UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA PARA LA AUTOMATIZACION DE ESTUDIOS SOCIOECONOMICOS

T E S I S

Que para obtener el Titulo de:

INGENIERO EN COMPUTACION

Presentan:

MARIA DEL SOCORRO ARENAS PEÑARROJA

VERONICA CARDENAS SANCHEZ

ENRIQUE VILLANUEVA VALDIVIA

ANGEL MUÑOZ ANDRADE

HUGO SERNA KUBLI

Director de Tesis: Ing. Alberto Templos Carbajal

Cd. Universitaria, D. F. 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

A la Universidad Nacional Autónoma de México
por la formación profesional que nos brindó.

Con profundo y eterno cariño, gratitud y en reconocimiento a la comprensión, ayuda y estímulos que sin escatimar esfuerzos y sacrificios durante nuestros estudios y a través de la vida nos han proporcionado e indicado con su ejemplo los caminos de la honradez y rectitud en el cumplimiento de nuestras obligaciones; dedicamos la presente tesis a nuestros padres.

A nuestros profesores y amigos por su apoyo.

Al Ing. Alberto Templos Carbajal por todo el apoyo brindado.

María del Socorro Arenas Peñarroja

Verónica Cárdenas Sanchez

Enrique Villanueva Valdivia

Angel Muñoz Andrade

Hugo Serna Kubli

SISTEMA PARA LA AUTOMATIZACIÓN DE ESTUDIOS SOCIOECONÓMICOS

ÍNDICE

INTRODUCCIÓN

La meta de todo departamento que desarrolla sistemas de información, es el de realizarlos de la manera mas eficiente. Para cumplir este objetivo, los sistemas de información deben satisfacer las necesidades reales de la empresa.

La clave del desarrollo verdaderamente productivo es el uso de herramientas correctas e integradas, disponibles para todas las actividades del ciclo de vida del sistema, que permitán generar mejores aplicaciones y como consecuencia sistemas adaptables cien por ciento a las necesidades del usuario.

El presente trabajo realizará un enfoque sistemático y secuencial del desarrollo de software (Ciclo de Vida del Sistema) que comprende las etapas de Análisis, Diseño e Implantación, aplicado a un caso específico, (Figura. I.1). El objetivo principal del proyecto es desarrollar un sistema integral que sea capaz de responder eficazmente a las necesidades de información y funcionalidad de una empresa cuya labor principal es el realizar estudios socioeconómicos.

En el primer capítulo se hablará de una manera muy general de la evolución que ha tenido el software hasta nuestros días, haciendo hincapié en los lenguajes de cuarta generación. Después se hará una reflexión de la importancia del uso de prototipos para auxiliar el desarrollo de sistemas y por último se realizarán algunas observaciones de la importancia de la selección y evaluación de las herramientas de software.

En la etapa de Análisis se establecen los requisitos de todos los elementos del sistema. Esto se logra con un análisis detallado de todos los procesos que integran la operación de las áreas a automatizar; esta recopilación de requisitos se logrará a través de una serie de entrevistas con la dirección o el usurario.

Entre las herramientas que se tienen para llegar a determinar un modelo que especifique los requisitos del sistema de programación se encuentran: diagramas de flujo de datos, diccionario de datos, diagrama de funciones, por mencionar algunas. Esto permitirá establecer la naturaleza de los programas a construir y determinar el dominio de la información de que hará uso el software.

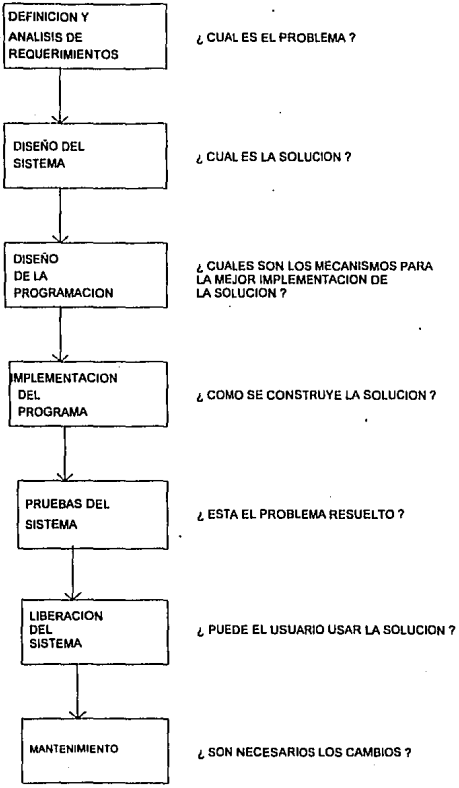


Figura 1.1 Actividades en el desarrollo de sistemas

Todo lo anterior tiene la finalidad de hacer uso de una metodología que permita ir de un modelo lógico a uno físico y de un diseño de arriba hacia abajo. Todo esto permite analizar el flujo de la información del proceso , y de esta manera llegar a un diseño que permita obtener los resultados esperados.

En la etapa de Diseño se describen las estructuras de datos a utilizar, los métodos de acceso, la arquitectura de las aplicaciones, el detalle procedimental y la caracterización de la interfase. Para realizar lo anterior se utilizará una herramienta de cuarta generación llamada CASE en el nivel Upper CASE y Lower CASE (ORACLE).

Las herramientas CASE de esta categoría dan soporte a las distintas fases del ciclo de vida de un proyecto: planificación del proyecto, definición de los requerimientos, análisis , diseño, definición de pruebas, codificación, gestión de la configuración del software y mantenimiento.

Atendiendo a la fase del ciclo de vida que cubren, existen dos tipos de productos CASE:

Producto de alto Nivel, llamados Upper CASE, CASE superior o CASE Front-End.

Son productos que cubren las primeras fases del ciclo de vida, planificación, análisis y diseño. La ventaja de este nivel es que pueden correr en equipos de cómputo independientes a la computadora donde se vaya a ejecutar la aplicación, aunque la codificación debe desarrollarse de forma independiente, usando la documentación generada por la herramienta. Obteniendo beneficios considerables en la documentación gráfica y en la integración de funciones y relaciones.

Productos de bajo nivel, llamados también Lower CASE, CASE inferior o CASE Back - End.

Son productos basados en el uso de la propia máquina a la que se destina la aplicación y están orientados a la generación de bases de datos, por programas y de archivos, dando soporte a la parte final del ciclo de vida, la codificación y las pruebas.

En la etapa de Implementación, el sistema se llega a crear de forma física. Esta fase termina con la realización de una prueba que se centra en la lógica de los programas (comprobando todos los componentes y procedimientos), en la eficiencia de los flujos de datos y en la interfase con el usuario , permitiendo con esto la verificación de la lógica del software y además, ver si los datos de entrada producen los resultados esperados.

CAPÍTULO 1 DESARROLLO DE UN SISTEMA DE INFORMACIÓN EMPLEANDO LENGUAJES DE 4a. GENERACIÓN.

Los sistemas de información han evolucionado conforme la tecnología computacional avanza. El desarrollo tecnológico en hardware y software en los casi 50 años de historia de la informática ha sido impresionante.

