El dato puede ser fácilmente reorganizado o aumentar su contenido, estructurar datos, distribuir y almacenar físicamente recursos que pueden cambiarse sin tener primero que reescribir los programas de aplicación.

2.2 MODELOS DE LAS BASES DE DATOS.

En un principio el procesamiento de los datos era en archivo, donde cada registro contiene datos de distinto tipo; donde uno o más es nombrado clave y es usada para secuencia de archivos y para localización de registros. Esto puede duplicar información, produciendo inconsistencias debido a la gran cantidad extra de espacio requerido para su almacenamiento.

En muchas bases de datos del mundo, todavía son usados un gran número de archivos los cuales no son planos (un archivo plano es aquel que contiene datos de un mismo tipo). Hay tres formas de almacenar datos que son: El modelo jerárquico, de red y el relacional.

EL MODELO JERÁRQUICO. Una base de datos jerárquica se ve como un árbol invertido, el elemento superior es conocido como raíz(*root*) y los elementos subsecuentes son llamados nodos y cada uno debe tener sólo un nodo relativo superior llamado padre y éste puede tener más de un nodo relativo inferior llamado hijo.

Realiza un acceso rápido para mostrar búsquedas individuales, también para aumentar nuevos registros, pero lento y díficil para mostrar resúmen de la información, para cambiar la estructura de la base de datos y para más relaciones complejas.

EL MODELO DE RED. Los nodos en la base de datos en red, pueden contar con varios nodos hijos y más de un nodo padre. Esto nos proporciona más flexibilidad que el modelo anterior, ya que permite aumentar nuevas relaciones y añadir nodos para acceso de datos, esto nos puede llevar a una gran complejidad, resultando problemas en los programas de aplicación y mantenimiento.

EL MODELO RELACIONAL. En los 70´ s el Dr. Ted Codd trabajó para IBM, donde empezó la aplicación de una rama de las matemáticas llamada *Teoría Relacional*, para almacenar datos en una computadora.

Cualquier representación de datos (en red o jerárquica) puede ser resumida en una forma tabular (tabla) combinando una o más dimensiones. La tabla es una referencia matemática como una relación y una base de datos construída a través de relaciones, es conocida como una base de datos relacional.

2.3 CODD Y LAS BASES DE DATOS RELACIONALES.

Codd [6] es el creador de las bases de datos relacionales. Actualmente multitud de bases de datos se distribuyen con el apellido *relacional*, analizando el modelo. En 1985 en un artículo publicado por la revista *Computerworld*, Codd daba 12 reglas que indican si una base de datos se debe considerar relacional:

1) Regla de información.

Toda la información de una base de datos, está representada en una y sólamente una forma, por valores en la posición de columnas con renglones en la tabla, nos referimos a este requerimiento como " El principio básico del modelo relacional".

2) Regla de acceso garantizado.

Marca que cada uno de los datos de una base de datos relacional tiene valor indisoluble, único y atómico, es decir; que no se puede descomponer, y que garantiza que sea lógicamente accesible especificando el nombre de la tabla, columna, renglón y valor de la llave primaria.

3) Tratamiento sistemático de valores nulos.

Los valores nulos (distintos de cadena de caracteres vacía o en blanco y distinta de cero o de cualquier otro número) es soportado por los SMBD completamente relacionales, para representar la falta de información inaplicable de un modo sistemático e independiente del tipo de datos.

4) Catálogo en línea dinámico basado en el modelo relacional.

La descripción de la base de datos se representa a nivel lógico, del mismo modo que los datos ordinarios, de tal manera que los usuarios autorizados, pueden aplicar el lenguaje de consulta relacional.

5) Regla de sublenguaje completo de datos.

Un sistema relacional puede soportar lenguajes que tengan una sintáxis lineal, puede ser usado por programas de aplicación e interactivos soportando operaciones con definición de datos (incluyendo definición de vistas), operaciones de manipulación (actualización, recuperación), forza la integridad y seguridad y manejo de operaciones de transacción (inicio, cumplimiento y regreso).

6) Regla de actualización de vistas.

Todas las vistas que sean teóricamente actualizables (es decir que admiten las operaciones de actualización que son modificación, inserción y borrado) pueden ser actualizadas por el sistema.

7) Insercción, modificación y borrado de alto nivel.

La capacidad de manejar un conjunto a la vez de operadores del sistema no sólo se aplica a la recuperación de datos, si no también a la inserción y borrado.

8) Independencia física de los datos.

Los programas de aplicación y las actividades realizadas por el sistema no altera el almacenamiento de los datos, cualquiera que sean los cambios efectuados, ya sea en la representación del almacenamiento o en los métodos de acceso.

9) Independencia lógica de los datos.

Los programas de aplicación y actividades del sistema permanecen lógicamente inalterados cuando se efectuan cambios sobre las tablas, cambios permanentes de la información de cualquier tipo.

10) Independiencia de integridad.

Las restricciones de integridad deben ser especificadas en forma independiente de las tablas y programas de aplicación. Debe ser posible cambiar tales restricciones sin ser afectadas las aplicaciones existentes.

11) Regla de independencia de distribución.

Las aplicaciones existentes deben continuar operando satisfactoriamente, aún cuando una versión de un SMBD es por primera vez introducida o cuando un dato existente es distribuído por el sistema.

12) Regla de no subversión.

Si el sistema muestra una interfaz de bajo nivel (un sólo registro cada vez), ese bajo nivel no puede ser utilizado para transtornar o suprimir las reglas de integridad y seguridad relacional expresada en un nivel superior (multiples registros a la vez).

Las tablas son arreglos rectangulares con renglones y columnas, donde los renglones se refieren a objetos (elementos) o procesos y las columnas describen a cada uno de los elementos del renglón, que tienen las siguientes propiedades:

- Cada elemento en la tabla representa un dato.

- En las columnas, todos los datos son de la misma clase.

- A cada columna se le asigna un nombre distinto.

- Todos los renglones son distintos y no se permite duplicarse.

- El orden del renglón no es significativo, el renglón puede ser intercambiado sin afectar la información o contenido de la tabla.

- El orden de las columnas no es prioritario, esto es cierto de acuerdo al punto (2).

2.3.1 MODELO DE DATOS. Tiene tres componentes que son los siguientes [4]:

1) *Estructura de datos*: Es una colección de objetos abstractos formado por datos.

2) *Operadores entre las estructuras*: Conjunto de operadores, con reglas bien definidas, que permiten manipular las estructuras de datos.

3) *Definición de integridad*: Colección de reglas y conceptos que pemiten expresar, qué valores de datos pueden aparecer válidamente en nuestro esquema.

La definición del modelo relacional de datos generalmente aceptada incluye la posibilidad de nulos, lo que lleva a definir los conceptos de clave, dominio, relaciones, atributos y entidad.

Estructura de datos: Dominio, relaciones, atributos y entidades.

+ Dominio: Es el conjunto de valores que puede tomar un atributo.

+ Relación: Es un conjunto o conexión de objetos que tienen el mismo tipo de características o atributos.

+ Atributo o característica: Cada atributo tiene un dominio asociado.

+ Entidad o n-áda: Es un elemento de datos con un conjunto finito de atributos. Consta de *n* valores, para cada uno.

Integridad: Los conceptos de claves o llaves primarias y foráneas consta de los valores nulos y también incluye dos reglas de integridad: Referencial y de entidad [5].

Llave primaria: Es un conjunto de uno o más atributos que sirven para distinguir cada entidad en la relación, y no puede tomar valores repetidos.

Llave foránea: Hace referencia a una llave primaria en otra relación, la cual puede tener una o más llaves foráneas.

2.3.2 SEGURIDAD DE LA INFORMACIÓN.

Integridad.

El modelo relacional tal y como lo definió E. F. Codd [5], proporciona las siguientes dos reglas de integridad:

Integridad de entidad.

Integridad referencial.

Es importante destacar que no todas las implementaciones de SMBD soportan estas reglas, además de estas, los diseñadores de bases de datos, pueden añadir sus propias restricciones respecto a la integridad, basadas en la aplicación.

La seguridad en los modelos relacionales se pueden establecer permitiendo a los usuarios acceder sólo a una "vista" (*view*) de la base de datos, en realidad la vista es un subconjunto, de tal forma que el segmento restante permanece protegido.

Los lenguajes de manipulación de datos, poseen proposiciones para restringir el acceso a los usuarios. Estas restricciones se pueden hacer a varios niveles,

si se restringe el acceso a los niveles más bajos el tiempo máquina aumenta. Algunos SMBD utilizan como mecanismo de seguridad el sistema operativo, para implementar su propio sistema de seguridad, en ocasiones no es posible. Por ejemplo, UNIX no suministra ningún mecanismo de seguridad a nivel de registro. Un SMBD ejecutando sobre un entorno UNIX, tiene que implementar estas especificaciones sin ningún soporte explícito del sistema operativo.

Para poder asignar estas llaves, el diseñador puede considerar lo siguiente:

Estabilidad: Considerar si algunas claves son menos propensas a sufrir modificaciones en sus valores.

Facilidad de uso: Por ejemplo, será más fácil de usar una clave numérica corta que otra alfanumérica con muchos caracteres.

Fiabilidad: Ver si alguna clave contiene dígitos de validación u otros mecanismos de autodetección o corrección de errores.

Universalidad: Puede haber llaves cuyo uso y conocimiento esté muy comprendido.

Valor nulo: Representa información incompleta o valores desconocidos y es representado por el signo "?".

Al fin de mantener la integrida de la base de datos relacional, se debe cumplir con algunas restricciones en cuanto a los valores de las llaves primarias:

a) Integridad de entidad (llaves primarias). Puesto que la llave primaria es un identificador designado para una relación, no debe tomar valores nulos para evitar ambigüedades.

b) Integridad referencial. Si se tiene una relación q con una llave primaria A de dominio D, y r otra relación con otro atributo B sobre D. Entonces cualquier valor del campo de A en r debe ser:

i) Nulo.

ii) El valor de una llave primaria de la otra relación q donde se tiene la llave primaria sobre D.

Operadores: Los primitivos del algebra relacional como la unión, diferencia, producto cartesiano, proyección y selección.

Los operadores del modelo relacional son de dos tipos, de actualización de entidades y de álgebra relacional.

Operadores de Actualización.

Para actualizar los valores de los atributos en las entidades, pueden efectuar las operaciones de agregar, borrar o modificar.

El manejo de las llaves foráneas hacen necesario establecer reglas, que determinan como manejar las operaciones de actualización de relaciones para no introducir inconsistencias, las cuales son:

Reglas para agregar.

- Al insertar una entidad en una relación, el valor de un atributo que es llave foránea puede ser nulo, o algún valor del dominio de la llave primaria.

Reglas para borrar.

Si se va ha borrar una entidad en una relación r con cierta llave primaria y otra relación q tiene a ese campo como llave foránea. Existen tres casos:

1) Borrado restringido: No se puede borrar una entidad en r que tenga entidades en q con el mismo valor como llave foránea.

2) Borrado en cascada: Al borrar una entidad en r se borrarán todas las entidades en q con ese valor.

3) Borrado con nulificación: Al borrar la entidad en r, a todas las entidades con igual valor en q se les pone el valor nulo.

Reglas para modificar.

* Modificar en cascada: Al modificar una llave primaria en r se le cambian los valores correspondientes en q.

* Modificación con nulificación: Al cambiar los valores de las llaves primarias en r a los correspondientes en q se les pone el valor nulo.

Operadores de conjunto.

Las bases de datos relacionales están basadas en los conceptos matemáticos de conjuntos. El álgebra relacional comprende las operaciones, que se pueden efectuar en los conjuntos: Unión, intersección, diferencia, producto cartesiano; como las específicas de las relaciones: Selección y proyección.

Propiedades de los operadores.

Se expresan en función de otros. Un estudio más a fondo de las propiedades de los operadores es prerequisito para poder optimizar la evaluación de expresiones; esto nos plantea dos problemas:

I) Transformar una expresión en otra equivalente.

II) Determinar si la equivalente puede evaluarse en un tiempo más corto.

Para resolver el primero hay que conocer las propiedades de los operadores.

1) $\cup$ es conmutativo y asociativo:

$(A \cup B) \cup C = B \cup (A \cup C)$.

2) Análogamente $\cap$.

3) Propiedad distributiva de $\cup$ y $\cap$:

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

4) Propiedad Conmutativa de la selección y el producto cartesiano:

$S(F1 \wedge F2)(A \times B) = (S(F1)(A)) \times (S(F2)(B))$.

donde la fórmula F1 sólo se refiere a los atributos de A y la F2 a los de B.

Si R y S son relaciones con todos los dominios iguales, se les puede aplicar todas las operaciones típicas de conjuntos.

R:

A	B	C
a	b	c
d	a	f
c	b	d

S:

D	E	F
b	g	a
d	a	f

Unión.

$R \cup S$. (Como unión de conjuntos.)

El resultado es una relación que incluye a todas las n–ádas de R y todas las de S, y ninguna más. Si hubiera alguna repetida de R y S, sólo figurará una vez en el resultado.

R y S deben ser del mismo grado.

$R \cup S$:

a	b	c
d	a	f
c	b	d
b	g	a

Diferencia.

R - S (Como diferencia de conjuntos.)

Nos da una relación que incluye a todas las n–ádas de R que no están en S.

R y S deben ser del mismo grado.

R - S:
a b c
c b d

Intersección.

$R \cap S$. (Como intersección de conjuntos.)

Proporciona una relación que incluye a todas las n–ádas de R que también están en S.

R y S deben ser relaciones del mismo grado.

$R \cap S$:
d a f

Producto cartesiano.

$R \times S$.

Obtendremos a todas las n–ádas posibles que se obtienen concatenando una de R con otra de S.

R y S pueden ser dos relaciones cualesquiera.

$R \times S$:

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

Permutación.

Esta operación se aplica a una sola relación, consiste en cambiar el orden de las columnas. Se denota por $\prod_{ijk}$ en donde se indican el orden en que estarán las columnas de la relación original. En este caso la primera columna es la i, la segunda la j y la tercera k.

Permutar $\prod_{ijk}(r)$:

B	C	A
b	c	a
a	f	d
b	d	c

Proyección.

$P(A_1, A_2, \ldots, A_n)(R)$: Proyección de la relación R sobre los atributos $A_1, A_2, \ldots, A_n$.

El resultado se forma de extraer de la relación R los atributos o columnas $A_1, A_2, \ldots, A_n$ y eliminando luego las n–ádas que resulten repetidas, si las hay.

A veces para abreviar, representaremos a una proyección, por ejemplo la $P(X, Y, Z)(R)$ como $R(X, Y, Z)$.

$P(X, Y, Z)(R)$:

a	c
d	f
c	d

Selección.

$S(F)(R)$: selección de la relación R de acuerdo con la fórmula F. El resultado se forma extrayendo de la relación R todas las eneadas que cumplan las condiciones expresadas en la fórmula F.

Esta fórmula se construye con operandos que pueden ser constantes, o atributos de R, ligados entre sí por los operadores definidos sobre los respectivos dominios.

$S(B=b)R$:

a	b	c
c	b	d

Cociente.

Sean R y S relaciones de grado (m+n) y (n), respectivamente. Esto lo representamos por R_{m+n}, S_n. Supongamos que S no es vacía.

Entonces se define el cociente entre R y S, que lo representamos por (R/S), como conjunto de todas las n–ádas de grado m tales que, al concatenarlas con todas las n–ádas de S, produce n–ádas contenidas en R.

R:

A	B	C	D
a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	e

S:

C	D
c	d
e	f

R/S:

A	B
a	b
e	d

CÁLCULO RELACIONAL.

El cálculo relacional es una de las bases del lenguaje de consulta relacional. Proporciona un conjunto de reglas para definir una relación. Si la expresión se aplica a una o más relaciones, el resultado sirve para definir una nueva, para consultar sobre un subconjunto de una ya existente, o actualizarla. Las expresiones utilizadas se construyen empleando los siguientes elementos.

Variables de n–ádas T, U, V,..., etc. cada variable de n–áda se restringe a variar sobre alguna relación con nombre. Si la variable de n–áda T representa a la n–áda *t* (en algún instante dado), entonces la expresión T.A representa al componente A de *t* (en ese instante), donde A es un atributo de la relación sobre la cual varia T.

Condiciones de la forma X*Y, donde * es cualquiera de los símbolos (=,≠,<,>,<=,>=) un operador relacional y al menos una entre Y y X es una expresión de la forma T.A y la otra es una expresión semejante o constante.

Fórmulas bien definidas (FBD), éstas se construyen a partir de condiciones, operadores lógicos tales como AND(Y), OR(O) o NOT(NO) y cuantificadores tales como "**exists**"(existe) o "**forall**"(para todos) de acuerdo con las reglas F1 - F2 que se presentan a continuación.

F1 : Toda condición es una FBD.

F2 : Si *f* es una FBD, entonces también lo son (*F*) y NOT(*F*).

F3 : Si *F* y *G* son FBD, también lo son (*F* AND *G*) y (*F* OR *G*).

F4 : Si *F* es una FBD es la cual T aparece como variable libre, entonces ∃ T *F* y ∀ T *F* son FBD.

F5 : Ninguna otra cosa es una FBD.

Con estos elementos se escriben las expresiones del cálculo relacional.

2.4 DISEÑO DE UN SISTEMA DE BASES DE DATOS.

Un buen diseño de base de datos, debe estar basado en la estructura global de los datos y también en el volúmen de los distintos atributos. En particular, debe considerarse el tamaño relativo de cada relación y tener en cuenta la frecuencia de acceso a ésta. Debe conocerse el atributo a través del cual se realizan las búsquedas en la relación. Esta información es útil para seleccionar el proceso y métodos de acceso.

Los SMBD proporcionan independencia entre los programas de aplicación, con respecto a los cambios estructurales en las bases de datos, tanto lógicos como físicos. Los programas una vez escritos, no necesitan volverse a escribir o ser recompilados, si se produce algún cambio en la definición de la base de datos.

Una de las ventajas de una base de datos relacional es su simplicidad. La única estructura de datos empleada es el registro. Diseñar una implica la agrupación de campos para formar un registro. Una relación universal es aquella que contenga todos los atributos de una base de datos. En el proceso de diseño, está se descompone en un conjunto de relaciones más pequeñas. Las relaciones finales, se deben someter a un conjunto de reglas que se denominan formas normales.

FORMAS NORMALES.

El proceso de normalización, se encarga de seguir una serie de pasos o normas que tras ser aplicadas, se obtienen los datos agrupados en diferentes tablas, de tal forma que obtenemos una estructura óptima para su implementación, manejo y explotación para diferentes aplicaciones futuras. Una tabla se dice que está en forma normal, cuando satisface un conjunto de restricciones impuestas por dichas normas.

El proceso de normalización parte de las formas normales definidas por E. F. Codd (1970), que formuló las tres primeras (1FN, 2FN y 3FN)[7]; más adelante se detectaron unas anomalías que llevaron a realizar una forma normal

más completa que la 3FN a la cual llamaron FNBC (forma normal de Boyce y Codd)[8], después Fagin definió la 4FN y 5FN[9].

Normalizar, es un proceso mediante el cual se examinan datos o grupos de datos y se aproximan a una forma que es capaz de ajustarse a cambios futuros, minimizando el impacto de cambios en las aplicaciones.

Es importante mencionar que la normalización describe la representación lógica de los datos, no la física.

En muchos de los casos, los datos existen como grupos de campos. Y estas agrupaciones al no estar normalizadas pueden ocasionar serios problemas en el futuro. Al normalizar, resulta una estructura de datos más clara y sencilla, la cual es necesaria para muchos de los subsecuentes pasos en los sistemas de información. La estructura resultante es también más estable y a la vez más abierta a los posibles cambios.

PRIMERA FORMA NORMAL (1FN).

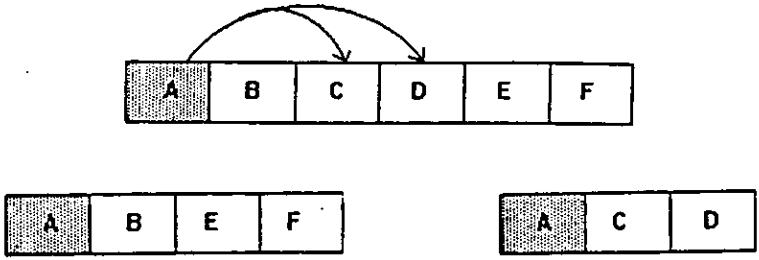
Se refiere a una clase de datos organizados, dentro de registros en los que no se encuentra repetición de grupos de campos dentro de un mismo registro. Es decir, son archivos planos, matrices bidimensionales de datos.

Ejemplo:

Sean A, B, C, D, E y F campos o distintos grupos de campos de un registro R representados de la siguiente manera:

Es decir que para ocurrencia de A existe repetición en los campos o grupos de campos C y D.

Al aplicar la primera forma normal, se convierte este registro en dos archivos planos.



SEGUNDA FORMA NORMAL (2FN).

Un registro R está en segunda forma normal, si está ya en primera forma normal y cada campo o atributo en el registro es funcionalmente dependiente de la llave candidata de R.

Para comprender la definición anterior, se definen los siguientes términos:

El campo B de un registro R es funcionalmente dependiente del "número de empleado", porque para cada número de empleado hay un determinado salario. En cambio, el "número de empleado" no es funcionalmente dependiente del campo "salario", porque más de un empleado puede tener el mismo salario.

-Dependencia Funcional. Un campo o colección de campos B de un registro R, puede tener dependencia funcional plena sobre otra colección de campos A del registro R, si B es funcionalmente dependiente de toda colección de campos A pero no de un subconjunto de A.

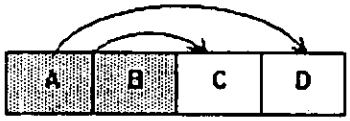
-Llave Candidata. Una llave candidata de un registro normalizado, es aquella que cumple con las siguientes propiedades:

i) Para cada ocurrencia de un registro, la llave identifica de una manera única a dicho registro.

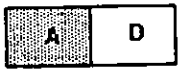
ii) No tiene redundancia, es decir que no se puede suprimir nada a la llave candidata sin que se destruya la propiedad de identificación única.

Ejemplo:

Se tiene el registro R con los campos o grupos de campos A, B, C y D relacionados como sigue:



La segunda forma normal exige que se separe el registro anterior de la siguiente forma:



En donde los campos no primarios C y D dependen funcionalmente de las llaves candidatas A y AB respectivamente.

Hay necesidad de separar el registro, porque D no tiene una dependencia funcional plena sobre la llave AB sino que sólo depende de A.

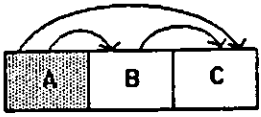
TERCERA FORMA NORMAL (3FN).

Un registro que está en segunda forma normal puede tener otro tipo de anomalías. Puede tener un campo que en si no es una llave pero identifica a otros campos. A esto se le llama *dependencia transitiva*, lo que pueden causar problemas. La tercera forma normal quita este problema.

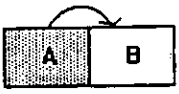
Entonces, formalmente se define que un registro R está en tercera forma normal si está ya en segunda forma normal y ningún campo no primario de R depende transitivamente de la llave primaria de R.

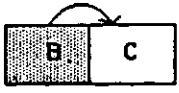
Ejemplo:

Sean A, B y C tres campos o distintas colecciones de campos de un registro R. Si C es funcionalmente dependiente de B y B es funcionalmente dependiente de A, entonces C es funcionalmente dependiente de A.



La conversión a la tercera forma normal quita la dependencia transitiva dividiendo el registro en dos:





FORMA NORMAL DE BOYCE-CODD(FNBC).

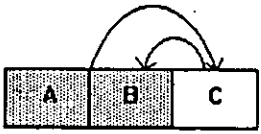
Tras la creación de la 3FN, se encontraron posteriormente anomalías.

Son el caso de las tablas que estando en 3FN mantienen una dependencia de un atributo secundario con parte de la clave. Para poder manejar esta dependencia en las aplicaciones, es imprescindible manejar una gran cantidad de registros innecesarios.

Un registro está en FNBC si y sólo si las únicas *dependencias funcionales* elementales, son aquellas en las que la clave principal (y la secundaria) determinan un atributo.

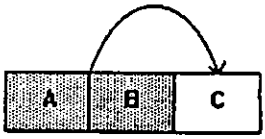
Ejemplo:

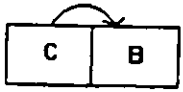
Se tiene el registro R con los campos o grupo de campos A, B y C relacionados de la siguiente manera:



Para pasar un registro a FNBC se realiza una proyección: se crea un registro con la parte de la clave que es independiente A y todos los atributos no primarios C.

Se crea otro registro con la parte de la clave restante y el atributo secundario del que depende, y será este último la clave del nuevo registro.





Lógicamente, en el primer registro el atributo C es clave ajena, a pesar de que pertenece a la clave principal y esto indica que mediante la unión se puede obtener el registro original.

- *Dependencia Multivaluada (DMV).* Sean A, B y C tres subconjuntos de atributos de una tabla T, se dice que A tiene una dependencia multivaluada con B, que A multidetermina a B, o que B depende multivaluadamente de A y se escribe:

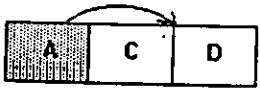
A—B o A *DMV* B.

CUARTA FORMA NORMAL (4FN).

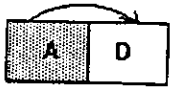
Esta forma normal se aplica para eliminar las *DMV* de los registros, puesto que implica gran redundancia y por tanto problemas de actualización y de integración de los datos.

Un registro se encuentra en 4FN si está en FNBC y las únicas *DMV* existentes entre dos atributos son: 1:*n* donde n 1, entonces un registro esta en 4FN si no mantiene *DMV*. Y para lograrlo se realizan proyecciones.

Ejemplo: En un registro se mantiene una *DMV*.



En el otro se deja la otra *DMV*, de tal modo que los atributos independientes, quedan cada uno en un registro.



- *Dependencia de Unión* (DU). Todas las dependencias anteriores son entre atributos. Ahora es una dependencia entre registros. Un registro T formado por los atributos $A_1 \times \cdots \times A_n$ tiene una *DU* con sus proyecciones $T_1 \times \cdots \times T_n$ si el registro original se puede obtener por medio de la unión de sus proyecciones $T_1^* \times \cdots \times T_n^*$.

QUINTA FORMA NORMAL (5FN).

Para que un registro se encuentre en 5FN debe estar en 4FN y todas sus *DU* vengan implicadas por las claves (primarias o secundarias) del registro. Para evitar los problemas de anomalías de actualización.

La 5FN se aplica en registros enormes que son intratables desde operaciones y que mantienen varias entidades todas relacionadas entre si de la forma 1:1. Estos registros están en 5FN pero no se aplica la *DU* para proyectarlas. Las proyecciones que se pueden formar, deben tener como atributos de unión una clave del registro, para cuando se realice la operación de unión, esto implica la relación de una n-áda con una única n-áda de otra proyección. Por lo tanto el registro se encuentra en 5FN y por razones de optimizar el diseño es importante crear las proyecciones.

El atributo que se utiliza para realizar la unión de las proyecciones, es la clave de la tabla original.

VENTAJAS DE NORMALIZAR.

Las ventajas que se obtienen tras la normalización de los datos para su eficaz manejo son:

- Flexibilidad.

La información que necesitan los usuarios se puede obtener de las tablas relacionales o relaciones mediante las operaciones del álgebra relacional. Por ejemplo: Uniendo tablas, seleccionando sus valores y proyectándolos.

- Precisión.

Las interrelaciones entre las tablas, consiguen mantener información diferente con toda exactitud.

- Seguridad.

Controla el acceso para consultar o actualizar información (a nivel tablas o de atributos).

- Facilidad de implementación.

Las tablas se almacenan físicamente como archivos planos.

- Independencia de datos.

Los programas no están ligados a las estructuras, con lo que se consigue aumentar la base de datos añadiendo nuevos atributos o nuevas tablas sin que afecten a los programas que los usan.

- Claridad.

La representación de la información es clara y sencilla para el usuario; son tablas simples.

- Facilidad de uso.

Los lenguajes manejan la información de forma sencilla, al estar los datos basados en el álgebra y cálculo relacional.

- Mínima redundancia.

La información no estará duplicada innecesariamente dentro de las estructuras.

- Máximo rendimiento de las aplicaciones.

La información, es seleccionada para ser útil a cada aplicación.

NORMALIZAR O DESNORMALIZAR.

Dado un esquema de diseño formado por varios esquemas de relación, $R_1, R_2, \ldots, R_n$, y un conjunto de dependencias, habrá en general algunos de ellos normalizados y otros no, y los normalizados lo estarán en distinto grado (FN1, FN2, FN3, FNCB, FN4, FN5). Es importante cuestionarse si es conveniente normalizar todos los esquemas y hasta qué grado. Esta pregunta debe responderla el analista o diseñador en cada caso particular, aunque deberá tener en cuenta algunas consideraciones de tipo general como las siguientes:

* Cuanto mayor sea la normalización, más fielmente se representa el mundo real en el esquema y, por tanto, el diseño será probablemente más estable.

* Si se normaliza al máximo, ni el sistema ni los programas deben responsabilizarse, para proteger la integridad de los datos, sólo de la unicidad de valores de las claves. Esto disminuye los riesgos de integridad y la redundancia, y por consiguiente mejora el rendimiento en operaciones de actualización.

* Por el contrario, una mayor normalización, suele implicar un número también mayor de relaciones (más pequeñas, o sea , con menos atributos cada una). Con lo que el rendimiento en consultas puede ser malo.

En conclusión, es conveniente normalizar, como directriz general de diseño, aunque puede haber excepciones por razones de rendimiento o facilidad de uso en consultas.

COMPROBACIÓN DEL DISEÑO DE UNA BASE DE DATOS.

Para una base de datos dada, no hay algoritmo que pueda determinar si su diseño es óptimo. Y al revés, no existe algoritmo alguno para diseñar una base de datos óptima. Sin embargo, hay varias pruebas que se deberían aplicar y detectar los posibles defectos.

Ventajas:

* Las relaciones entre todos los atributos deben estar representadas. Si no lo están, el diseño es incompleto o incorrecto. Si está incompleto, puede necesitar que se añada otra relación para representar la relación que falta.

* Una relación está presente en dos relaciones diferentes. Si es así, creará anomalías en las actualizaciones.

Estas pruebas verifican la corrección del proceso de normalización. Además debe tenerse en cuenta el factor rendimiento, si el tiempo de respuesta es demasiado lento para las recuperaciones, entonces, las relaciones están sobrenormalizadas y hay que desnormalizarlas meticulosamente, redefiniendo las relaciones de las bases de datos. El diseño es casi siempre un proceso interactivo, el cual debe probar su rendimiento y ajustárlo bien para óptimizarlo.

Los cambios en el diseño de la base de datos, para mejorar su rendimiento, deben ser transparentes a los programas de aplicación. La independencia de datos, juega un papel importante en el proceso.

VISTAS.

Son tablas ficticias cuya definición y n-ádas se obtienen a partir de una ó más tablas bases, mediante las cuales proporcionan determinada información al usuario.

Se llaman "vistas" a una expresión o fórmula algebraica a la que se le asigna un nombre. Este nombre puede utilizarse, como operando en la construcción de nuevas fórmulas a las que se les puede asignar un nombre a la vez.

El uso de vistas puede aplicarse para:

* Proteger la confidencialidad de acceso a los datos.

* Simplificar la formulación de expresiones complejas.

* Evitar que cambios en el diseño afecten a los programas ya existentes.

* Pueden estar definidas a partir de otras vistas.

Una vista que vaya a admitir las operaciones de actualización (modificación, inserción y borrado), debe estar limitada por las siguientes restricciones:

- Debe estar liberada de una sóla tabla base.

- Cada n-áda de la vista debe corresponder con un único atributo de la tabla base .

COMPONENTES DE UN SMBD.

El proceso más importante es el de consulta, interacciona con el usuario mediante órdenes específicas en un lenguaje denominado de consulta. Estos lenguajes son poderosos y bastante flexibles para proporcionar soluciones múltiples a un problema. Es decir, que podemos consultar la información necesaria a nuestras necesidades de una manera sencilla y de varias formas para obtener los mismos datos.

SEGURIDAD DEL SOFTWARE.

La disponibilidad de los mecanismos que controlen o protejan los programas o datos, es lo que llamamos seguridad.

Un sistema que maneja información y/o lleva a cabo acciones específicas puede ser objeto de actividades impropias o ilegales, lo que puede perjudicar o beneficiar a individuos o instituciones que controlan el sistema.

Una prueba de seguridad intenta verificar que tan sensibles son los mecanismos de protección incorporados en el sistema. Por lo que la seguridad del sistema debe ser probada en su vulnerabilidad ante una acción directa de penetrabilidad, así como acciones indirectas.

Durante la prueba de seguridad, todo se vale, intentar con las claves de acceso, por cualquier medio externo, a través del software, con algún diseño realizado para este propósito; intentando bloquear el sistema, negando así el servicio a otras personas; se pide que se cometan errores, al momento de recuperar información se trata de entrar al sistema, curiosando con los datos presentados.