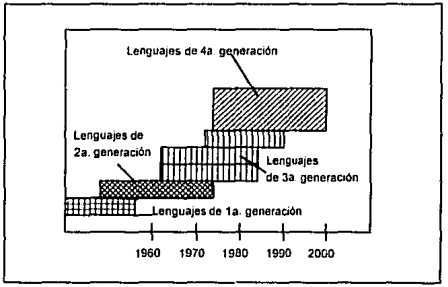


Figura 1.1 Etapas de evolución del software

Esta evolución puede apreciarse más fácilmente si se consideran las características de las cuatro generaciones (Figura 1.1)

Primera generación de los lenguajes: la primera generación de lenguajes se remonta a los días en que se codificaba a nivel máquina. Todavía continúan llevándose a cabo bastantes trabajos con lenguajes de primera generación. El código máquina y su equivalente más humanamente legible, el lenguaje ensamblador, representan la primera generación de lenguajes. Estos lenguajes dependientes de la máquina muestran el menor nivel de abstracción con el que se puede representar un programa.

Existen tantos lenguajes ensambladores como arquitecturas de procesadores con sus correspondientes conjuntos de instrucciones. Desde un punto de vista de la ingeniería del software, esos lenguajes sólo se deben usar cuando un lenguaje de alto nivel no satisfaga los requisitos o no esté disponible.

Segunda generación de lenguajes: La segunda generación de lenguajes fue desarrollada a finales de los años 50 y principios de los 60 y ha servido como base para todos los lenguajes de programación modernos (tercera generación). La segunda generación de lenguajes está caracterizada por su amplio uso, la enorme cantidad de bibliotecas de software y la gran familiaridad y aceptación. Nadie pone en duda que FORTRAN, COBOL, ALGOL y BASIC son lenguajes de base, debido a su madurez y aceptación.

Tercera generación de lenguajes: Los lenguajes de tercera generación (también llamados lenguajes de programación moderna o estructurada) están caracterizados por sus potentes posibilidades procedimentales y de estructuración de datos. Los lenguajes de esta clase se pueden dividir en tres categorías: lenguajes de alto nivel de propósito general (PASCAL, Modula 2 y C), lenguajes de alto nivel orientados a los objetos (C++, Smaltalk y Eiffel) y los lenguajes especializados (LISP, PROLOG y FORTH).

Lenguajes de cuarta generación (L4G): Los lenguajes de cuarta generación, al igual que los lenguajes de inteligencia artificial, contienen una sintaxis distinta para la representación del control y para la representación de las estructuras de datos. Sin embargo, un L4G representa estas estructuras en un mayor nivel de abstracción, eliminando la necesidad de especificar los detalles algorítmicos.

1.1 DEFINICIÓN Y GENERALIDADES

Los lenguajes de cuarta generación se dividen en dos categorías:

Lenguajes de petición: hasta ahora, la gran mayoría de los L4G se han desarrollado para ser usados conjuntamente con aplicaciones de bases de datos. Tales lenguajes de petición permiten al usuario manipular de forma sofisticada la información contenida en la base de datos previamente creada.

Generadores de programa: los generadores de programas son otra clase de L4G, aunque algo más sofisticada. Más que basarse en una base de datos predefinida, un generador de programas permite al usuario crear programas en un lenguaje de tercera generación usando notablemente menos sentencias. Estos lenguajes de programación de muy alto nivel hacen un gran uso de la abstracción de datos y procedimientos.

Aunque los lenguajes de petición y los generadores de programas son los L4G más comunes, existen otras categorías. *Los lenguajes de soporte a la toma de decisiones* permiten que los no programadores lleven a cabo una gran variedad de análisis, que van desde los simples modelos de hoja de cálculo bidimensionales hasta los sofisticados sistemas de modelos estadísticos y de investigación operativa. *Los lenguajes de prototipos* se han desarrollado para apoyar el uso de prototipos facilitando la creación de interfases de usuario y de diálogos, además de proporcionar medios para la modelización de datos.

Características fundamentales de un lenguaje de cuarta generación:

- Es diseñado para una clase particular de aplicaciones: manejo de bases de datos. Por ello puede anticiparse a lo que el usuario quiere hacer en los programas y es menos complejo que los lenguajes de propósito general como PASCAL, C, COBOL, etc. Un programa escrito en un lenguaje de cuarta generación tiene muchas menos instrucciones que uno escrito en un lenguaje de propósito general.

- Incorpora ventajas de lenguajes procedimentales y no procedimentales. Un *lenguaje procedimental* es aquel en el que se especifica como hacer algo paso a paso, mientras que en un *lenguaje no procedimental* se especifica el resultado final y el lenguaje proporciona el procedimiento. Como ejemplo considérese el caso de leer información de la pantalla. En un lenguaje procedimental hay que especificar paso a paso en que orden deben leerse los datos, en qué posición de la pantalla y verificar si son del tipo adecuado además de ver si se dieron valores válidos. En el no procedimental sólo se usa una instrucción especial y se especifican los valores a leer. El lenguaje se encarga de pedir los datos, permite editarlos y verificar si son del tipo correcto (según se especifico previamente en la forma de captura).

1.2 RAZONES PARA EL USO DE PROTOTIPOS.

El ciclo de vida es el modelo más antiguo y más ampliamente utilizado en la ingeniería de software; sin embargo, tiene algunas debilidades entre las que se pueden citar:

1.- Los proyectos reales raramente siguen el flujo secuencial que propone el modelo.
2.- Es difícil obtener del cliente los requisitos exactos del sistema en la etapa de análisis.
3.- El cliente debe tener paciencia ya que hasta las últimas etapas del desarrollo se puede ver una aproximación del producto final.

Para atacar estas debilidades se han creado otros modelos de desarrollo más completos y complejos; sin embargo, la estructura fundamental del desarrollo sigue siendo la misma.

El modelo de *prototipos* para el desarrollo de software intenta cubrir algunas de las deficiencias ya comentadas del modelo del ciclo de vida. La figura 1.2 muestra la secuencia de sucesos del modelo.

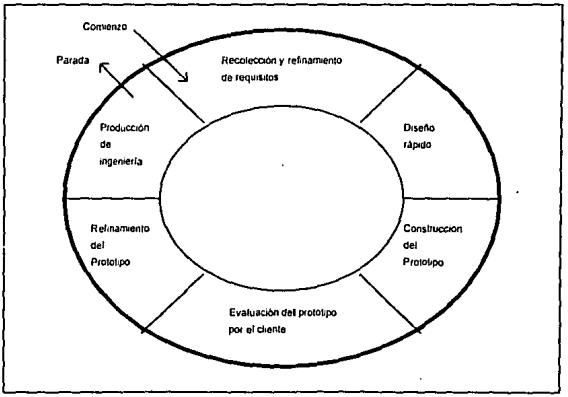


Figura. 1.2.- Modelo de Creación de prototipos

Como en todos los modelos, la construcción de prototipos comienza con la recolección de requisitos realizada en la etapa de análisis. El ingeniero de software y el cliente definen los objetivos globales del producto, identifican todos los requisitos conocidos y perfilan las áreas donde será necesario una mayor definición. Luego se produce un diseño rápido que se enfoca principalmente a los aspectos de software visibles al usuario. Basados en este diseño se crea un *prototipo* que es evaluado por el cliente y ayuda a refinar los requisitos del producto. Se produce un proceso iterativo en el que el *prototipo* es refinado para que satisfaga las necesidades del cliente, al mismo tiempo que facilita al que lo desarrolla una mejor comprensión de lo que hay que hacer.