Con el tiempo y los recursos necesarios, una prueba de seguridad terminará por penetrar en el sistema. El papel del diseñador del sistema es hacer que el costo de la penetración sea mayor que el valor de la información obtenida mediante el proceso de violación del sistema.

LENGUAJES DE CONSULTA.

Aunque la palabra consulta (*query*) connota la realización de una pregunta, en la terminología de los SMBD abarca también operaciones tales como la actualización y borrado de información contenida en la base de datos. Los lenguajes de consulta (*query languages*) soportados por la mayoría de los SMBD se basan en tres conceptos matemáticos: el álgebra relacional, el cálculo relacional de tuplas y el dominio del cálculo relacional. Los lenguajes comerciales suministran las características de estos lenguajes matemáticos y algunas adicionales.

LENGUAJES DE CONSULTA POPULARES.

El mercado de los SMBD en UNIX está dominado en gran parte por productos que soportan el lenguaje SQL (del inglés *Structured Query Language*. Lenguaje Estructurado de Consulta), como lenguaje de consulta relacional. Otro

de cierta popularidad es el QUEL, soportado por INGRES y que también soporta SQL. QUEL es más común en círculos académicos. Además QUEL está basado en el cálculo relacional, mientra que el SQL está basado en el cálculo relacional y el álgebra relacional. Un tercer lenguaje denominado *Query-By-Example* (QBE), desarrollado por IBM, tiene algunas características adicionales que no poseen ni el SQL ni el QUEL, se diseñó para usarse pantallas, y no es estrictamente un lenguaje.

2.5 INFORMIX-SQL.

INFORMIX-SQL es un sistema manejador de bases de datos (SMBD), el cual pertenece a la familia de productos relacionales, producidos por *Informix Software Inc.*(ISI), basado en un lenguaje de consulta estándar(*Structured Query Lenguage* (SQL), fácil de usar.

La organización de una base de datos relacional es considerada como una colección de tablas por lo que es distinta a una base de datos jerárquica o de red. INFORMIX-SQL proporciona herramientas para la manipulación de está en todos los aspectos:

- Crear y borrar base de datos.
- Crear, modificar y borrar tablas en una base de datos.
- Capturar datos desde una terminal.
- Proporciona informes.
- Presenta información compleja de la base de datos.
- Produce reportes desde la base de datos.
- Realiza aplicaciones individuales.
- Carga y descarga datos.
- Registra hechos y verifica la integridad de la base de datos.
- Proporciona las facilidades de un programa integrado.

INFORMIX-SQL es parte de una familia de productos como son: C-ISAM, INFORMIX-SQL, INFORMIX-4GL, INFORMIX-QUICKSTEP, ESQL/C, ESQL/COBOL,

ESQL/ADA, ESQL/FORTRAN, INFORMIX-SE, INFORMIX-TURBO, INFORMIX-ONLINE, INFORMIX-NET e INFORMIX-STAR.

Los miembros de la familia son continuamente perfeccionados y nuevos miembros son introducidos. La mayoría de estos productos han estado disponibles en varias versiones.

Para entender a los productos de la familia de INFORMIX es útil entender cómo el producto accesa a la base de datos. Todas las versiones de INFORMIX-SQL usan un doble proceso, a la que nos referimos también como cliente/servidor, para manejar la base de datos.

Esto significa que hay dos programas separados en cooperación para proporcionar un servicio al usuario. A uno de estos programas se denomina *front-end* el cual es responsable de interactuar con el usuario, y para traducir las solicitudes del usuario en declaraciones que muestran la información correcta desde la base de datos. El *front-end* no hace actualmente cambios a la base de datos; estos son hechos por otro programa llamado *back-end* o ingeniería de la base de datos, quien maneja las complejidades, sabe dónde el dato es almacenado y cuál es la mejor forma de accesarlo; algunas versiones del *back-end* están diseñadas para manejar redes y otras formas distintas de base de datos distribuídas cuando es necesario.

En *UNIX*, es un programa separado completamente, el cual es corrido automáticamente por el usuario y el programa del usuario se comunica con la base de datos a través de dos barras (||) o *pipes* (una forma de canales de comunicación) cada vez que el programa del usuario necesita ir por datos, envía un mensaje adecuado y espera una respuesta, una lectura o hacer cualquier otra cosa que le indique la retroalimentación.

Hay varias razones para organizar el acceso a la base de datos de esta forma:

1.- Garantiza que la base de datos siempre se maneje de la misma manera. En particular, significa que los permisos son tratados uniformemente y también las claves de las tablas. Esto significa que el mismo programa *front-end* puede trabajar con diferentes *back-ends*, y un *back-end* muestra un servicio para todas las clases de programas *front-ends*.

2.- Todos los programas usan los beneficios de la base de datos, como la optimización, cuando interpreta las solicitudes del usuario.

3.- La base de datos puede tener privilegios especiales. En particular la ingeniería estándar necesita ser hábil para crear archivos muy grandes, pero un usuario ordinario de UNIX puede crear sólo archivos moderadamente grandes. Los privilegios especiales, protegen la información de la base de datos de programadores inexpertos y poco cuidadosos.

4.- Aunque la ingeniería de la base de datos, puede ser implementada como una librería de subrutinas, esto significa que cada programa, puede ser tan grande que este listo, porque una copia del código puede ser almacenada con cada programa.

5.- La base de datos puede ser reemplazada por una nueva versión (o todo un programa diferente) sin tener que recompilar algunos programas. Si la ingeniería es reemplazada por un programa diferente, la nueva puede obedecer las mismas reglas -usa el mismo protocolo de comunicación- como la ingeniería de base de datos estándar.

Las herramientas como el *front-end* y los programas escritos usan la terminación *SQL*, todas las herramientas usan una de las ingenierías de las bases de datos para accesar a la base de datos.

INFORMIX-SQL proporciona a los usuarios que no son programadores, un conjunto completo de herramientas para manejar base de datos, como ya se describió.

DESARROLLO DE INFORMIX.

INFORMIX-SQL es producido por *Informix Software Inc* (ISI) en California, EEUU. Se desarrolló en 1980 como una Compañía de Sistema de Base de Datos por Roger Sippl y en 1986 fue cambiado el nombre a *Informix Software* para reflejar el nombre de los productos por los cuales fue dado a conocer. El primer producto producido fue C-ISAM el cual estuvo disponible para UNIX y DOS.

En 1982, surgió *Informix*, una sistema manejador de bases de datos, el cual usó C-ISAM para almacenar los datos. El nombre de **INFORMIX** proviene de las palabras "**information**"(información) y "UNIX.

Por 1984, surge INFORMIX-SQL el cual usa SQL como un lenguaje de consulta. Con *INFORMIX-SQL*, la base de datos como un todo, es almacenada en un directorio y la información sobre la estructura de la base de datos fue manejada

desde tablas creadas por el usuario. Una de las herramientas proporcionadas con INFORMIX-SQL es una forma de convertir bases de datos INFORMIX 3.3 en una INFORMIX-SQL con una estructura equivalente y los mismos datos como en la base de datos INFORMIX 3.3.

Cuando la versión 2.00 de INFORMIX-SQL fue realizada a mediados de 1986, los lenguajes de programación INFORMIX-4GL versión 1.00 fue diseñado también. Este tiene muchas características importantes para desarrollar aplicaciones rápidamente.

DISPONIBILIDAD DE INFORMIX-SQL.

INFORMIX-SQL está disponible en un gran número de máquinas de todos tamaños y corriendo en muchos sistemas operativos diferentes. El número de máquinas diferentes disponibles está contínuamente creciendo.

La máquina más pequeña en la cual corre INFORMIX-SQL son las PC's y que corran DOS primero PC-DOS ó MS-DOS. Para lo cual se requiere un disco duro de 640K de memoria total. La versión DOS no tiene todas las características de seguridad ya que no reconoce el concepto de usuario, es un sistema único para un usuario, lo cual significa, que sólo un usuario puede usar una máquina a la vez. Hay versiones disponibles para máquinas que trabajan DOS en red también. Esto implementa registros cerrados para que más de una persona, tenga acceso a los datos, al mismo tiempo en la red. INFORMIX-SQL está también disponible en máquinas que corren OS/2.

INFORMIX-SQL puede estar en máquinas que van de pequeñas a medianas y en máquinas multi-usuario. Un típico ejemplo puede ser una Compaq Deskpro 386/25 con 70 MB ó más disco, corriendo Xenix o una de las otras versiones de UNIX con el chip Intel 80386. INFORMIX-SQL está también disponible en máquinas del tamaño de una máquina que usa los chips de la Motorola.

Tambien INFORMIX-SQL está disponible en máquinas grandes como Amdahl IBM la cual corre UTS (una variante de UNIX y en la supercomputadora Cray-2 que corre UNICOS) otra variante de UNIX.

DIFERENTES VERSIONES.

Han sido cuatro las versiones importantes de INFORMIX-SQL: 1.10, 2.00, 2.10 y 4.00. La operación básica de INFORMIX-SQL no ha cambiado entre estas versiones, pero números detalles han cambiado. Las versiones 2.00 y 2.10 son muy similares y la versión 4.00 tiene algunas características extras.

2.6 ADMINISTRADOR DE LA BASE DE DATOS.

Controla el sistema que maneja los ciclos de información de los usuarios. Reestructura la base de datos con herramientas adicionales para análisis, procedimientos y comunicación. Los recursos requieren la asignación de capital de inversión que el administrador tendrá que obtener de la gerencia, para su buen funcionamiento necesitará básicamente de dos componentes: El sistema y el contenido.

Dados los recursos técnicos y tiempo, debera ser posible satisfacer todas las solicitudes, donde el administrador determinará las fuentes potenciales de datos, que a menudo se encuentran en el dominio de otros usuarios.

El esquema es la herramienta principal de que dispone el administrador para la asignación de recursos. La parte interna proporciona control sobre la eficiencia y responsabilidad de los procesos, y la matriz asociada de protección controla el acceso. La selección del formato externo determina la visión operativa que un usuario tiene del modelo de la base de datos. El esquema conceptual y la descripción del modelo están restringidos por la realidad y no es fácil modificarlos por necesidades específicas.

Además, el administrador debe contar con herramientas de vigilancia y medios para reestructurar la base de datos. El proceso automático es un concepto atractivo que hasta el momento sólo se ha aplicado en situaciones limitadas y experimentales.

CARACTERISTICAS DE UN ADMINISTRADOR:

Comprender las necesidades del usuario y la capacidad de manejar estas necesidades.

Poseer suficiente autoridad para manejar la mayoría de las necesidades del usuario, esto implica la capacidad para obtener o afinar los recursos de recolección de datos, programación, computación y comunicación.

Capacidad de programación para el mantenimiento del sistema y el desarrollo de herramientas.

3

EL SISTEMA DE BECARIOS DE LA FACULTAD DE CIENCIAS U.N.A.M.

3.1 PLANTEAMIENTO DEL PROBLEMA.

En el Departamento de Becas el proceso para registrar a becarios es muy laborioso, principalmente por que se tiene que manejar un expediente por becario y tenerlo almacenado en un archivo de papel. Esto complica la búsqueda de información de los becarios o asesores, para uso del Departamento o de otros servicios. El proceso para obtener estos datos resulta muy entretenido, ya que se tienen que revisar cada expediente.

Pero en la actualidad esto puede resolverse fácilmente, ya que se cuenta con *software* de aplicación desarrollado para estos casos, como son los SMBD (Sistemas Manejadores de Bases de Datos) que nos proporcionan muchas ventajas y siguiendo la metodología relacional se puede obtener un buen sistema para el control de becarios; donde el acceso a la información sea de una manera sencilla y rápida para todos los Departamentos.

El registro de becarios se lleva a cabo a través de las solicitudes que el Departamento de Becas da a los estudiantes o trabajadores de la UNAM y que después regresan con los datos solicitados y anexando algunos documentos necesarios para el trámite de la beca. Esta información es almacenada en un expediente y a su vez en un archivo de papel pero los datos más importantes se capturan en una base de datos sencilla.

Para el Departamento resulta muy entretenido buscar o consultar información específica de algún(os) becario(s) en los expedientes; y en la base de datos con que cuentan pueden consultar poca información.

Para agilizar la búsqueda de la información, se necesita sistematizar las funciones de trámites necesarios de los becarios, como son las altas, bajas, actualizaciones y consultas de los becarios y asesores sin dejar de archivar los documentos necesarios de cada becario.

El sistema debe contar con funciones importantes y necesarias para el manejo de los datos de los becarios. El uso de éstas funciones debe ser sencillo para el personal del Departamento de Becas.

Con el apoyo del personal del Departamento, quienes saben mejor las necesidades que debe cubrir el sistema, el cual se desarrollará hasta lograr su mayor

optimización, con esto se reducirá el tiempo necesario para realizar cada actividad del Departamento, consiguiendo así, ahorro de espacio de almacenamiento de la información, sustituyendo las búsquedas en el archivo de papel.

Definición del problema.

Se desarrollará un sistema que contará con las siguientes funciones:

- Alta de becarios.
- Modificación de datos.
- Baja de becarios.
- Alta de asesores.
- Baja de asesores.
- Actualización de datos.
- Reportes

Las cuales son importantes para el Departamento de Becas, porque ayudarán al personal a realizar su trabajo de una manera sencilla y práctica. Así mismo se tiene la opción a futuro de poder compartir esta información a los demás Departamentos a través de la red, las funciones permitidas son: Consultas y reportes de becarios y asesores.

Con el sistema se pretende poder almacenar información de los becarios, buscando poder consultar información más fácilmente y de esta manera dar información más precisa a las personas que lo soliciten en un tiempo más corto.

Los requerimientos son:

Datos personales del becario:

Nombre completo.
Registro federal de contribuyentes.
Carrera.
Teléfono particular.
Teléfono de la oficina.

Datos laborales:

Dependencia de adscripción.
Nombramiento.
Horas contratadas.
Antigüedad académica.
Institución donde labora.

Antecedentes académicos:

Licenciatura en:

Facultad o escuela.
Porcentaje de créditos.
Promedio.
Fecha de titulación.
Institución.

Especialidad en:

Facultad o escuela.
Porcentaje de créditos.
Promedio.
Fecha de titulación.
Institución.

Maestría en:

Facultad o escuela.
Porcentaje de créditos.
Promedio.
Fecha de titulación.
Institución.

Doctorado en:

Facultad o escuela.

Porcentaje de créditos.
Promedio.
Fecha de titulación.
Institución.

Otros estudios en:

Facultad o escuela.
Porcentaje de créditos.
Promedio.
Fecha de titulación.
Institución.

Características de la beca:

Institución que la otorga.
Nacionalidad de la beca.
Tipo de beca.
Tema o proyecto de investigación.
Inicio y término de la beca.
Renovación de la beca.
Prórroga de la beca (según sea el caso).
Si cumplió (con la beca en el tiempo determinado).
Fecha de titulación.
Se incorporó a la facultad.

Datos del asesor:

Nombre completo.
Area de especialización
Máximo grado académico.
Dependencia de adscripción.
Teléfono.

Estos requerimentos nos indican los campos que tendrán las pantallas de captura y a través de éstas se realizarán más fácilmente las funciones del sistema.

De acuerdo al equipo computacional con que se cuenta, el sistema se desarrollará bajo las siguientes condiciones:

Se instalará en una computadora 386, la cual tiene como sistema operativo *SCO UNIX System V/386* y cuenta con el manejador de base de datos *INFORMIX-SQL, software* sugerido por el Centro de Computo de la Facultad de Ciencias. La máquina funcionará como cliente/servidor, cuenta con la opción de ser instalada en red y así realizar sólo la función de servidor.

3.2 ANÁLISIS DEL SISTEMA.

El análisis del sistema se realizó partiendo de la situación actual del Departamento de Becas, estudiando las actividades que desempeñan para llevar a cabo el control de la información de los becarios.

Ya que las secretarias se encargan de realizar estas actividades, se les ha pedido que indiquen la manera de poder automatizar las funciones del sistema.

Ellas explicaron como se lleva a cabo el trámite de registro de becarios, el estudiante recibe una solicitud que llenan con los datos necesarios y además anexan documentos que el Departamento requiere. Con estos papeles se les crea un expediente y se captura cierta información en la base de datos, archivando después los expedientes.

Con base en las necesidades del Departamento el sistema debe contar con funciones que realicen altas, bajas, actualizaciones y consultas de becarios y asesores, además debe contar con la opción de reportes, los cuales nos proporcionan información específica de becarios y asesores, dándonos ciertas estadísticas como el número de becarios con ciertas características indicadas.

Definición de requerimientos.

Con la información proporcionada en el Departamento de Becas y el análisis realizado, se elabora una descripción más detallada de las características que debe cubrir cada función del nuevo sistema, para observar mejor el manejo de la información se representa está por medio del Flujo de Datos Yordon [10] siendo las siguientes:

En el control de becarios existen datos de entrada como son los datos del becario y del asesor obtenidos a través de la solicitud de la beca. A partir de estos tenemos unas salidas del sistema que son las consultas, los reportes y las estadísticas.

DIAGRAMA DE CONCENTRADOS

EXPANSIÓN DE PROCESOS A PRIMER NIVEL

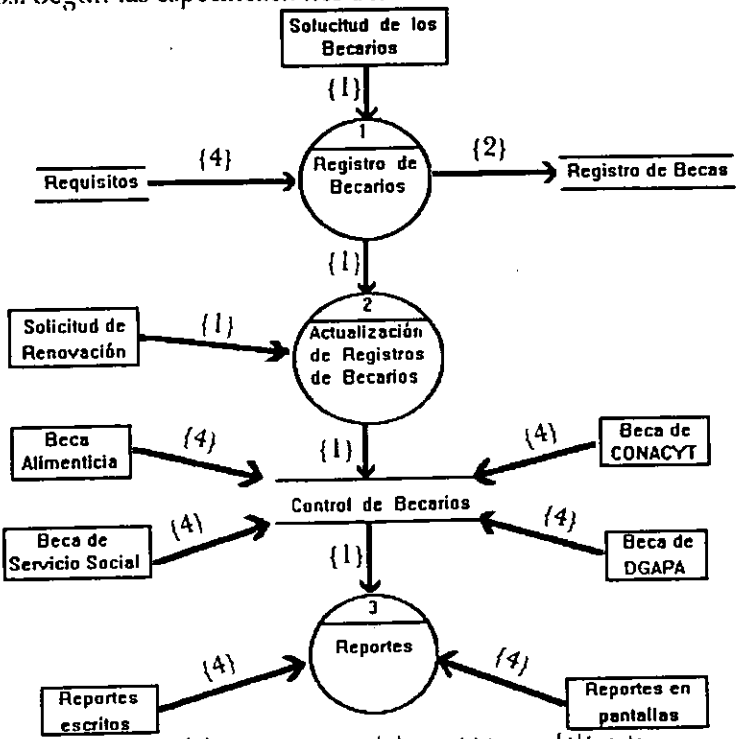
Las funciones principales del sistema se presentan en la siguiente figura:

(1) A través de la solicitud del becario y los requisitos que debe cubrir se hace el registro de la beca solicitada, pasar al siguiente proceso.

(2) Ya que se ha registrado el becario, es frecuente tener que aumentar, actualizar o quitar algunos datos del becario, en ocasiones por una solicitud de renovación ó por solicitud del Departamento o del becario, pasar al siguiente proceso.

En el Control de Becarios, se tiene concentrada toda la información de los estudiantes de las distintas becas que existen: Beca Alimenticia, Beca de Servicio Social, Beca CONACYT y Beca de DGAPA, pasar al siguiente proceso.

(3) Con la información almacenada se obtienen reportes por pantalla, archivo o impresos. Según las especificaciones del usuario.



* {1} Datos del Becario y Asesor. {2} Datos del Becario. {3} Datos del Asesor. {4} Listado.

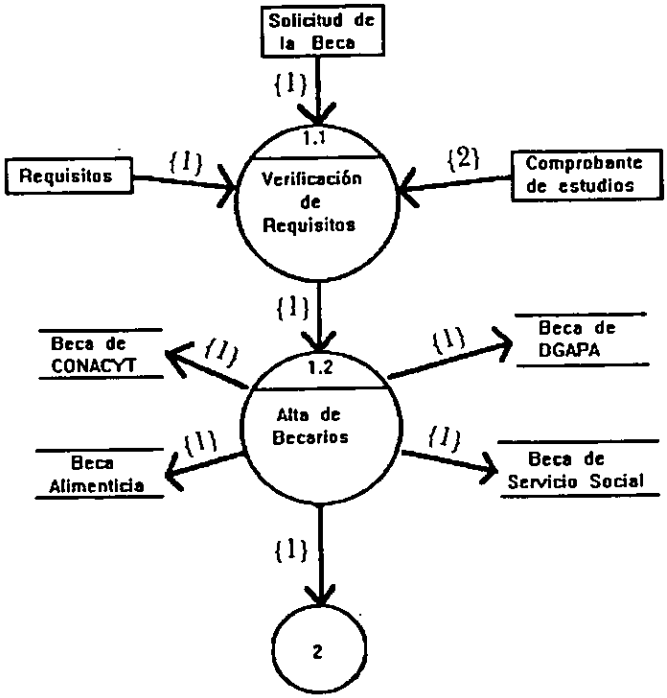
Cada una de las burbújas anteriores se pueden expander a un nivel más detallado como se ve acontinuación.

1. Registro de becarios y asesores.

(1.1) Verificar la solicitud de la beca, los requisitos y comprobantes de estudio que estén correctos y pasar al siguiente proceso.

(1.2) Se registran de los datos dando de alta al becario de acuerdo al tipo de beca que solicito: Beca de CONACYT, Beca Alimenticia, Beca de DGAPA y Beca de Servicio Social, pasar al siguiente proceso.

*



* {1} Datos del Becario y Asesor. {2} Datos del Becario. {3} Datos del Asesor. {4} Listado.

2. Actualización de registros de becarios y asesores.

(2.1) El control de becarios se hace por medio de las consultas de los becarios y los datos dependientes de ellos como son los del asesor y de la beca.

(2.2) La actualización es una parte importante en el control de becarios ya que siempre es necesario hacer correcciones o aumentar datos de los archivos de los becarios.

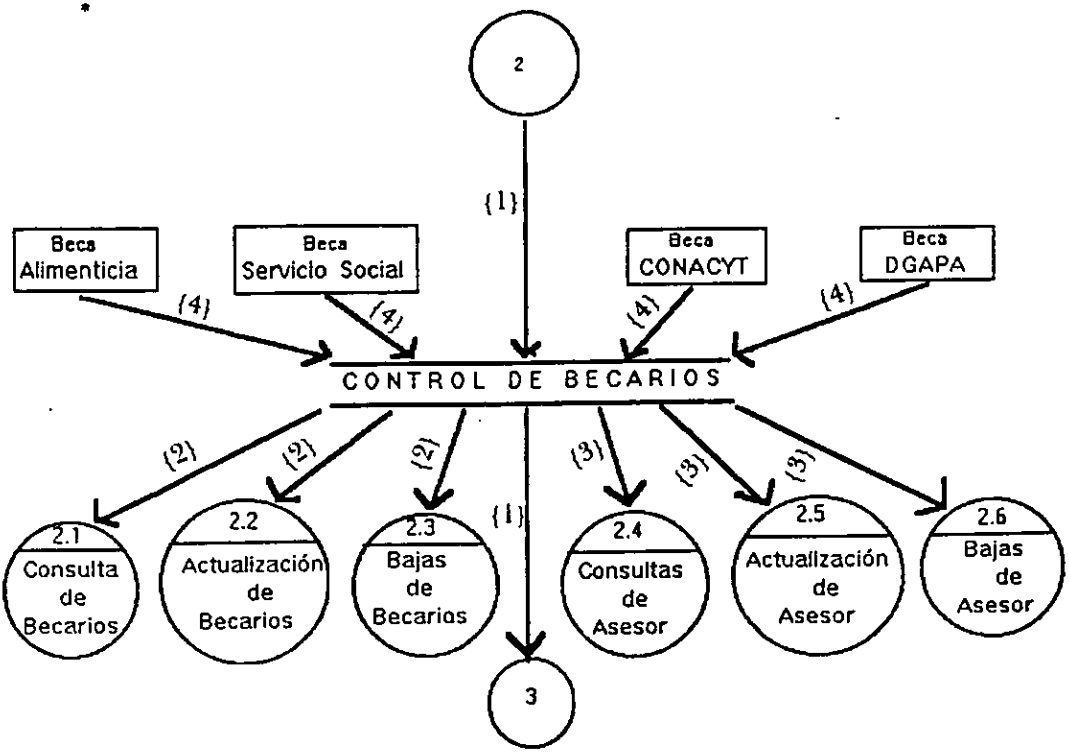
(2.3) Existe el proceso de dar de baja a becarios.

(2.4) Son importantes las consultas de asesores ya que hay datos que son importantes para el Departamento y que dependen de ellos.

(2.5) Las actualizaciones en cuanto a los datos del asesor se dan y por lo tanto hay que considerar el proceso.

(2.6) También las bajas de los asesores son importantes, ya que dependen de los becarios.

*



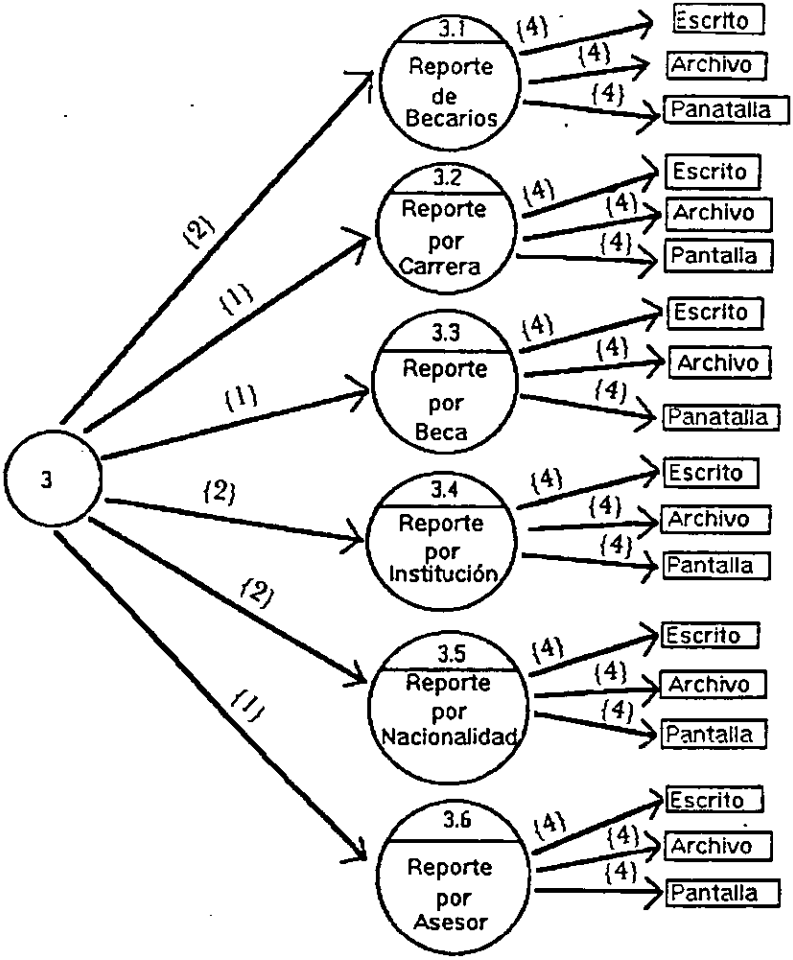
* {1} Datos del Becario y Asesor. {2} Datos del Becario. {3} Datos del Asesor. {4} Listado.

3. Reportes

Los reportes son una fuente importante de información ya que cuentan con datos específicos, por lo que se clasifican como sigue:

(3.1) Reporte de becarios.
(3.2) Reporte por carrera.
(3.3) Reporte por beca.
(3.4) Reporte por institución.
(3.5) Reporte por nacionalidad.
(3.6) Reporte por asesor.

La información obtenida se puede presentar en tres formas, por escrito, guardarse en un archivo o sólo presentarse en pantalla.

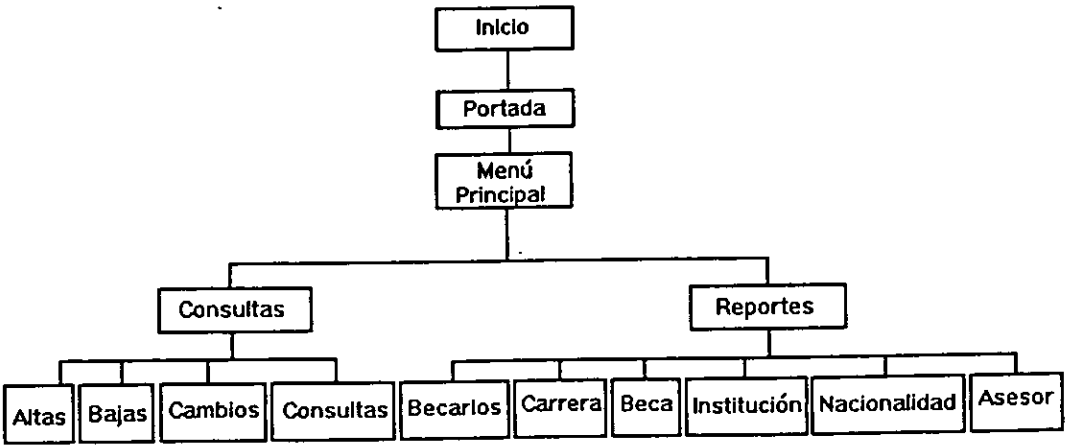


* {1} Datos del Becario y Asesor. {2} Datos del Becario. {3} Datos del Asesor. {4} Listado.

3.3 DISEÑO DEL SISTEMA.

El diseño será estructurado de manera jerárquica y quedará distribuído de la siguiente forma:

El inicio del sistema, se da desde el sistema operativo con un comando de línea, presentándose la portada y al mismo tiempo el menú principal, en donde existen 2 opciones: Las consultas y los reportes. A su vez en las consultas se pueden hacer altas, bajas, actualizaciones y consultas de la información de los becarios y asesores. Para el caso de los reportes cuenta con seis opciones para presentar la información: Por becarios (en general), por carrera, por tipo de beca, por institución, por nacionalidad y por asesor.



El diseño se desarrolla por pantallas, con los campos de los datos sugeridos por el Departamento de Becas. Las pantallas presentan información de un sólo becario a la vez, para el caso de los reportes se seleccionan del menú presentando.

Pantalla de presentación.

En la pantalla del sistema aparece el nombre del sistema, y el plantel en el cual se utiliza.

Menú principal.

En la misma pantalla aparecen las opciones de consultas y reportes, en la parte inferior de la pantalla nos indica como seleccionar la opción y como salirse del sistema.

Consultas.

Aparece el nombre de las pantallas de cada aplicación:

Datos Generales del Solicitante.

Antecedentes Académicos del Solicitante.

Características de la Beca.

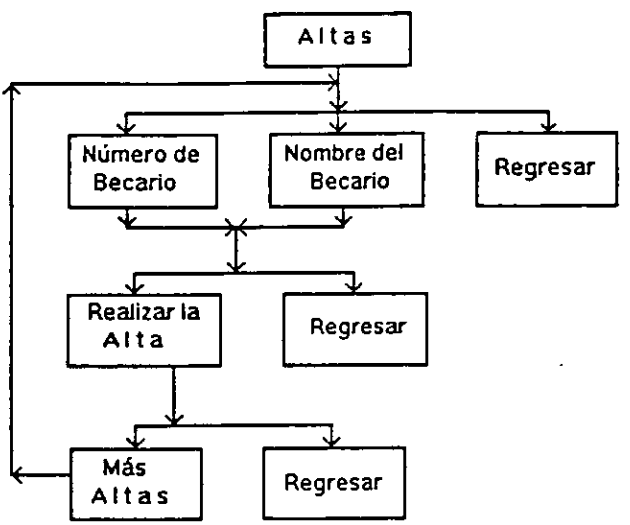
Datos del Asesor.

Estas pantallas nos ayudan a realizar más facilmente las altas, actualizaciones, bajas, búsquedas de datos; por medio del menú con que cuenta *INFORMIX-SQL*. Por medio de las pantallas nos enfocamos más fácilmente a los datos que nos interesan, esto hace que la información se maneje de una manera sencilla.

ALTAS.

Para hacer una alta de becarios es necesario verificar por medio del campo de número o nombre del becario si no existe en cada pantalla, para no duplicar valores. En la parte inferior de cada pantalla indicará si existen o no los datos del becario.