La clave para el éxito en este modelo está en definir al comienzo las reglas del juego; esto es, el cliente y el ingeniero deben estar de acuerdo en que el *prototipo* se construya para servir sólo como un mecanismo de definición de requisitos para posteriormente, desarrollar el producto real.

1.3 CONSIDERACIONES PARA LA EVALUACIÓN Y SELECCIÓN DE UN LENGUAJE DE 4a. GENERACIÓN.

Cuando comienza un proyecto, uno de los factores que hay que tener en cuenta es el de los recursos informáticos necesarios para llevarlo a cabo: los recursos hardware, los recursos software y los recursos humanos. Los recursos de hardware son:

* El ordenador y máquinas donde se realice el desarrollo del nuevo sistema.
* El ordenador y máquinas destino (donde se vaya a ejecutar el nuevo sistema).

Los recursos humanos se estiman empleando diversos métodos de evaluación del esfuerzo de desarrollo para obtener el número de personas-mes, o personas-año.

Los recursos del software, que a menudo se descuidan durante la planificación, suelen ganar importancia durante las fases siguientes, aunque es recomendable especificar cuanto antes los requerimientos de recursos del software, de este modo se podrá realizar una selección oportuna.

Se pueden seleccionar tres tipos de herramientas para el software: herramientas orientadas al código, herramientas de cuarta generación, y herramientas CASE.

Herramientas orientadas al código

* Compiladores
* Utilerías específicas que acompañan a los compiladores
* Editores
* Ligadores
* Cargadores
* Depuradores

Herramientas de cuarta generación

Las herramientas de cuarta generación permiten especificar las características de una aplicación a muy alto nivel, y realizan la generación de código automático basándose en esas especificaciones. Se emplean lenguajes de cuarta generación no procedurales para la creación de informes, definición de pantallas, de menús, etc. También incorporan un lenguaje de consulta y manipulación de datos (SQL, *Structured Query Language*). A continuación se mencionan los tipos más comunes de herramientas:

* Generadores de código
* Herramientas de bases de datos

Herramientas CASE

Las herramientas de esta categoría dan soporte a las distintas fases del ciclo de vida de un proyecto: planeación del proyecto, definición de los requerimientos, análisis, diseño, definición de pruebas, codificación, gestión de la configuración del software y mantenimiento.

* CASE de alto nivel (soporte al análisis y diseño)
* CASE de bajo nivel (soporte a la codificación, pruebas y mantenimiento)
* Herramientas IPSE (*Integrated Project Support Environment*)
* Sistemas expertos
* CASE integrado
* Herramientas de soporte IRP (*Information Resource Planning*)
* Herramientas de Reingeniería

Cuando ya se hayan definido todas las necesidades, hay que comenzar con la selección de las herramientas que sean las más adecuadas. Por supuesto, es imprescindible tener la certeza de que el problema se puede solucionar empleando alguna herramienta. Una vez decidido esto, hay que estudiar la herramienta más idónea para modelar el nuevo sistema, así como valorar los costos y requerimientos de hardware y software de la herramienta.

Para elegir la forma adecuada del producto es necesario tener en cuenta sus cualidades técnicas, para ello será necesario tener una lista de comprobación con la características principales de las herramientas que se evalúen.

Se deberán considerar los siguientes aspectos:

* Estabilidad del fabricante. Se prevé que dentro de unos años van a desaparecer muchos fabricantes menores.
* Capacidad del vendedor para dar el soporte y formación, tanto sobre la herramienta como sobre la metodología.
* Precio del producto. Hay que tener en cuenta que se prevé una disminución general de los precios dentro de los próximos años.
* Plataformas Hardware que admite la herramienta.
* Calidad de la documentación adjunta a la herramienta.
* Metodologías y técnicas soportadas.
* Interfaz gráfico (facilidad de utilización, si se aprende rápidamente, si se puede personalizar, si ayuda en los errores, si el nivel de ayuda es un diagnóstico o si es correctivo, etc.)
* Interfaz con otras herramientas. El grado de apertura hacia otras herramientas.
* Generación automática de código (para qué plataformas, etc.)

Después de lo anterior, se debe mencionar que cada aplicación tiene sus puntos fuertes y débiles. Sin embargo, el objetivo del analista deberá concentrarse en seleccionar la herramienta de software que permita la realización de las funciones del sistema con el costo mas bajo posible.

Por lo anterior, surge la pregunta ¿ Qué tipo de herramienta deberá seleccionarse para este proyecto?

La herramienta de software necesaria para desarrollar el sistema debe permitir un ambiente en el que sea posible guardar y recuperar información de la base de datos en forma conveniente y eficiente. Todo esto lo permite un Manejador de Base de Datos (DBMS).

Dentro de los Manejadores encontramos los de red, jerárquicos y los relacionales. Teniendo en cuenta la topología de los DBMS anteriores la que más ventaja ofrece es la del modelo relacional. Ya que además de garantizar una seguridad en el manejo de los datos nos proporcionan un óptimo mantenimiento de la información, asegurando con esto una mayor confiabilidad del sistema.

Dentro de los manejadores relacionales se encuentra ORACLE, INGRES, DB2 e INFORMIX-4GL. De esta gama de manejadores se seleccionó INFORMIX-4GL, el cual permitirá realizar las funciones del sistema a un costo más accesible.

CAPITULO 2 ANÁLISIS DEL PROYECTO

En esta etapa se pretende especificar de manera total y consistente los requerimientos del sistema, (figura 2.1) para lo cual es necesario el empleo de métodos para la especificación de requerimientos.

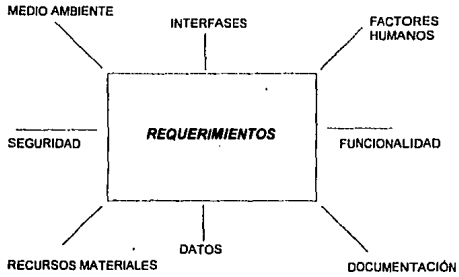


Figura 2.1 Diagrama de Requerimientos

Previamente se analizarán las necesidades del cliente, con lo que se llega a establecer la existencia del problema a solucionar. Para ello es necesario realizar una serie de pasos que permitan tener una visión del área que se investiga.

El análisis del proyecto incluye el descubrimiento, refinamiento, modelado y especificación del software que se desarrollará como resultado de esta etapa.

La etapa de análisis de este proyecto esta integrada de los siguientes procesos:

- COMENTARIOS GENERALES DE LA ACTIVIDAD DE LA EMPRESA.
- COMENTARIOS GENERALES DEL PROYECTO.
- CALENDARIO DE ACTIVIDADES PARA CADA UNA DE LAS ETAPAS.
- DEFINICIÓN Y ESTRUCTURACIÓN DE REQUERIMIENTOS DE USUARIO.
- DISEÑO TÉCNICO GENERAL.

El resultado de estos procesos permitirá generar el Documento de Análisis.

A continuación se desarrollarán cada uno de los procesos que integran la etapa de análisis para el proyecto de estudios socioeconómicos.