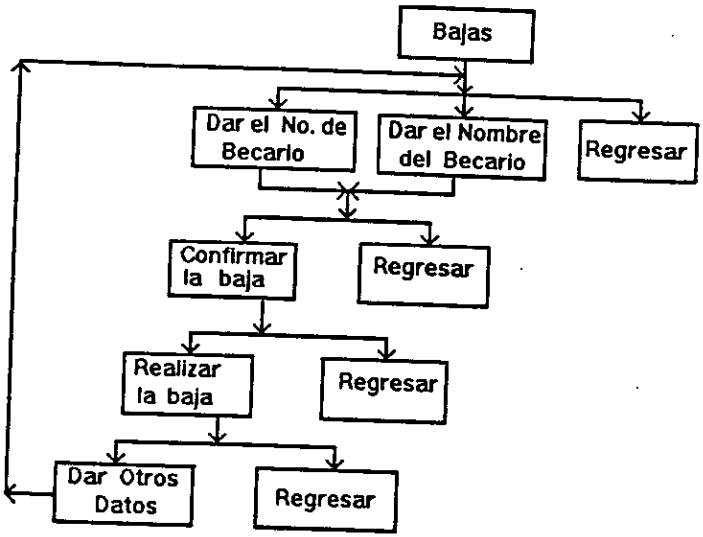
Solicitar la pantalla de datos de los registros que se desean dar de alta.
Presentar las pantallas que nos ayudan a llenar la base de datos.
Verificar que los registros a dar de alta no existan en la base de datos, de lo contrario no realizar la alta.
Realizar la alta.
Permitir tantas altas como se deseen.



BAJAS.

Para dar de baja un becario, se buscan los datos a través del número o nombre del becario, una vez presentados los datos a borrar se realiza la baja, confirmando con un mensaje en la parte inferior de la pantalla.

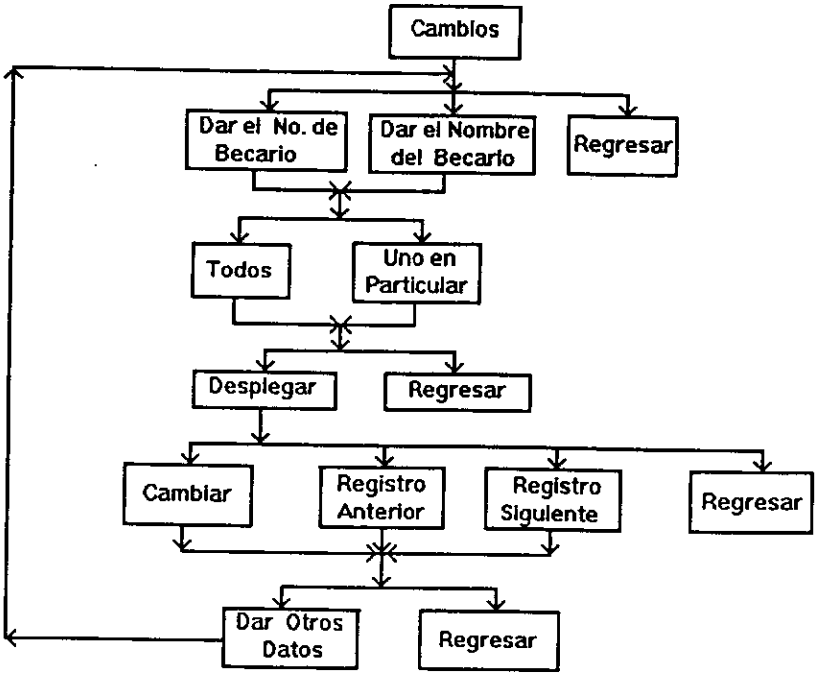
Solicitar la pantalla de datos
Buscar los registros que se desean dar de baja en la base de datos.
Verificar si son los registros deseados.
Realizar la baja.
Permitir realizar tantas bajas como se deseen.



CAMBIOS.

Para realizar una o más actualizaciones, se busca mediante el número o nombre del becario se hace la selección del renglón de datos a modificar y se lleva a cabo la actualización, confirmándose a través de un mensaje en la parte inferior de la pantalla.

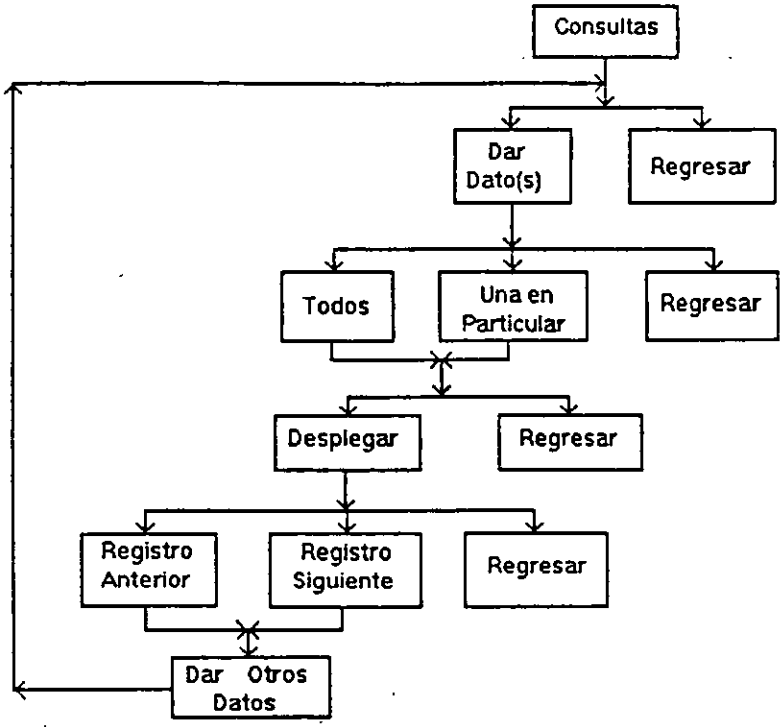
Solicitar la pantalla de datos.
Buscar los registros que se desean actualizar.
Presentar los datos de los registros buscados.
Verificar si son los datos que se desean actualizar.
Verificar que los cambios sean correctos.
Actualizar el cambio en la base de datos.



CONSULTAS.

Para consultar datos, solicitar la pantalla y ocupando la opción de búsqueda nos podemos desplazar a cualquier campo de la pantalla de aplicación y teclear los datos en el campo, el sistema presentará los datos por becario a la vez, en la parte inferior de la pantalla indicará cuántos registros con esas características encontró.

Solicitar las pantallas a consultar.
Presentar las pantallas a consultar.
Llenar los registros con las características a consultar.
Presentar todos los registros que cumplan con los parámetros señalados.



REPORTES.

El tipo de reportes con que cuenta el sistema son los siguientes:

Reporte de becarios.
Reporte por carrera.
Reporte por beca.
Reporte por institución.
Reporte por nacionalidad.
Reporte por asesor.

En la pantalla aparece el menú y en la parte inferior nos indica como seleccionarlos o como salir del menú. Ya que se ha seleccionado un reporte, presenta otra pantalla donde solicitará ciertos parámetros para dar el reporte. Cuando se han dado los parámetros da la opción a ser mandado a pantalla, a un archivo (dar el nombre que se le asignará al archivo) o a la impresora.

Las estadísticas están incluídas en los reportes, ya que al final del reporte aparece el número de becarios con las características señaladas.

==========

No. pag: 1 Fecha: 1995/09/13

==========

LISTA DE BECARIOS

==========

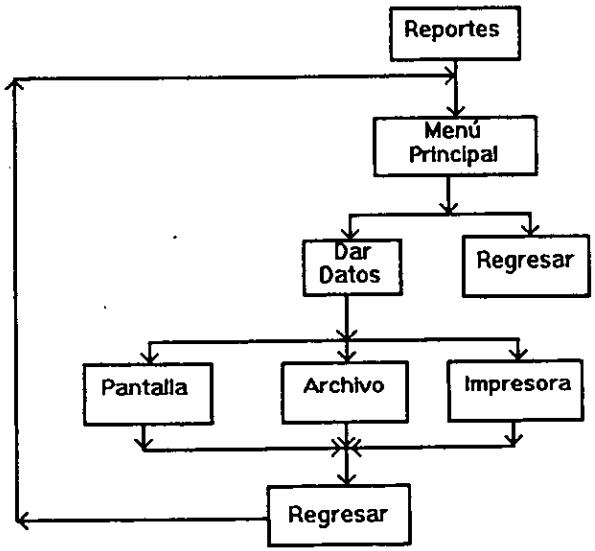
NUMERO	NOMBRE	CARRERA	TIPO DE BECA
1	AGUILA MARTINEZ CLAUDIA	ACTUARIA	TESIS DE LIC.
2	BARAJAS GUZMAN MARIA GUADALUPE	BIOLOGIA	MAESTRIA
3	CONTRERAS ARIZAGA FABIAN	BIOLOGIA	MAESTRIA

NUMERO TOTAL DE BECARIOS : 3

Se deberá respaldar la información de la base de datos en diskettes, dando así seguridad al sistema para el manejo de la información.

Los reportes se seleccionan de un menú con opción de regresar al menú principal, después de proporcionar el reporte regresa al menú anterior.

Al obtener un reporte el sistema indica el número de becarios que cumplen con ciertos requisitos solicitados; conociendo así algunas estadísticas de los becarios, para el mejor uso que se le asigne.



3.4 DISEÑO DE LA BASE DE DATOS.

Considerando las necesidades del Departamento de Becas se definieron las tablas que forman la base de datos.

El nombre de cada programa corresponde a la función que desempeña, el nombre de las tablas dependen de los datos que contienen.

Tal es el caso de las tablas de los datos académicos, la primera letra de cada campo indica a que tabla del grado académico pertenece, todos los demás campos son totalmente distintos para no haber confusión y el nombre del campo no excede de doce caracteres.

El nombre de la base de datos es becarios ya que se refiere al tipo de información que contiene.

TABLAS

Datos_becario

Nombre de la columna	tipo y longitud del campo
num_becario	serial
nombre_b	caracter(20)
apater_b	caracter(15)
amater_b	caracter(15)
r_f_c	caracter(10)
carrera	caracter(15)
tel_part	caracter(12)
tel_ofic	caracter(12)

Situacion_lab

Nombre de la columna	tipo y logitud del campo
num_becario	entero
dep_adsc	caracter(30)
nombra	caracter(30)
hrs_cont	entero
antg_acad	caracter(10)
institucion	caracter(20)

antc_acad_lic

Nombre de la columna	tipo y longitud del campo
num_becario	entero
lic_en	caracter(20)
lfac_esc	caracter(30)
lporc_cred	decimal(5,2)
lpromedio	decimal(5,2)
lfecha_tit	fecha
linstitucion	caracter(20)

antc_acad_espc

Nombre de la columna	tipo y longitud del campo
num_becario	entero
espc_en	caracter(50)
efac_esc	caracter(30)
eporc_cred	decimal(5,2)
epromedio	decimal(5,2)
efecha_tit	fecha
einstitucion	caracter(20)

antc_acad_maestria

Nombre de la columna	tipo y longitud del campo
num_becario	entero
maestria_en	caracter(20)
mfac_esc	caracter(30)
mporc_cred	decimal(5,2)
mpromedio	decimal(5,2)
mfecha_tit	fecha
minstitucion	caracter(20)

antc_acad_doc

Nombre de la columna	tipo y longitud del campo
num_becario	entero
doct_en	caracter(20)
dfac_esc	caracter(30)
dporc_cred	decimal(5,2)
dpromedio	decimal(5,2)
dfecha_tit	fecha
dinstitucion	caracter(20)

antc_acad_otros

Nombre de la columna	tipo y longitud del campo
num_becario	entero
otros_en	caracter(20)
ofac_esc	caracter(30)
oporc_cred	decimal(5,2)
opromedio	decimal(5,2)
ofecha_tit	fecha
oinstitucion	caracter(20)

caract_beca

Nombre de la columna	tipo y longitud del campo
num_becario	entero
inst_otorga	caracter(20)
nac_beca	caracter(15)
tipo_beca	caracter(20)
t_p_invs	caracter(110)
inicio	caracter(12)
termino	caracter(12)
renov1	caracter(2)
renov2	caracter(2)
prorroga	caracter(2)
cumplio	caracter(2)
f_titulo	fecha
inc_fac	caracter(2)

datos_asesor

Nombre de la columna	tipo y longitud del campo
num_becario	entero
nombre_a	caracter(20)
apater_a	caracter(15)
amater_a	caracter(15)
area_esp	caracter(45)
max_gr_acad	caracter(20)
dep_adsc	caracter(30)
telefono	caracter(12)

DICCIONARIO DE DATOS.

Un Diccionario de Datos, es una descripción central, de la estructura de la información almacenada en la base de datos. Todos los programas se apoyan en este diccionario para determinar dónde y cómo son almacenados los datos. Es importante determinar los valores que pueden tomar, de tal manera que no se vaya a escribir un registro con formato incorrecto y de ésta manera corromper la base de datos.

Este es más que una documentación del formato de los archivos de ésta forma ofrece más facilidades a un manejador de bases de datos para su operación y funcionamiento.

El diccionario de datos contiene el nombre del campo, el tipo de información que acepta cada campo, la longitud del campo y el nombre del archivo donde se encuentra cada campo.

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCION	NOMBRE DEL ARCHIVO
num_becario	caracter	4	Número del becario.	datos_becario, situacion_lab, antc_acad_lic, antc_acad_maestria, antc_acad_doct, antc_acad_espc, antc_acad_otros, datos_beca, caract_beca, datos_asesor.
nombre_b	caracter	20	Nombre del becario.	datos_becario.
apater_b	caracter	15	Apellido paterno del becario.	datos_becario.

amater_b	caracter	15	Apellido materno del becario.	datos_becario.
r_f_c	caracter	10	Registro Federal de Contribuyente.	datos_becario
carrera	caracter	15	Carrera del becario.	datos_becario, caract_beca.
tel_part	caracter	12	Teléfono particular.	datos_becario.
tel_ofic	caracter	12	Teléfono de la oficina.	datos_becario.
dep_adsc	caracter	30	Dependencia de adscripción.	situacion_lab.
nombra	caracter	30	Nombramiento.	situacion_lab.
hrs_cont	entero		Horas contratadas.	situacion_lab.
antg_acad	caracter	10	Antigüedad Académica.	situacion_lab.
institucion	caracter	20	Institución donde labora.	situacion_lab.
Lic_en	caracter	20	Nombre de la licenciatura.	antc_acad_lic.
lfac_esc	caracter	30	Facultad o escuela de la licenciatura.	antc_acad_lic.
lporc_cred	decimal	5,2	Porcentaje de créditos de la licenciatura.	antc_acad_lic.
lpromedio	decimal	5,2	Promedio de la licenciatura.	antc_acad_lic.

lfecha_tit	date	-	Fecha de titulación de la licenciatura.	antc_acad_lic.
linstitucion	caracter	20	Institución de la licenciatura.	antc_acad_lic.
espc_en	caracter	50	Nombre de la especialización.	antc_acad_espc.
efac_esc	caracter	30	Facultad o escuela de la especialización.	antc_acad_espc.
eporc_cred	decimal	5,2	Porcentaje de créditos de la especialización.	antc_acad_espc.
epromedio	decimal	5,2	Promedio de la especialización.	antc_acad_espc.
efecha_tit	date	-	Fecha de titulación de la especialización.	antc_acad_espc.
einstitucion	caracter	20	Institución de la especialización.	antc_acad_espc.
maestria_en	caracter	50	Nombre de la maestría.	antc_acad_maestria.
mfac_esc	caracter	30	Facultad o escuela de la maestría.	antc_acad_maestria.
mporc_cred	decimal	5,2	Porcentaje de créditos de la maestría.	antc_acad_maestria.
mpromedio	decimal	5,2	Promedio de la maestría.	antc_acad_maestria.
mfecha_tit	date	-	Fecha de titulación de la maestría.	antc_acad_maestria.

minstitucion	caracter	20	Institución de la maestría.	antc_acad_maestria.
doc_en	caracter	50	Nombre del doctorado.	antc_acad_doct.
dfac_esc	caracter	30	Facultad o escuela del doctorado.	antc_acad_doct.
dporc_cred	decimal	5,2	Porcentaje de créditos del doctorado.	antc_acad_doct.
dpromedio	decimal	5,2	Promedio en el doctorado.	antc_acad_doct.
dfecha_tit	date	-	Fecha de titulación del doctorado.	antc_acad_doct.
dinstitucion	caracter	20	Institución del doctorado.	antc_acad_doct.
oest_en	caracter	50	Nombre de otros estudios.	antc_acad_otros.
ofac_esc	caracter	30	Facultad o escuela de los estudios.	antc_acad_otros.
oporc_cred	decimal	5,2	Porcentaje de créditos de los estudios.	antc_acad_otros.
opromedio	decimal	5,2	Promedio de los estudios.	antc_acad_otros.
ofecha_tit	date	-	Fecha de titulación de los esudios.	antc_acad_otros.
oinstitucion	caracter	20	Institución de los estudios realizados.	antc_acad_otros.

inst_otorga	caracter	20	Instituto que otorga la beca.	caract_beca.
nac_beca	caracter	20	Nacionalidad de la beca.	caract_beca.
tipo_beca	caracter	20	Tipo de beca.	caract_beca.
t_p_invs	caracter	110	Tema o proyecto de investigación.	caract_beca.
inicio	caracter	12	Fecha de inicio de la beca.	carcat_beca.
termino	caracter	12	Fecha de término de la beca.	caract_beca.
renov1	caracter	2	Primera renovación sí o no.	caract_beca.
renov2	caracter	2	Segunda renovación sí o no.	carct_beca.
prorroga	caracter	2	Tiene prórroga la beca sí o no.	caract_beca.
cumplio	caracter	2	Cumplió con la beca sí o no.	caract_beca.
f_titulo	date	-	Fecha de titulación del becario.	caract_beca.
inc_fac	caracter	2	Se incorporó a la Facultad sí o no.	caract_beca.
nombre_a	caracter	20	Nombre del asesor.	datos_asesor.
apater_a	caracter	15	Apellido paterno del asesor.	datos_asesor.

amater_a	caracter	15	Apellido materno del asesor.	datos_asesor.
area_esp	caracter	45	Area de especialización del asesor.	datos_asesor.
max_gr_acad	caracter	20	Máximo grado académico del asesor.	datos_asesor.
dep_adsc	caracter	30	Dependencia de adscripción del asesor.	datos_asesor.
telefono	caracter	12	Teléfono del asesor.	datos_asesor

Una vez definida la base de datos, las tablas de los datos y los índices, se procedió a realizar las pantallas a través de las cuales se van a capturar los datos, éstas se realizaron de acuerdo a la información señalada por el Departamento de Becas, en cada pantalla se ocupan más de una tabla esto es debido a la necesidad de los datos que se deben presentar.

3.5 INSTALACIÓN.

El *software* de soporte está instalado en una computadora PC 386, la cual funcionará como cliente/servidor a la vez y se encuentra en el Centro de Cómputo. A la vez se tendrá un respaldo del sistema en diskettes, esté se encuentra vacío y listo para almacenar información.

Para la instalación del sistema se contó con todo lo necesario, como es la computadora, impresora y el espacio necesario para el equipo, así como un ambiente adecuado.

La instalación se realizó sin tener que molestar a los usuarios, y la participación de ellos es requerida para el entrenamiento y la asesoría necesaria para enseñarles el uso del sistema.

Es importante mencionar al usuario, lo sencillo que resulta su trabajo al ocupar el sistema, ya que automatiza muchas actividades antes realizadas, como son: Las consultas, actualizaciones, bajas y reportes, realizando éstas de manera fácil y rápida ya que toda la información importante de los becarios se encuentra concentrada en la base de datos.

Además, el sistema operativo en el cual se trabaja es UNIX, el cual presenta grandes ventajas de trabajo por ser multiusario y de multitarea, y el manejador de bases de datos que se utiliza es *INFORMIX-SQL*, ya que puede controlar bastante información, facilitando su manejo por ser un lenguaje de cuarta generación.

Es necesario estar al pendiente del usuario por cierto tiempo, para auxiliarlo en algunas dudas o problemas que pueda tener durante el uso del sistema y así poder hacer modificaciones a los programas o formatos, según sean necesarias o la modificación total del sistema.

Al no existir ya ningún problema con el uso del sistema, el Departamento de Becas se queda con la responsabilidad del sistema y su mantenimiento.

3.6 CONSTRUCCIÓN.

Con el diseño completo del sistema se elaboró la construcción física de los programas, codificándolos cada uno a través de INFORMIX-SQL, para conformar la estructura del sistema. También se utilizó programación shell para adaptar algunas instrucciones del sistema operativo para accesar al sistema de una manera muy sencilla. Y así la interfaz sea clara para el usuario.

La codificación de los programas será de la siguiente manera:

La base de datos.
Las tablas.
Las pantallas.
Los reportes.
Los menús.

Se empezó por crear una base de datos llamada "becarios", enseguida se estructuraron las tablas donde se guardan los datos de los becarios, de acuerdo al tipo de información, y basándose en éstas se prosiguió a formar las pantallas de captura de datos, se especificó la relación que llevan los campos con las tablas para controlar la información que se va almacenar, con la información que se tengan en la tablas se pueden obtener los reportes que también guardan una gran relación y finalmente se procede a crear los menús, empezando por dar las opciones que estarán en el menú principal y después las subopciones de cada opción principal, definiendo los programas con los cuales tienen que relacionarse para la ejecución de cada una.

Cada tabla es creada de acuerdo a los datos del becario:

- datos_becario

Contiene los datos personales de cada becario como son: Número de becario (lugar que ocupa en la base de datos para el control de la información), nombre completo, registro federal de contribuyente, carrera, teléfono partícular y/o de su oficina.

- situacion_lab

Contiene los datos laborales del becario si es el caso, como son: Número de becario, dependencia de adscripción, nombramiento, horas contratadas, antigüedad académica e institución donde labora.

- antc_acad_lic

Contiene los datos de los estudios a nivel licenciatura, como son: Número de becario, nombre de la licenciatura, facultad o escuela donde se cursó, porcentaje de créditos de la licenciatura, promedio, fecha de titulación si es el caso, e institución donde se realizó.

- antc_acad_espc

Contiene los datos de los estudios de la especialización, como son: Número de becario, nombre de la especialización, facultad o escuela donde se cursó, porcentaje de créditos de la especialización, promedio, fecha de titulación si es el caso, e institución donde se realizó.

- antc_acad_maestría

Contiene los datos de los estudios a nivel maestría, como son: Número de becario, nombre de la maestría, facultad o escuela donde se cursó, porcentaje de créditos de la maestría, promedio, fecha de titulación si es el caso, e institución donde se realizó.

- antc_acad_doct

Contiene los datos de los estudios a nivel doctorado, como son: Número de becario, nombre del doctorado, facultad o escuela donde se cursó, porcentaje de créditos del doctorado, promedio, fecha de titulación si es el caso, e institución donde se realizó.

- antc_acad_otros

Contiene los datos de otros estudios como son: Número de becario, nombre de los estudios, facultad o escuela donde se cursaron, porcentaje de créditos de los estudios, promedio, fecha de titulación si es el caso, e institución donde se realizó.

- caract_beca

Contiene los datos de la beca que solicitó el estudiante, como son: Número de becario, instituto que otorga la beca, nacionalidad de la beca, tipo de beca, tema o proyecto de investigación, inicio de la beca, término de la beca, renovación de la beca, prórroga de la beca, si cumplió con la beca, fecha de titulación y si se incorporó a la U.N.A.M..

-datos_asesor

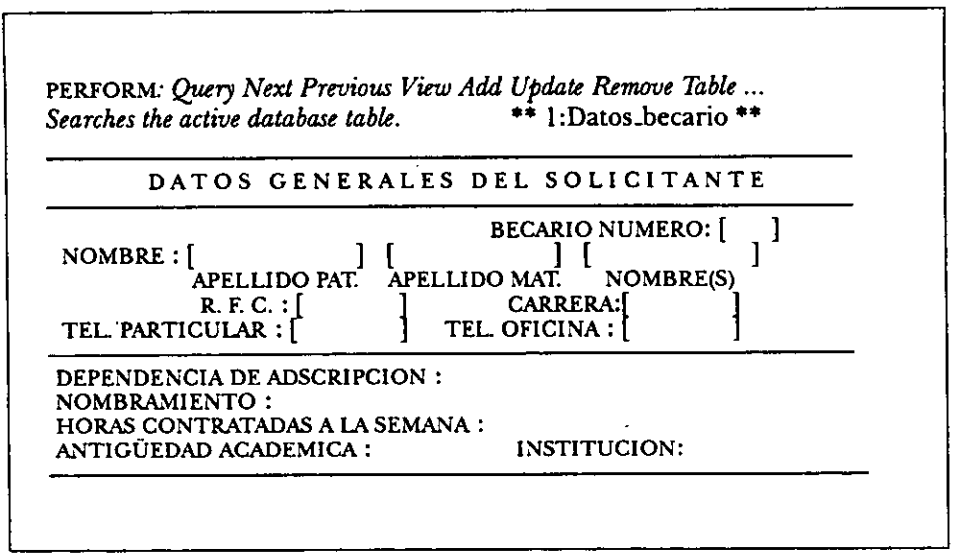
Contiene los datos generales del asesor, como son: Número de becario, nombre completo del asesor, máximo grado académico del asesor, dependencia de adscripción y teléfono.

Las tablas describen el tipo de campos de cada dato(columna), su longitud y el valor del campo (el cual nos indica la columna si es índice o no).

Cuando ya se han creado las tablas, se pueden hacer las pantallas, que sirven como interfaz, para que el usuario utilice los datos de la base de datos de una manera fácil y práctica.

Las siguientes pantallas conforman el sistema:

DATOS GENERALES DEL SOLICITANTE.



En esta pantalla se muestran los datos personales del solicitante y en caso de ser personal académico de la Facultad se muestran los datos de su situación laboral. Para la realización de esta pantalla se ocuparon las tablas de: datos_becario y situacion_lab. El número del becario es la llave primaria de las tablas.

- D_G_B (Datos Generales del Solicitante).

Esta pantalla hace uso de las siguientes tablas:

* datos_becario

* situacion_lab

ANTECEDENTES ACADEMICOS DEL SOLICITANTE.

PERFORM: *Query Next Previous View Add Update Remove Table ...*
Searches the active database table ** 1: Datos_becario **

ANTECEDENTES ACADEMICOS DEL SOLICITANTE

BECARIO NUMERO: [ ]
NOMBRE: [ ] [ ] [ ]
APELLIDO PAT. APELLIDO MAT. NOMBRE(S)
R. F. C. : [ ] CARRERA : [ ]
TEL. PARTICULAR : [ ] TEL. OFICINA: [ ]

LICENCIATURA EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION :

En esta pantalla se presentan los datos personales del becario y los datos académicos de la licenciatura. Aquí se ocuparon las tablas de: datos_becario y antc_acad_lic; ocupando como llave primaria el número del becario.

Con la opción *Screen* cambiamos a la siguiente pantalla.

PERFORM: *Query Next Previous View Add Update Remove Table ...*
Searches the active database table ** 1:Datos_becario **

ANTECEDENTES ACADEMICOS DEL SOLICITANTE

BECARIO NUMERO :
ESPECIALZACION EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION :

MAESTRIA EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION :

Esta pantalla presenta los datos académicos de la especialización y la maestría si es el caso. Se ocuparon las tablas de: datos_becario, antc_acad_espc y antc_acad_maestria; tomando como llave primaria el número del becario.

Con la opción *Screen* cambiamos a la siguiente pantalla.

PERFORM: *Query Next Previous View Add Update Remove Table ...*
Searches the active database table ** 1:Datos_becario **

ANTECEDENTES ACADEMICOS DEL SOLICITANTE

BECARIO NUMERO :

DOCTORADO EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION:

OTROS ESTUDIOS EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION :

En la pantalla se presentan los datos académicos del doctorado y otros estudios si es el caso. Ocupando las tablas de datos_becario, antc_ acad_doc y antc_acad_otros; considerando como llave primaria el número del becario.

- A_A_B (Antecendentes Académicos del Becario).

Esta pantalla hace uso de las siguientes tablas:

* datos_becario

* antc_acad_lic

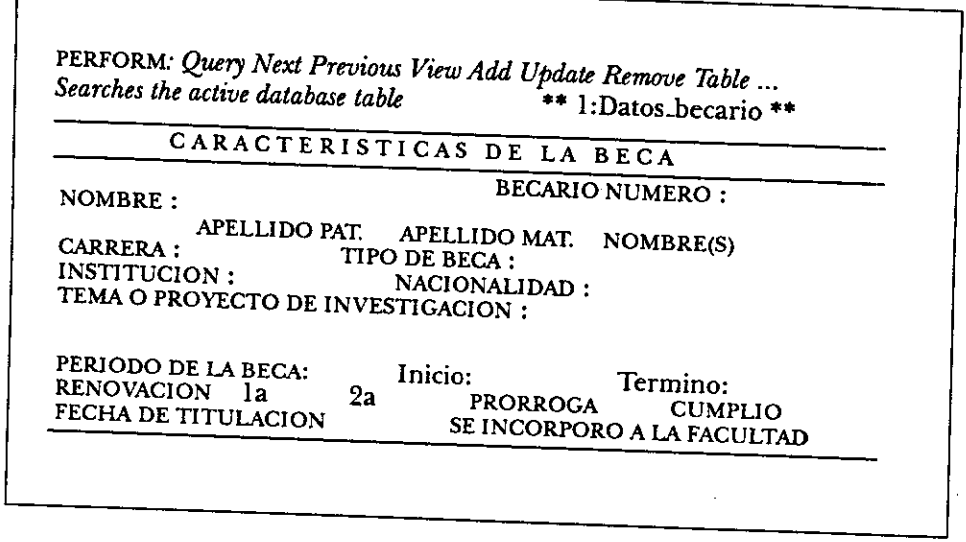
* antc_acad_espc

* antc_acad_maestria

* antc_acad_doct

* antc_acad_otros

CARACTERISTICAS DE LA BECA



Esta pantalla muestra el nombre del becario y datos sobre la beca. Para la elaboración de la pantalla se ocuparon las tablas de datos_becario y caract_beca; tomando como llave primaria el número del becario.

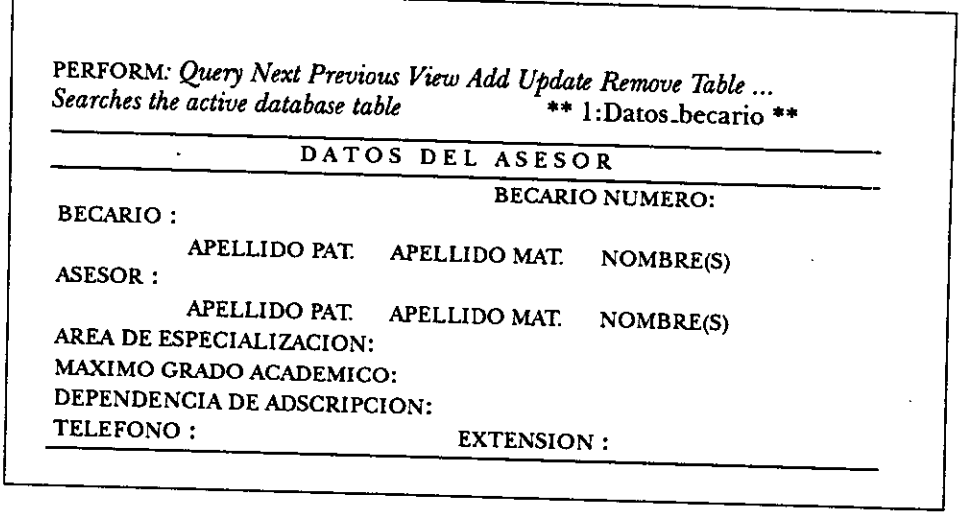
- C_G_B (Características Generales de la Beca).

Esta pantalla hace uso de las siguientes tablas:

* datos_becario

* caract_beca

DATOS DEL ASESOR.



La pantalla proporciona el nombre del becario y los datos del asesor. Se ocuparon las tablas: datos_becario y datos_asesor; tomando como llave primaria el número del becario.

En cada pantalla se pueden realizar las siguientes funciones: Altas, bajas, consultas y cambios. Y todas cuentan con un menú que permite realizar las opciones y salir de las pantallas al menú principal.

-D_A (Datos del Asesor).

Esta pantalla hace uso de las siguientes tablas:

* datos_becario

* datos_asesor.

Cada una de estas pantallas se codificó y compiló por separado y en las corridas se detectaron fallas de formatos y longitud principalmente, corrigiéndose satisfactoriamente.

Posteriormente se hicieron los reportes de acuerdo a los datos más solicitados, para que de esa manera se automatizará la búsqueda de información específica. Para ello cada reporte se codificó, compiló y documentó por separado.

REPORTES.

REPORTE DE BECARIOS.

Contiene cuatro columnas de datos: Número de becario, nombre completo del becario, carrera y tipo de beca de los becarios existentes. Al inicio pregunta a dónde se quieren enviar la información: A la pantalla, a un archivo o a la impresora, si se indica que a pantalla presenta la información, si la opción es a un archivo pregunta por el nombre que se le va a dar al archivo y si es a impresora lo manda inmediatamente. Al final indica el número de los becarios existentes.

REPORTE DE BECARIOS POR CARRERA.