2.1 COMENTARIOS GENERALES DE LA EMPRESA

Actualmente las empresas enfrentan una economía difícil, lo que ha provocado fuertes reducciones de personal con la finalidad de minimizar sus altos costos en nómina y por ende sanear su situación económica. Por este motivo, la empresa que requiere este sistema ofrece sus servicios en materia de RECURSOS HUMANOS permitiendo tener personal de tiempo completo, sin la necesidad de que una empresa en particular lo contrate directamente, evitando así la responsabilidad patronal. A continuación se da una descripción general de las funciones de la empresa.

La organización interna de la empresa la conforman: un director general, dos socios directores y tres departamentos, dichos departamentos son: ADMINISTRACIÓN DE NÓMINA, BUSQUEDA-SELECCIÓN DE PERSONAL Y ESTUDIOS SOCIOECONÓMICOS LABORALES. (Figura 2.2)

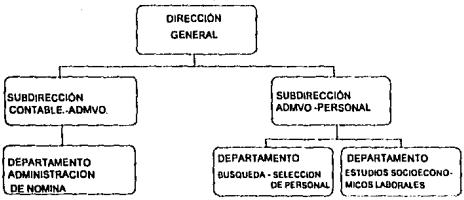


Figura 2.2 Organigrama de la Empresa

Dado que la empresa en cuestión presta sus servicios a otras empresas, a estas últimas se les llamará empresa-cliente.

DEPARTAMENTO DE NÓMINA.

El principal objetivo de este departamento es el de relevar a la empresa-cliente de costos fiscales, laborales y administrativos, inherente a la contratación del personal temporal no sindicalizado. Por lo que se deja la total responsabilidad patronal y de seguridad social en manos de la empresa.

La empresa-cliente puede reclutar, seleccionar al candidato, enviarlo para ser contratado y darlo de alta como empleado de la empresa por el tiempo y sueldo establecido por la empresa-cliente, o bien esta empresa realizará dicho proceso.

Para los siguientes casos se utilizará el servicio de Administración de Nómina:

-Proyectos especiales o cargas de trabajo. Las contrataciones para períodos cortos o inciertos, se puede canalizar a esta empresa, sin necesidad de incrementar la nómina de la empresa-cliente o hacer extensivo el paquete de prestaciones que tenga ésta.

-Personal en periodos de prueba. La empresa evita de manera inicial que durante un período la empresa-cliente tenga que otorgar las prestaciones con las que cuenta actualmente, así mismo, esta evaluará si es el candidato idóneo para otorgarle la planta.

-Personal por honorarios.- Este puede ser integrado al servicio de Administración de Nómina, el cual le dará la seguridad de la responsabilidad laboral, así como la de que el personal se encuentre protegido ante cualquier accidente de trabajo.

-Reubicación de su personal.- La empresa-cliente puede reubicar ex-empleados para trabajos especiales que requieren de este tipo de personal, que ha sido entrenado y capacitado por la empresa-cliente.

-Problemas de cálculo de personal.- Le permite a la empresa-cliente seguir con el nivel de personal necesario, sin tener que reflejarlo en su nómina.

Ventajas del servicio:

- Presentar el gasto como servicios externos deducibles.
- Ahorro en prestaciones adicionales.
- Contar con el número de personas adecuadas con el fin de mantener un nivel óptimo de productividad.

DEPARTAMENTO DE BÚSQUEDA Y SELECCIÓN DE PERSONAL

Los servicios especializados para la atinada selección de personal que concuerde con sus objetivos proporciona una valiosa ayuda a las empresas-cliente; estas aún contando con recursos muy completos, no siempre pueden localizar en tiempo limitado al personal idóneo para sus puestos vacantes. Este servicio esta enfocado a posiciones de alta dirección, mandos gerenciales, secretarial ejecutivo y administrativos.

DEPARTAMENTO DE ESTUDIOS SOCIOECONÓMICOS LABORALES

En este departamento se lleva a cabo la realización de estudios socioeconómicos para la selección de personal, lo cual permite conocer a el candidato desde un punto de vista social, familiar, económico y laboral, permitiendo disminuir los riesgos en la admisión del candidato a la empresa-cliente.

Dado que este departamento es el objeto de estudio para el diseño del proyecto, es necesario comprender el proceso para la realización de un estudio socioeconómico, el cuál comprende lo siguiente:

1.- El cliente solicita el estudio (recepción de orden) el cual involucra la siguiente información: La empresa que lo solicito, el tipo de estudio y los datos necesarios para hacer el contacto con el investigado.

2.- Se asigna a un encuestador (investigador) el estudio, el encuestador deberá hacer contacto con el investigado para fijar fecha y hora de visita a su domicilio.

3.- El encuestador estando en el domicilio del investigado, deberá hacer una serie de preguntas para llenar el formato de visita. El formato deberá recabar la siguiente información:

- Área de datos personales
- Área laboral
- Área de integración económica familiar.

4.- El encuestador debe verificar los datos del área laboral de los tres últimos trabajos, obtenidos de la entrevista. La verificación se realiza haciendo contacto con los departamentos de recursos humanos de cada una de las empresas en las que el investigado trabajó.

5.- El encuestador debe realizar resúmenes (por visita, y de cada empresa que se verificó). Estos resúmenes así como el formato de visita pasan al departamento de procesamiento donde se hace la captura de toda la información.

6.- El informe debe pasar al departamento de corrección de estilo, donde se corrige la redacción y ortografía.

7.- Una vez que se ha verificado la información, corregido el estilo y verificados los puntos anteriores, se hace entrega del estudio al cliente.

2.2 COMENTARIOS GENERALES DEL PROYECTO

Una fuente de ingreso de la empresa la constituye la realización de estudios socioeconómicos. La elaboración de estos se realiza en un tiempo considerable ya que comprende desde la emisión de la orden (recepción de orden); qué cliente solicita el estudio; la asignación del estudio a un investigador, la realización de una entrevista con el investigado del cuál el cliente está solicitando información; la aprobación de este estudio y finalmente la entrega del estudio al cliente. Cada estudio realizado involucra consultas y almacenamiento de información. Además de que el número de estudios a realizar va en aumento, lo que impide controlar la información generada. Como consecuencia de esto, la empresa decidió realizar *la automatización de dichos estudios socioeconómicos.*

Con el sistema a desarrollar se pretende dar un mejor servicio a los clientes y llevar el control adecuado y eficiente del proceso de estudios socioeconómicos.

De la información obtenida del punto 2.1, y de lo comentado en el párrafo anterior, a continuación se establece el objetivo que persigue la creación del sistema, así como los módulos de que se conformará.

A) OBJETIVO DEL PROYECTO.

Diseñar e implementar un sistema de información empleando un lenguaje de cuarta generación que permita automatizar estudios socioeconómicos y ayude a la toma de decisiones de manera rápida y oportuna.

B) FUNCIONES GENERALES DEL SISTEMA.

1.- RECEPCIÓN DE ORDEN.

2.- ESTUDIOS.

3.- CATÁLOGO DE EMPRESAS.

4.- CATÁLOGO DE CLIENTES

5.- CATÁLOGO DE INVESTIGADORES.

6.- NÓMINA.

C) SEGURIDAD Y REQUERIMIENTOS ESPECIALES.

La seguridad es una tarea fundamental en un sistema de información, por lo que esta deberá cubrir los siguientes objetivos:

- Cumplir estrictamente con las leyes para asegurar la privacía del cliente.
- Proteger registros financieros y del personal sensible.
- Reducir la probabilidad de espionaje industrial.