Contiene cuatro columnas de datos: Número de becario, nombre completo del becario, tipo de beca y asesor. Al inicio solicita cuál carrera se quiere consultar, después indica a donde se quiere enviar la información (pantalla, archivo o impresora). Al final indica el número de los becarios existentes con las características señaladas.

REPORTE POR BECA.

Contiene cuatro columnas de datos: Número de becario, nombre completo del becario, carrera y asesor. Al inicio en la pantalla solicita el tipo de beca que se quiere consultar y después pregunta a dónde enviar la información (pantalla, archivo o impresora). Y al final hace el conteo de los becarios existentes con las características señaladas.

REPORTE POR INSTITUCION.

Contiene cuatro columnas de datos: Nombre completo del becario, tipo de beca, fecha de inicio y término de la beca. Al principio en la pantalla se solicita el nombre de la institución y después pregunta a dónde se quiere enviar la información (pantalla, archivo o impresora). Y al final indica el número de los becarios existentes con las características dadas.

REPORTE POR NACIONALIDAD.

Contiene cuatro columnas de datos: Nombre completo del becario, tipo de beca, fecha de inicio y término de la beca. Al comienzo de la pantalla solicita la nacionalidad de la beca y después pregunta a dónde enviar la información (pantalla, archivo o impresora). Y al final reporta el número de los becarios que existen en la base de datos con la característica señalada.

REPORTE POR ASESOR.

Al principio de la pantalla solicita algunos datos del asesor: Los apellidos y el nombre del asesor, después pregunta a dónde enviar la información (pantalla, archivo o impresora) y presenta los siguientes datos: Nombre del asesor y grado académico, además aparecen cuatro columnas de datos: Nombre completo del becario, tipo de beca, carrera e institución que la otorga. Al final indica el número de los becarios que dependen del asesor.

En los reportes se presenta un conteo del número de becarios que cumplen cierta característica, los datos presentados indican cierta estadística en cada reporte.

Una vez que se tienen todos los programas codificados, compilados y documentados se procede a formar el menú principal y los submenús del sistema. Esto es algo muy sencillo debido a las características de *INFORMIX-SQL*, ya que primero se estructura el menú principal.

Opciones del menú principal:

* Consultas

* Reportes

El submenú de consultas contiene las siguientes opciones:

* Datos Generales del Solicitante

* Antecedentes Académicos del Solicitante

* Datos del Asesor

* Características de la Beca

Cada opción llama a la pantalla en turno.

El submenú de reportes está integrado por:

* Reportes de Becarios

* Reporte de Becarios por Carrera

* Reporte por Beca

* Reporte por Institución

*Reporte por Nacionalidad

* Reporte por Asesor.

Donde cada opción llama al reporte seleccionado.

3.7 SEGURIDAD DEL SISTEMA DE BECARIOS.

Una forma de mantener la seguridad del sistema es a través del acceso con una clave única, está se da por jerarquías de importancia: El administrador es el único que tiene todos los permisos del sistema, tanto del sistema operativo como de la base de datos. Existen usuarios con el permiso de escritura y lectura para poder actualizar, modificar y borrar la información; lo cual le asigna una responsabilidad mayor para el manejo del sistema. Los demás usuarios que sólo consultan datos tienen tambien una clave de acceso pero sólo con permiso de lectura. Esta es una manera de proteger los programas y la información que contiene la base de datos. La estructura de base de datos dicta la organización, los métodos de acceso, el grado de asociatividad y las alternativas de procesamiento de información.

Otra forma de asegurar la integridad de la información es con la validación de datos que corresponden a cada campo de las pantallas, así se evita capturar caracteres erróneos que no correspondan.

3.8 MANTENIMIENTO.

El mantenimiento consiste en revisar en un principio que el sistema trabaje adecuadamente, como ya se explicó anteriormente después de las pruebas de evaluación del funcionamiento del sistema, se procede a realizar un curso para los usuarios del sistema en el cual, se les da una explicación detallada de la manera de utilizar el sistema.

El material a utilizar para la capacitación en esté caso es el manual del usuario que se encuentra en el apéndice. Allí se describe la forma de como se presenta el sistema, y nos indica como trabajar con las pantallas y como realizar los reportes.

Durante la capacitación puede ser posible modificar si es necesario algunas fallas o incongruencias que resulten del trabajo práctico de los usuarios, esto nos lleva a obtener un sistema más eficaz. Es importante mencionar que el período de mantenimiento es durante toda la vida del sistema, en el cual se le pueden realizar modificaciones según las necesidades del servicio de becas.

Para calquier consulta sobre el sistema dirigirse al Centro de Cómputo de la Facultad de Ciencias.

CONCLUSIONES

CONCLUSIONES.

El presente trabajo ha sido concebido con la intensión de elaborar un sistema de control de becarios de la Facultad de Ciencias de la U.N.A.M.

El presente sistema está destinado para personas con pocos conocimientos sobre computación, pero que a través de una previa capacitación, logren máximizar la aplicación del sistema y así descubran las ventajas para realizar su trabajo.

En base a la ingeniería del software es importante una comunicación intensa entre el cliente y el analista. El cliente debe comprender los objetivos del sistema y ser capaz de exponerlos claramente. El analista debe saber qué preguntas hacer, qué consejos dar y qué investigación realizar. Si la comunicación se rompe en esta fase, el éxito del proyecto entero estará en peligro.

Se diseñó este sistema de tal manera que considera aspectos de la ingeniería del software, considerada como una disciplina que integra métodos, herramientas y procedimientos para el desarrollo del software. Se han propuesto varios paradigmas diferentes , cada uno exhibiendo ventajas y desventajas, pero todos tienen una serie de faces genéricas en común haciendolo más accesible y sencillo posible desde el punto de vista computacional, para que cualquier persona autorizada pueda hacer uso del sistema de manera eficiente.

El sistema se desarrollo en un sistema operativo UNIX, el cual nos proporciona un entorno adecuado para ser utilizado en red y después de un análisis de modelos de las bases de datos, se aplicó la metodología de las Bases de Datos Relacionales que nos acerca a un sistema óptimo. Además se ocupó una herramienta de cuarta generación llamada INFORMIX-SQL, para la estructura del sistema.

La administración del sistema realiza el mantenimiento que se da durante toda la vida del mismo, el cual consiste en comprender, corregir, adaptar y/o mejorar el software, para adaptar el sistema a un mejor servicio. Así como proporcionar la atención debida a los usuarios asignados para su manejo.

La computadora por el momento está como cliente/servidor, el cual puede ser convertido a servidor, con la instalación de una dirección en el sistema operativo UNIX e instalándose terminales en otros lugares donde necesiten la información de los becarios. Ya que otros Departamentos también necesitan datos de

los becarios y lo relacionado con ellos por lo que es muy importante compartirlos. El uso de la computadora como sistema de información es un objetivo del costo-beneficio.

El sistema desarrollado es también una opción para otras instituciones que manejen datos de becarios, con la facilidad de adaptarse a las necesidades particulares.

MANUAL DEL USUARIO

MANUAL DEL USUARIO.

1. ENTRADA AL SISTEMA.

El sistema "BECARIOS DE LA FACULTAD DE CIENCIAS, U.N.A.M" está en un sistema operativo UNIX por lo cual hay que tener una clave de acceso (*Login*) y una de identificación personal(*Password*). La manera de entrar al sistema es a través de un comando de línea. En la pantalla aparecerá:

```
scosysv

Welcome to SCO System V/386

scosysvilogin:

Password:
```

En *scosysvilogin* se teclea la clave del usuario y en el *Password* va la clave secreta del usuario que da acceso al sistema UNIX apareciendo el prompt #.

```
        Welcome to SCO System V/386
                From
        The Santa Cruz Operation, Inc.


#
```

Para que el usuario pueda tener acceso al sistema "BECARIOS DE LA FACULTAD DE CIENCIAS, U.N.A.M" debe contar con una cuenta en el sistema operativo, un *login* y un *password*. El *login* si aparece en la pantalla pero el *password* no por ser secreto.

Cuando ya hayas entrado al sistema operativo y aparezca el prompt se teclea:

```
# $BECAS
```

or

```
# $becas
```

Esto nos da acceso al sistema "BECARIOS DE LA FACULTAD DE CIENCIAS, U.N.A.M.", presentando el menú principal:

```
                BECARIOS DE LA FACULTAD DE CIENCIAS, U.N.A.M.

1. CONSULTAS

2. REPORTES




User space bar, arrow keys, or type number to make selection.
Enter "e" to return to previous menu or exit.
Enter carriage return to execute selection:
```

Usando la barra espaciadora, las teclas de las flechitas o tecleando el número de la opción se hace la selección del menú.

2. CONSULTAS.

Al seleccionar la opción 1 aparece otro submenú.

CONSULTAS

1. DATOS GENERALES DEL SOLICITANTE
2. ANTECEDENTES ACADEMICOS DEL SOLICITANTE
3. DATOS DEL ASESOR
4. CARACTERISTICAS DE LA BECA

Use spacebar, arrow keys, or type number to make selection.
Enter "e" to return to previous menu or exit.
Enter carriage return to execute selection:

PERFORM: *Query Next Previous View Add Update Remove Table ...*
Searches the active database table. ** 1:Datos_becario **

PERFORM: ... *Screen Current Master Detail Output Exit*
Shows the next page of the form ** 1:Datos_becario *table* **

Se cuenta con el siguiente menú:

* *Query*(Consulta): Consulta la base de datos a partir de valores introducidos en uno o varios campos.

* *Next*(Siguiente): Si se han encontrado más de un renglón en la base de datos, se puede ir del renglón extraído al siguiente.

* *Previous*(Anterior): Nos muestra el renglón anterior al que estamos viendo.

* *View*(vista): Nos muestra parte o toda la base de datos sin temor a ser alterada.

* *Add*(Altas): Nos permite aumentar datos a la base de datos.

* *Update*(Actualizar): Una vez almacenados los datos es posible actualizar o cambiar algún campo.

* *Remove*(Borrar): Nos permite borrar algún renglón de la base de datos, hay que manejarlo con cuidado.

* *Table*(Tablas): Nos permite activar tablas en una pantallas con más de una tabla y cambiar de pantallas.

* *Screen*(Pantalla): Al tenerse definidas múltiples pantallas en un único archivo de formato, esta opción nos permite movernos a través de las pantallas.

* *Current*(Corriente): Nos permite ver los datos del renglón actual en la base de datos.

* *Master*(Maestra): Activa la tabla maestra de una relación de tablas.

* *Datail*(Detallada): Activa una tabla que dependa de la tabla maestra.

* *Output*(Salida): Nos permite almacenar la imagen de la pantalla del renglón o renglones actuales en un archivo UNIX.

* *Exit*(Salida): Sale de *PERFORM* y regresa al menú de Consultas.

En la opción (1) aparece la siguiente pantalla:

PERFORM: *Query Next Previous View Add Update Remove Table ...*
Searches the active database table. ** 1:Datos_becario **

DATOS GENERALES DEL SOLICITANTE

BECARIO NUMERO: [ ]
NOMBRE : [ ] [ ] [ ]
APELLIDO PAT. APELLIDO MAT. NOMBRE(S)
R. F. C. : [ ] CARRERA:[ ]
TEL. PARTICULAR : [ ] TEL. OFICINA : [ ]

DEPENDENCIA DE ADSCRIPCION :
NOMBRAMIENTO :
HORAS CONTRATADAS A LA SEMANA :
ANTIGüEDAD ACADEMICA : INSTITUCION:

En la opción (2) aparece la siguiente pantalla:

PERFORM: *Query Next Previous View Add Update Remove Table ...*
Searches the active database table ** 1: Datos_becario **

ANTECEDENTES ACADEMICOS DEL SOLICITANTE

BECARIO NUMERO: [ ]
NOMBRE: [ ] [ ] [ ]
APELLIDO PAT. APELLIDO MAT. NOMBRE(S)
R. F. C. : [ ] CARRERA : [ ]
TEL. PARTICULAR : [ ] TEL. OFICINA: [ ]

LICENCIATURA EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION :

Con la opción *Screen* cambiamos a la siguiente pantalla:

PERFORM: *Query Next Previous View Add Update Remove Table ...*
Searches the active database table ** 1:Datos_becario **

ANTECEDENTES ACADEMICOS DEL SOLICITANTE

BECARIO NUMERO :

ESPECIALZACION EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION :

MAESTRIA EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION :

Con la opción *Screen* cambiamos a la siguiente pantalla:

PERFORM: *Query Next Previous View Add Update Remove Table ...*
Searches the active database table ** 1:Datos_becario **

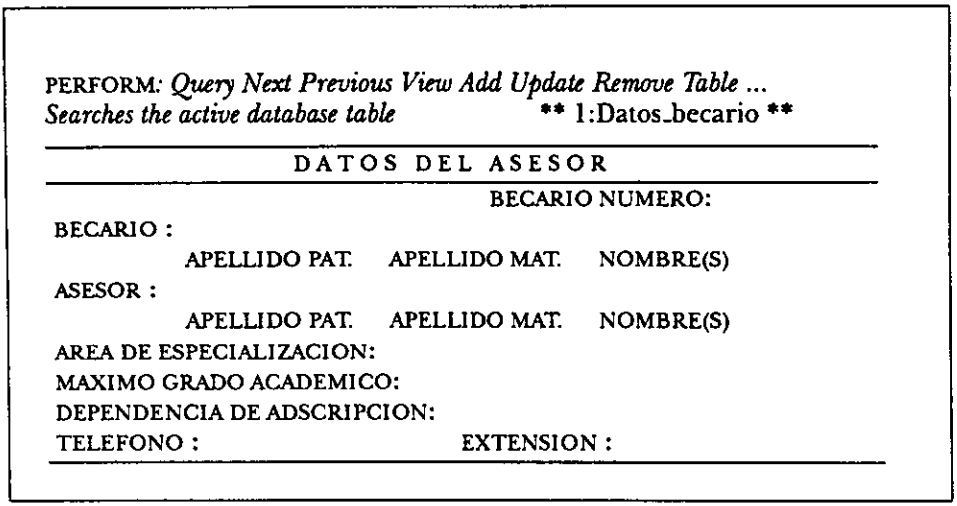
ANTECEDENTES ACADEMICOS DEL SOLICITANTE

BECARIO NUMERO :

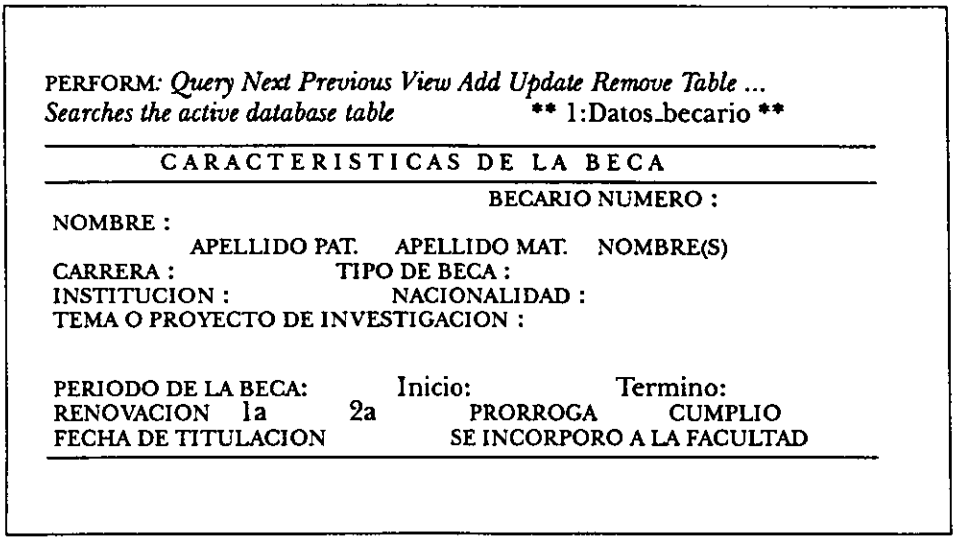
DOCTORADO EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION:

OTROS ESTUDIOS EN :
ESCUELA O FACULTAD :
PORCENTAJE DE CREDITOS : PROMEDIO :
FECHA DE TITULACION : INSTITUCION :

En la opción (3) aparece la siguiente pantalla:



En la opción (4) aparece la siguiente pantalla:



ALTAS.

Usar la opción *Add* del menú para crear un nuevo renglón en la tabla activa. Seguir estos pasos para hacer las altas:

1) Se puede teclear "a" para seleccionar la opción *Add*.

2) Teclear el dato que se desea aumentar en el primer campo de la pantalla.

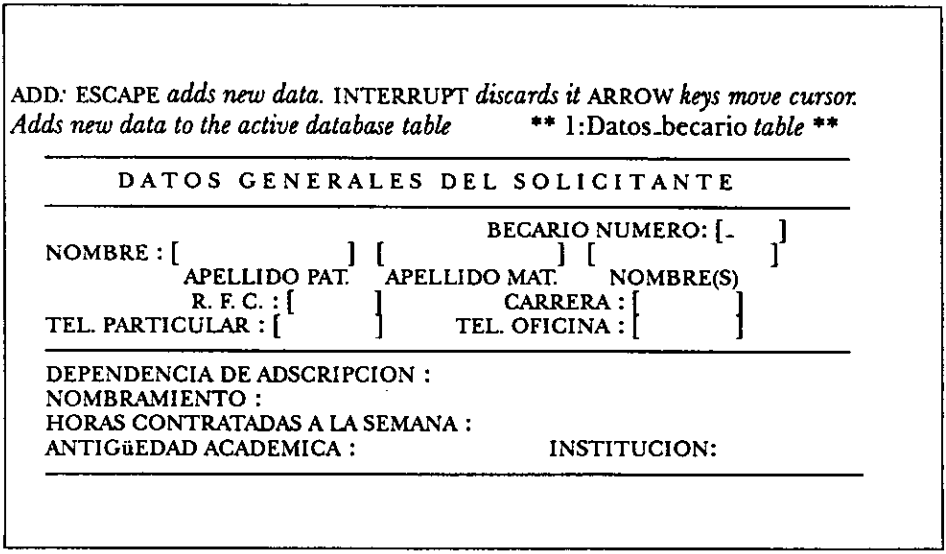
3) Presionar [*Enter*] y el cursor cambia al siguiente campo.

4) Teclear el dato que se desea aumentar.

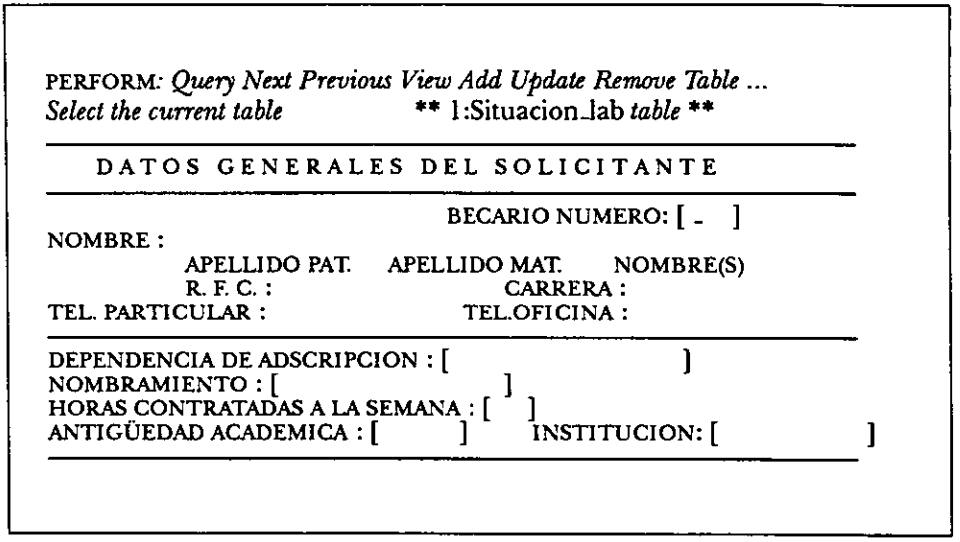
5) Presionar [*Enter*] y el cursor cambia al siguiente campo, teclear el dato deseado.

6) Repetir estos pasos hasta terminar de meter la información deseada.

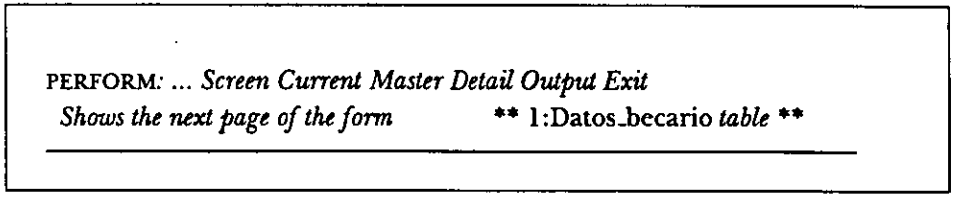
7) Presionar [*Esc*] para aumentar el renglón nuevo a la tabla. Para limpiar los campos en caso de no estar de acuerdo con los datos tecleados se presiona Ctrl-C y presionar [*Interrupt*] para cancelar la opción *Add* sin almacenar el renglón.



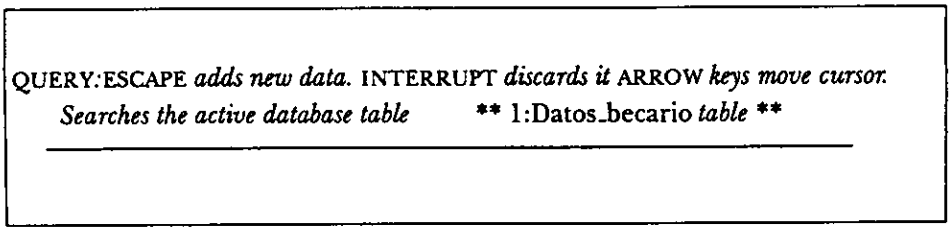
Si en la pantalla se ocupan más de una tabla, después de llenar los datos de la tabla activa, con la opción *Table* se activa la siguiente tabla, quedando algunos campos comunes en la pantalla.



Si la opción cuenta con más de una pantalla, al terminar de teclear los datos de la pantalla teclear [*Esc*] para ir al menú y teclear *Screen* para cambiar de pantalla.



8) Teclear *Query* para buscar si hay datos en la tabla activa y repetir los pasos del (1) al (7).



Al terminar de llenar la pantalla ir al menú y teclear la opción deseada:

ESCAPE(Ejecutar): Aumenta un renglón de datos en la base de datos.

INTERRUP(Interrumpir): Cancela la acción de aumentar datos.

ARROW(Flecha): Permite mover el cursor a cualquier campo utilizando las flechas.

En este caso la tecla *ESCAPE* tiene la función contraria de realizar la acción en lugar de descartarla.

Despúes de teclear cualquier opción regresa al menú de *PERFORM*, donde se puede seleccionar otra opción.

CAMBIOS.

Usar la opción *Update*, para cambiar los datos de un renglón existente.

Sólo se pueden cambiar los datos de un renglón corriente, éste es el que aparece en la pantalla.

Se usa la opción *Query* para desplegar el renglón que se desea actualizar, para seleccionar el renglón se llena el campo con la cadena específica, esto puede ser tecleando toda la cadena o usando asterísco(*) antes o después de una letra para no teclear toda la cadena o el signo de interrogación(?) para sustituir una letra en la cadena, para seleccionar otros datos antes de realizar la búsqueda se teclea Ctrl-C y limpia los campos, la función de búsqueda (*Query*) va y busca todos los renglones que cumplan con la cadena específica y los presentan, las opciones *Next* y *Previous* presentan los renglones siguientes y anteriores al corriente respectivamente.

Una vez que aparece el renglón en la pantalla, se actualiza de la siguiente manera:

1) Teclear "u" para seleccionar la opción *Update* y el cursor se mueve al primer campo de la forma.

UPDATE:ESCAPE *adds new data.* INTERRUPT *discards it* ARROW *keys move cursor.*
Changes this row in the active database table ** 1:Datos_becario *table* **

2) Use las flechas para mover el cursor al campo deseado y cambiar los datos en tantos campos como se requiera.

3) Presiona [*Esc*] para almacenar los cambios en la tabla activa. Si no se desean guardar los cambios, presionar [*Interrupt*] para cancelar la opción *Update*.

Cuando se presiona [*Esc*], modifica el renglón y despliega el siguiente mensaje:

This row has been changed

Es decir este renglón ha sido cambiado.

Cuando se presiona [*Interrupt*], cancela el cambio y despliega un mensaje:

The row remains unchanged

Es decir el renglón no se ha cambiado.

BAJAS.

Al igual que en los cambios primero hay que buscar el renglón a dar de baja con *Query* y usando *Next* y *Previous* vemos los renglones siguientes y anteriores respectivamente.

Una vez que aparece el renglón en la pantalla, se realiza la baja.

Usar la opción *Remove* para borrar un renglón de la tabla activa.

La opción *Remove* sólo afecta la información del renglón corriente que aparece en la pantalla.

1) Teclear "r" para seleccionar la opción *Remove* y en la parte superior izquierda pregunta:

Remove: Yes No

2) Para borrar el renglón teclear *Y*

Removes this row from the active table

Borra el renglón de la tabla activa.

3) Si teclea *Y*, borra el renglón de la tabla y limpia la pantalla, desplegando el siguiente mensaje:

Row deleted

4) Si teclea *N* y regresa al menú anterior.

3. REPORTES.

Regresando al menú principal, seleccionar la opción 2 aparece otro submenú.

Reportes

1. REPORTE DE BECARIOS
2. REPORTE DE BECARIOS POR CARRERA
3. REPORTE POR BECA
4. REPORTE POR INSTITUCION
5. REPORTE POR NACIONALIDAD
6. REPORTE POR ASESOR

Use spacebar, arrow keys, or type number to make selection.
Enter "e" to return to previous menu or exit.
Enter carriage return to execute selection:

Cada reporte pregunta a dónde se envia la información. Teclear la letra de la opción deseada en mayúscula o minúscula. Si tecleas **a** o **A** solicita el nombre del archivo.

Al seleccionar la opción (1) automáticamente aparece:

```
Salida a (P)antalla, (A)rchivo o (I)mpresora : P

==========================================
No. pag: 1                    Fecha: 95/09/13
==========================================

                LISTA DE BECARIOS
                =================

NUMERO  NOMBRE                          CARRERA     TIPO DE BECA
   1    AGUILA MARTINEZ CLAUDIA         ACTUARIA    TESIS DE LIC.
   2    BARAJAS GUZMAN MARIA GUADALUPE  BIOLOGIA    MAESTRIA
   3    CONTRERAS ARIZAGA FABIAN        BIOLOGIA    MAESTRIA

NUMERO TOTAL DE BECARIOS: 3
```

Al seleccionar la opción (2), solicita la carrera que deseas consultar.

```
TECLEAR ACTUARIA, BIOLOGIA, FISICA O MATEMATICAS: BIOLOGIA
Salida a (P)antalla, (A)rchivo o (I)mpresora : P

==========================================
No. pag: 1                    Fecha: 95/09/13
==========================================

            REPORTE DE BECARIOS DE BIOLOGIA
            ===============================

NOMBRE                          TIPO DE BECA  ASESOR

BARAJAS GUZMAN MARIA GUADALUPE  MAESTRIA      ALVAREZ FRANCISCO
CONTRERAS ARIZAGA FABIAN        MAESTRIA      ESPINA SONIA SOFIA

NUMERO TOTAL DE BECARIOS: 2
```

Al seleccionar la opción (3), dar el tipo de beca que se quiere consultar.

```
TECLEAR EL NIVEL DE BECA: EL, TL, ES, RE, EM, TM, ED, O TD: TM
Salida a (P)antalla, (A)rchivo o (I)mpresora : P
==========================================
   No. pag: 1                    Fecha: 95/09/13
==========================================
              REPORTE DE BECARIOS DE MAESTRIA
              ========================

NOMBRE                          CARRERA   ASESOR

BARAJAS GUZMAN MARIA GUADALUPE  BIOLOGIA  ALVAREZ FRANCISCO
CONTRERAS ARIZAGA FABIAN        BIOLOGIA  ESPINA SONIA SOFIA

NUMERO TOTAL DE BECARIOS: 2
```

Nivel de la beca:

EL: Conclusión de estudios de licenciatura.

TL: Tésis de licenciatura.

ES: Especialización.

RE: Requisitos para ingresar a la maestría.

EM: Estudios de maestría.

TM: Tésis de maestría.

ED: Estudios de doctorado.

TD: Tésis de doctorado.

Al seleccionar la opción (4), solicita la institución que se desea consultar.

TECLEAR DGAPA O CONACYT: DGAPA

Salida a (P)antalla, (A)rchivo o (I)mpresora : P

No. pag: 1 Fecha: 95/09/13

REPORTE DE BECARIOS DE DGAPA

NOMBRE	BECA	INICIO	TERMINO
AGUILA MARTINEZ CLAUDIA	TESIS DE LIC.	03 01 1992	
BARAJAS GUZMAN MARIA GUADALUPE	MAESTRIA	01 01 1993	
CONTRERAS ARIZAGA FABIAN	MAESTRIA	03 01 1994	09 30 1995

NUMERO TOTAL DE BECARIOS: 3

Al seleccionar la opción (5), especificar si se desea consultar becas nacionales o extranjeras.

TECLEA EL TIPO DE BECA(NACIONAL O EXTRANJERA): NACIONAL

Salida a (P)antalla, (A)rchivo o (I)mpresora : P

No. pag: 1 Fecha: 95/09/13

REPORTE DE BECARIOS CON BECA NACIONAL

NOMBRE	BECA	INICIO	TERMINO
AGUILA MARTINEZ CLAUDIA	TESIS DE LIC.	03 01 1992	
BARAJAS GUZMAN MARIA GUADALUPE	MAESTRIA	01 01 1993	

NUMERO TOTAL DE BECARIOS: 2

Al seleccionar la opción (6), dar los datos del asesor a consultar.

```
ASESOR.
APELLIDO PATERNO: IBARGÜENGOITIA
APELLIDO MATERNO: GONZALEZ
NOMBRE(S): GUADALUPE

Salida a (P)antalla, (A)rchivo o (I)mpresora : P

==========================================
   No. pag: 1                  Fecha: 95/09/13
==========================================

          Asesor: IBARGÜENGOITIA GONZALEZ GUADALUPE
          Grado Académico: MAESTRIA

                    BECARIOS
                  ==========

NOMBRE                    BECA          CARRERA    INSTITUCION
AGUILA MARTINEZ CLAUDIA   TESIS DE LIC. ACTUARIA   DGAPA

Número total de becarios : 1
```

Después de dar los datos pregunta dónde enviar la información, si se envía a un archivo pregunta el nombre del archivo dónde se va a almacenar el reporte.

```
TECLEA EL TIPO DE BECA A CONSULTAR (NACIONAL O EXTRANJERA): NACIONAL
Salida a (P)antalla, (A)rchivo o (I)mpresora: A
Nombre del Archivo : Reporte1




Press Return to continue
```

Después de teclear *Return* regresa al menú de reportes.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

[1] "Introducción a la ciencia de la computación."
Vladimir Zwass
Compañía editorial Continental S. A. de C. V.
México, D. F. 1985.

[2] "Moderm Operating Systems"
Andrews S. Tanenbaum
Prentice-Hall
1987.

[3] "El entorno de programación UNIX."
Brian W. Kernighan, Rob Pike
Prentice-Hall
1987.

[4] "Bases de datos relacionales."
Rivero Cornelio E.
Parainfo
1988.

[5] "Extending the database relational model to capture more meaning."
Codd E. F.
ACM TODS, 4, 4, (Diciembre 1979) 397-434.

[6] "Is your DBMS really relational?";
"Does your DBMS run by the rules?"
Codd E. F.
Computerworld, October 14, 1985
Computerworld, October 21, 1985

[7] "Futher normalization of the database relational model". En *Data Base Systems*
Courant Computer Science Symposia Series, Vol. 6 Englewood Cliffs
Prentice-Hall
Nueva Jersey, 1972.

[8] "Recent Investigations into relational data base systems" Proc. IFIP Congress, 1974. También en Proc. ACM Pacific Conference, San Francisco,

abril, 1975; se puede obtener con el ACM Golden Gate Chapter P. O. Box 24055, Oakland, California 94623.

[9] "Multivalued dependiences and a new normal form relational database" ACM Transactions on database systems 2, num 3, september, 1977

[10] "What ever happeded to structured analisys?"
Edward Yordon
Datamation, vol. 32, no. 11, pp 133-138
1 de Junio de 1986.

[11] "Introducción à los sistemas de las bases de datos."
Date C. J.
Addison Wesley Iberoamericana
1986.

[12] "Diseño y manejo de base de datos."
Angel Lucas Gómez
Paraninfo
1993

[13] "Using INFORMIX-SQL"
Jonathan Leffler
Addison Wesley Publishing Company
1992

[14] "INFORMIX-SQL."
A tutorial and reference
Tony Lacy-Thompson
Prentice-Hall
1991

[15] "SCO UNIX"
Using Covers Xenix
The LeBlond Group Geoffrey T.
Osborne McGraw-Hill

[16] "Handbook of relational database desing"
Candance C. Fleming
Barbara Von Halle
Addison-Wesley
1989