Con esto se pretende que personas no autorizadas tengan acceso a la información del sistema.

Dentro de los requerimientos especiales en el proceso de captura del estudio, toda la información capturada tendrá que ser almacenada en un disco flexible para posteriores aclaraciones. Además se desea que la información que compone el área que corresponde a la experiencia laboral ingrese al archivo de empresas y a los archivos de referencias.
Cabe hacer notar que este sistema tiene contemplado crecer a futuro, dadas las necesidades de manejo de información rápida y eficaz, por lo que se necesitará la instalación de una red de computadoras, y por tanto tendremos otra medida de seguridad dadas las opciones que presenta el sistema en red, por ejemplo, NOVELL ver. 3.1.

D) INTERFASES DEL SISTEMA.

La interfase deberá permitir que el usuario opere clara y fácilmente el sistema, haciendo uso de la pantalla y el teclado de tal forma que el usuario pueda introducir, validar y procesar los datos, así como recuperar información del sistema.

La interfase del usuario orientada por menús es la utilizada en este proyecto.

Dentro del sistema tenemos interfases de comunicación vía software, por ejemplo la implementación de menús, pantallas de trabajo (reportes, consultas), etc.

2.3 CALENDARIO DE ACTIVIDADES PARA CADA UNA DE LAS ETAPAS

Dirigir un proyecto complejo puede resultar difícil. A menudo, el mejor método es fraccionar el proyecto en un conjunto de tareas menores, que sean mas fáciles de controlar y dirigir. El peligro de tal división es el de concentrarse en las actividades o tareas individuales y por consiguiente, perder de vista la globalidad del proyecto. Se necesita pues, una herramienta válida para realizar la subdivisión en tareas o actividades más pequeñas y a la vez, conservar la visión general de proyecto. PERT y CPM son dos de las mejores técnicas conocidas, las cuáles se explican a continuación:

PERT (Program Evaluation and Review Techniques), es una técnica de revisión y evaluación de programas. Es particularmente útil en los proyectos de investigación y desarrollo, donde el tiempo que se requiere para completar las diversas actividades tiene un alto grado de incertidumbre.

La técnica CPM (Critical Path Method), método de ruta crítica , es mas útil en las situaciones donde la duración del proyecto puede controlarse como el incremento o decremento de los recursos.

En general, el PERT se debería usar cuando el tiempo de las actividades son extremadamente cortas, y el CPM cuando la experiencia previa permite que los tiempos estén relativamente bien definidos. La clave de PERT Y CPM consiste en dividir el proyecto en etapas lógicas o actividades, y luego ordenar esas actividades en una secuencia.

De lo anterior se desprende que, en todo proyecto es necesario calendarizar las actividades que lo componen . Por lo que se han puesto en el mercado una gama extensa de herramientas gráficas, entre las que se encuentra SUPERPROJECT. A continuación se explicará en que consiste dicha herramienta.

El Superproject es un programa para computadora que auxilia en la administración de proyectos de cualquier índole.

En el contexto de Superproject, un proyecto es una serie de tareas relacionadas, que de llevarse a cabo (algunas en secuencia, otras simultáneamente) alcanzaran un objetivo específico.

La administración de proyectos es un proceso que tiene los siguientes pasos:

- Definir el objetivo del proyecto.
- Identificar y definir el itinerario de tareas que forman parte del proyecto.
- Asignar los recursos (humanos y materiales) necesarios para llevar a cabo cada una de las tareas.
- Llevar el registro del avance de cada una de las tareas.
- Evaluar y ajustar el itinerario del proyecto.

Todo ello con las restricciones de:

Lograr el tiempo más corto posible.
A costo mínimo.
Y con resultados que reúnan o excedan los requerimientos.

La principal función del Superproject es, obtener los resultados deseados en el tiempo especificado, sin exceder el presupuesto fijado.

En que ayuda Superproject?

En la definición de las actividades del proyecto, en el establecimiento del orden en que deben sucederse unas a otras, cuánto tiempo duraran, cuándo empiezan y cuándo terminan, etc.

Posteriormente se puede evaluar la factibilidad de que el proyecto concluya en el tiempo esperado con los recursos disponibles.

Con Superproject se puede ver como afecta a el proyecto los cambios en los recursos, el tiempo requerido, fecha de comienzo y otros factores que pueden afectar el itinerario.

Algo que debe recordarse siempre es que un proyecto consta de cuatro elementos fundamentales: tareas, tiempo, dinero y recursos; Los cuales habrá que especificar uno por uno, asignar recursos, establecer fechas, etc.

Antes de empezar se debe tener en mente que para aplicar Superproject acertadamente se deberá llevar a cabo la planeación del proyecto, lo cual significa que se debe definir el objetivo y desarrollar la estrategia para lograrlo.

Existen dos tipos de planeación, la estratégica y la táctica. En la primera se define el objetivo del proyecto, tomando en cuenta la política y procedimientos propios de la empresa en relación a el medio ambiente: social, económico, tecnológico, etc.

Dentro del marco de la planeación estratégica se lleva a cabo la planeación táctica, que consiste en especificar los detalles del proyecto, como son: quién hace qué, por cuánto tiempo, a qué costo y en qué orden.

Por lo tanto los pasos básicos de la planeación son:

- Definir el objetivo clara y concisamente.
- Fraccionar el proyecto en fases.
- Fraccionar las fases del proyecto en tareas.
- Asignar los recursos necesarios a cada tarea.
- Estimar cuanto tiempo requiere cada tarea.
- Establecer el itinerario en que sucederán las tareas.
- Determinar si el costo final y las fechas de terminación de las tareas y el proyecto son aceptables, de ser necesario nuevamente planifique.
- Comunicar el plan a las personas involucradas en el proyecto.

Superproject ofrece cuatro vistas diferentes del mismo proyecto, estas son:

GRÁFICA PERT .- Secuencia de ejecución de tareas.
Nombre de opción en el menú View: Pert Chart.

BOSQUEJO.- Bosquejo del proyecto.
Nombre de opción en el menú View: Outline.

GRÁFICA DE GANTT.- Duración de tareas.
Nombre de opción en el menú View: Gantt

GRÁFICA JERÁRQUICA.- Estructura del proyecto por fases .
Nombre de opción en el menú View: Work Breakdown Chart.

Cada una de las vistas presenta al proyecto desde diferentes perspectivas.

A continuación se muestran los resultados obtenidos, después de haber utilizado Superproject Plus al proyecto.

Este proyecto se dividió en las siguientes fases con sus respectivas tareas:

a) Análisis

- a.1) Reconocimiento del problema y generalidades del proyecto.
- a.2) Requerimientos del usuario.
- a.3) Diseño técnico general.
- a.4) Documento de análisis.

b) Diseño Detallado

- b.1) Modelado de datos.
- b.2) Diseño de la base de datos.
- b.3) Especificación de programas.

c) Implantación del sistema

- c.1) Instalación
- c.2) Pruebas

Asignación de los recursos necesarios a cada tarea, estimación de tiempo para cada tarea, y establecimiento del itinerario en que sucederán las tareas:

	Duración en días	**Secuencia**
Fase 1 Análisis (Inicio)		1
1.- Reconocimiento del problema y generalidades del proyecto.	6	2
2.- Requerimientos del usuario.	8	3
3.- Diseño técnico general.	20	4, 5, 6
4.- Documento de análisis.	6	5, 6
Fase 2 Diseño Detallado		
5.- Modelado de datos.	7	6
6.- Diseño de la base de datos.	7	7
7.- Especificación de programas.	26	9
Fase 3 Implantación		
8.- Instalación.	10	9
9.- Pruebas.	10	(Final)

Gráfica de Gantt

El diagrama de Gantt nos muestra como se distribuyó el tiempo para cada una de las tareas del proyecto.

Cada barra nos indica el tiempo en días que se llevará el realizar las etapas del proyecto. A continuación se realiza una comparación del tiempo que tardaría en desarrollarse el proyecto con y sin planeación:

a) Proyecto con Planeación: El tiempo total que ocupará el proyecto es de 105 días (Figura 2.3).

b) Proyecto sin Planeación: El tiempo total que ocupará el proyecto es de 150 días, aprox., tomando en consideración la variedad de factores externos, que tiene cualquier desarrollo de sistemas.

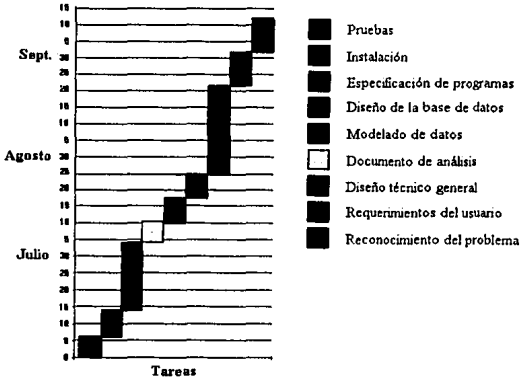


Figura 2.3 Gráfica de Gantt para el desarrollo del sistema.

Como se observa es importante la planeación de cada una de las tareas ya que nos ayuda a prever cualquier tipo de anomalía y en su caso, darle una solución en el tiempo comprendido para cada tarea.

2.4 DEFINICIÓN Y ESTRUCTURACIÓN DE REQUERIMIENTOS DEL USUARIO

La definición de requerimientos implica establecer las características que debe tener el sistema a desarrollar y al mismo tiempo satisfacer las necesidades del usuario.

El analista deberá construir un modelo lógico con estos requerimientos, que sirva de base para las etapas siguientes en el desarrollo del sistema. Auxiliándose de herramientas como: Diagramas de Flujo de Datos, Diagramas Jerárquicos de Funciones, etc.

A) DIAGRAMA DE FLUJO DE LAS ÁREAS INVESTIGADAS.

El objetivo del diagrama de flujo de datos es desarrollar gráficamente la transformación sucesiva de datos a lo largo de un conjunto de procesos.

Los diagramas de flujo de datos especifican las fuentes y destinos de los datos, su almacenamiento, transformaciones y los flujos entre ellos. Un diagrama de flujo de datos (DFD) conduce a la construcción del modelo lógico del sistema.

Un DFD emplea cuatro símbolos básicos: un cuadro define un origen o destino de los datos. Un rectángulo con puntas redondeadas o un círculo representa un proceso que transforma datos. Un rectángulo abierto por un extremo es un almacén de datos y por último la flecha representa un flujo de datos.

Se puede empezar a construir un DFD en el nivel más alto, desde esta perspectiva el sistema se muestra como un único proceso identificando los orígenes y destinos de los datos (también se le conoce como modelo fundamental del sistema).

Desde el punto de vista técnico el DFD de alto nivel no es útil, por lo que es necesario descomponer el proceso en sus funciones principales. Una vez identificadas las funciones principales e incorporadas al DFD, el analista de sistemas debe comenzar a descomponer cada una de estas funciones a niveles más detallados.

A medida que se va descomponiendo un DFD en niveles mayores de detalle se realiza implícitamente una descomposición funcional del sistema.

El proceso de modelado con DFD debe ir progresando de lo general a lo particular, detallando por separado cada proceso hasta el nivel necesario que muestre los alcances y limitaciones del sistema. Si un DFD explota hasta el último detalle de cada proceso, prácticamente se estará creando la estructura funcional detallada del sistema.

Cuando un Sistema es complejo se recomienda que el alcance del DFD sea del primer nivel o segundo nivel, ya que la complejidad de los diagramas particulares le restaran utilidad al modelo del DFD.

Para ejemplificar lo comentado anteriormente se obtendrá el diagrama de flujo de datos del proyecto (Figura 2.4).

Entidades Externas del proceso:

- Cliente
- Departamento de Estudios Socioeconómicos.

Tipo de Entradas que se generan hacia las entidades externas:

- Orden.

Tipo de Salidas que se generan hacia las entidades externas:

- Factura.
- Reporte y consultas (on-line).
- Estudio procesado.

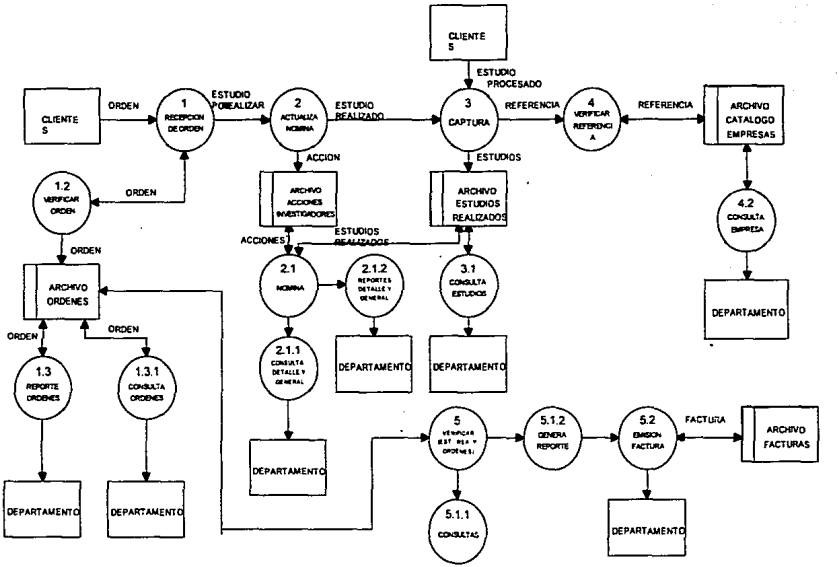


Figura 2.4 Diagrama de flujo de datos del proyecto

B) DIAGRAMA JERÁRQUICO DE LAS FUNCIONES DEL SISTEMA.

Cualquier función de negocios se puede descomponer en otras funciones, pero de niveles de detalle inferiores. Esta descomposición se realiza hasta lograr funciones atómicas esto es, que no pueden descomponerse más. Está descomposición también proporciona arreglos de funciones en grupos jerárquicos conocidos, así como funciones jerárquicas de negocios (Figura 2.5).

El diagrama jerárquico es una herramienta gráfica utilizada para definir que funciones ocurren en un negocio (o proceso), agrupando las funciones dentro de conjuntos lógicos para poderlas verificar. Además, facilita agregar o eliminar funciones en la estructura para reconfigurar la jerarquía en cualquiera de sus rutas.

C) DESCRIPCIÓN GENERAL DE LAS FUNCIONES DEL SISTEMA.

EL sistema se divide en los siguientes módulos: Recepción de Ordenes, Estudios, Catálogo de Empresas, Catálogo de Clientes, Catálogo de Investigadores y Nómina.

A continuación se explica en que consiste cada uno de los módulos que componen el sistema.

LA RECEPCIÓN DE ORDEN:

Este módulo permitirá llevar el control de las empresas solicitantes y de las personas a quién se les realizará el estudio. Este se subdivide en tres módulos cada uno de ellos tiene una función específica, el primero permite la captura de la orden, el segundo la consulta de las mismas y el tercero involucra la realización de un reporte de ordenes recibidas por día.

ESTUDIOS:

Permitirá ingresar nuevos estudios así como la consulta de ellos. Este módulo se subdivide en dos, el primero comprende las diferentes capturas de cada tipo de estudio como la impresión de cada uno ellos. Y el segundo comprende la consulta de los estudios realizados.

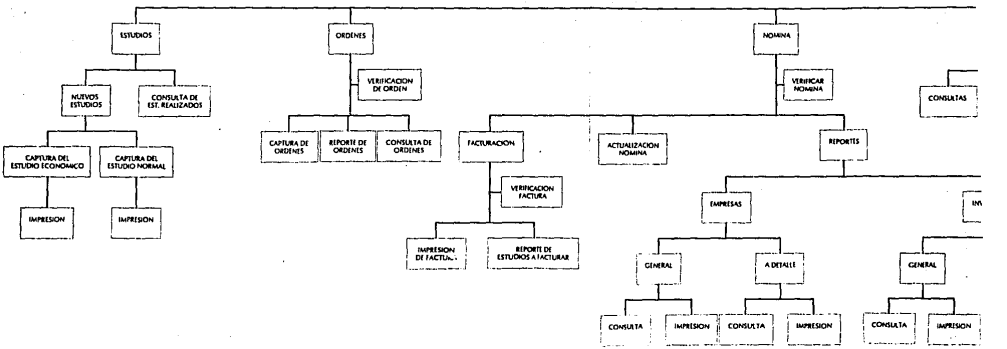


Figura 2.5 Diagrama jerarquico del siste

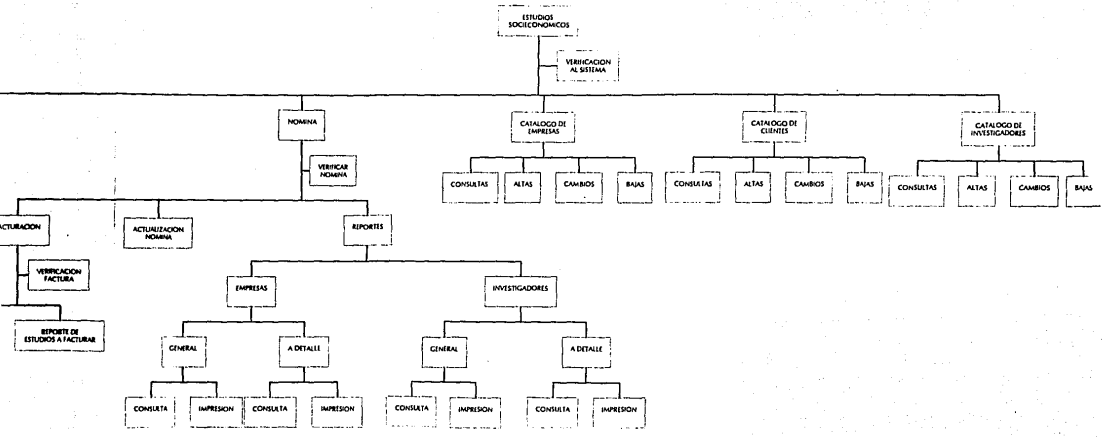


Figura 2.5 Diagrama jerarquico del sistema

CATÁLOGO DE EMPRESAS:

Con este módulo se tendrá un control de todos los datos necesarios para solicitar información y para elaboración de posteriores estudios. Este módulo se divide en cuatro, estos cubren las funciones de dar de alta, baja, modificación y consulta.

CATÁLOGO DE CLIENTES:

Con este catálogo se tendrá un control de los clientes que solicitan los estudios. Este módulo se divide en cuatro, cubriendo las funciones de dar de alta, baja, modificación y consulta.

CATÁLOGO DE INVESTIGADORES:

Se tendrá un control de los investigadores. Este módulo se divide en cuatro, y se cubren las funciones de dar de alta, baja, modificación y consulta.

NÓMINA:

Módulo Actualización:

Permitirá llevar el control del ingreso del investigador por estudio realizado.

Módulo Facturación:

Permitirá obtener un listado de los estudios entregados y realizar la factura correspondiente.
Se subdivide en dos módulos: el primero tiene la función de proporcionar la consulta por pantalla de los estudios entregados en un determinado período de tiempo y con la opción de poder imprimir un listado; el segundo consiste solamente en la impresión de la factura.

Módulo Reportes:

Permitirá la obtención de reportes. Este se subdivide en dos módulos:
El primero tiene la función de obtener un reporte general o a detalle de las empresas. El segundo permite obtener un reporte también de manera general o a detalle de los investigadores.

D) DEFINICIÓN DE PANTALLAS Y REPORTES

Se determinó que los requerimientos de pantallas y reportes son los que se muestran a continuación.

Estas pantallas se utilizaron principalmente como una referencia de ubicación de los campos así como de los menús, los reportes muestran la programación.

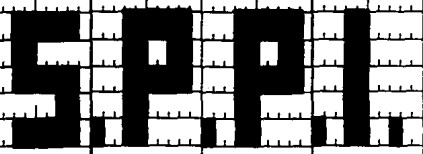
A continuación se muestran estos formatos de pantallas y reportes:

COLUMN

1 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80

ROW

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24



TECLEA LA CLAVE DE ENTRADA: XXXX

COLUMN

ROW	1 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80
01	ORDEN	ESTUDIOS	NOMINA	CAT EMPRESAS	CAT CLIENTES	CAT INVEST	SALIR	
02								
03								
04								
05								
06								
07								
08								
09								
10			MENU					
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								

COLUMN

ROW	1 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80
01								
02				RECEPCION DE ORDEN				
03								
04		No.		FECHA			ID	
05		ORDEN		PETICION			EMPRESA	
06	[	]		[	]		[	]
07								
08			TIPO			FECHA		
09			ESTUDIO			ENTREGA		
10			[ ]			[	]	
11								
12	DATOS INVESTIGADO:							
13								
14		NOMBRE				DIRECCION		
15	[				[			]
16		COLONIA				MUNICIPIO		
17	[			]		[		]
18		CIUDAD				ESTADO REPUBLICA		
19		[	]			[	]	
20								
21	TEL PART:	[	]			EXT OF:	[	]
22								
23			PUESTO:	[				
24								

2.5 DISEÑO TÉCNICO GENERAL

A) IDENTIFICACIÓN DE POSIBLES ALTERNATIVAS DE SOLUCIÓN

De acuerdo a las características y necesidades del departamento de estudios socioeconómicos de la empresa, se evaluaron las siguientes alternativas de solución:

1.- Se buscó en el mercado un paquete computacional que permitiera controlar las actividades del departamento de Estudios Socioeconómicos de la empresa, pero no fue posible adaptarlos a los requerimientos establecidos.

2.- La segunda alternativa fue cotizar el sistema fuera de la empresa y poder determinar si era posible adquirirlo. No fue posible, el costo excedía el presupuesto fijado por la empresa.

3.- Finalmente se determinó desarrollar el sistema dentro de la empresa.

Estas decisiones fueron tomadas con el fin de explotar los recursos disponibles en la empresa y en base al presupuesto con el que se dispone. Desde luego también influyó la decisión de los directivos de la empresa.

B) DEFINICIÓN DE REQUERIMIENTOS DE SEGURIDAD

-Se tendrán que implementar claves de entrada para el Sistema y la parte de Nómina, para restringir el acceso a personal no autorizado. El DBMS utilizado permite hacer uso de mecanismos de seguridad para el acceso restringido a los usuarios, permitiendo con ello el acceso solamente a cierta información de la base de datos.

C) IDENTIFICACIÓN POTENCIAL DE PROGRAMAS A DESARROLLAR

Se debe desarrollar un programa principal donde se verifique la entrada al sistema, al estar en el se podrá llamar a seis posibles subrutinas disponibles: Orden, Estudios, Catálogo de Empresas, Catálogo de Clientes, Catálogo de investigadores y Nómina.

SUBRUTINA DE ORDEN:

La orden de recepción llamará tres subrutinas, la primera será donde se captura la orden, la segunda será donde realicen las consultas de ordenes, y la tercera realizará la impresión del reporte de ordenes del día.

SUBRUTINA DE ESTUDIOS:

Los estudios llaman a su vez a dos subrutinas llamadas Nuevos estudios y Consultas.

La Subrutina de nuevos estudios llama a su vez a dos subrutinas donde se capturan los dos tipos de estudio, la tercera Subrutina será para impresión de cada uno de estos.

La Subrutina de Consulta solo hará desplegar la información requerida del archivo de estudios.

SUBRUTINA DE EMPRESAS:

El Catálogo de Empresas se divide en cuatro subrutinas que son alta, baja, cambio y consultas.

SUBRUTINA DE CLIENTES:

El Catálogo de Clientes se divide en cuatro subrutinas que son alta, baja, cambio y consultas.

SUBRUTINA DE CATÁLOGOS DE INVESTIGADORES:

El Catálogo de Investigadores llama a cuatro subrutinas que son alta, baja, cambio y consultas.

SUBRUTINA DE NÓMINA:

Antes de entrar a este módulo se tendrá un programa que verifique el acceso, este se compone de tres subrutinas que son Actualización de nómina, Reportes de Investigados - Empresas y Facturación.

Actualización Nómina

Se compone de cuatro subrutinas que son alta, baja, cambio y consultas.

Reportes:

Dentro de la subrutina de nómina tenemos 2 reportes de investigadores y 2 reportes de empresas.

Tanto para los reportes de investigadores como para los de empresas tenemos los reportes generales y los de detalle, estos pueden ser mandados a la pantalla o directamente a la impresora.

Facturación:

Se divide en dos llamadas que son: la consulta de los estudios entregados y la impresión de la factura.

D) ANÁLISIS DE COSTO-BENEFICIO

Un egreso implica gastar dinero para obtener todo lo que se necesita en un momento dado; una inversión permite obtener más dinero a futuro. Se dice que el desarrollo de un sistema es una inversión. Para todas las etapas de desarrollo se deben de determinar los fondos respectivos y al mismo tiempo estimar ciertos beneficios futuros, estos pueden ser en forma de reducción de costos de operación o nuevos ingresos. Debemos hacer hincapié, en lo siguiente, si los beneficios no superan los costos no es conveniente realizar el desarrollo de un nuevo sistema.

El análisis de costo/beneficio proporciona a la dirección de una empresa (Cliente) una visión de los costos, beneficios y riesgos que están asociados al desarrollo de un sistema, con el objetivo de hacer comparaciones y tener alternativas de inversión.

La comparación de costos y beneficios implican en primer lugar, la realización de una estimación del costo de desarrollo del sistema que se requiere construir.

Cada etapa del ciclo de vida del sistema tiene un costo. En el costo total de una fase, el analista debe considerar aspectos como el personal, equipo, materiales, gastos generales y otros como son los honorarios de una consultoría.

El costo de desarrollo solo tiene lugar una sola vez. Cuando finaliza el desarrollo del sistema, dan inicio los gastos operativos, que se presentaran durante toda la vida del sistema. Se puede decir entonces que el costo de desarrollo es una inversión, mientras que los costos de operación son gastos.

Cuando se invierte dinero, siempre se espera obtener algo a cambio, esto es un beneficio. Los beneficios pueden ser cualitativos o cuantitativos.

Estimación de Costos: En la mayoría de las organizaciones, la estimación de los costos se basa en la información obtenida de datos históricos, que permite identificar los factores de costo y determinar su importancia dentro de la organización.

Entre las técnicas de estimación de costos se encuentran:

- Técnicas de descomposición
    - Costo por esfuerzo de tareas.
    - Costo por líneas de código.
- Técnica del modelo por algoritmo o módulo.
    - COCOMO (Modelo Constructivo de Costo)

A continuación se desarrollará el método de Estimación de Esfuerzo en el que se aplica un número de personas-día, mes o año para solucionar cada etapa del proyecto; asociando un costo a cada unidad de esfuerzo se determina el costo estimado. Una vez determinadas las funciones del sistema y las fases del desarrollo del mismo deberán representarse como parte de una tabla. Más adelante se construirá la tabla para la estimación del esfuerzo.

Se deberá estimar el esfuerzo (persona-mes) para este caso, asociado a cada etapa de desarrollo de cada una de las funciones principales del sistema.

Estos datos son fundamentales en la tabla que se deberá construir. Posteriormente se aplicarán los coeficientes de trabajo (costo/ unidad de trabajo) a cada una de las fases de desarrollo. Este coeficiente de trabajo es probable que varíe para cada fase.

Por último se calculan los costos y el esfuerzo para cada etapa o fase de desarrollo del sistema.

Estimación del costo por esfuerzo de tarea.

Fases	B	C	D	G
Tareas A	***Análisis***	***Diseño***	***Implantación***	***Totales***
Estudios	5	5	5	5
Ordenes	5	5	5	5
Nómina	5	5	5	5
Catálogo Empresas	5	5	5	5
Catálogo Clientes	5	5	5	5
Catálogo Investigadores	5	5	5	5
I Total	5	5	5	5
II Coeficiente N$	3,500	3,500	3,500	
III Costo Total N$	17,500	17,500	17,500	**52, 500**

En la columna A se representan todas las funciones a realizar durante el proyecto. La columna B, C, y D contienen el número de personas requeridas para cada tarea o fase. La columna G es el número total de personas para realizar la función. Se obtiene sumando horizontalmente las columnas. El renglón I se obtiene sumando cada columna y nos dice cual es el número total de personas para cada Tarea. El renglón II es el costo para cada etapa o tarea, y por último el renglón III se obtiene de III = I x II obteniéndose con esto el costo total por tarea.

El costo total de las tareas es la suma de todo el renglón III y es el costo del sistema.
El esfuerzo estimado para todas las tareas es la suma de todo el renglón I.

E) DEFINICIÓN DE LA CONFIGURACIÓN DE HARDWARE Y SOFTWARE

Para diseñar este sistema se contó con dos opciones: ORACLE e INFORMIX-4GL, de los cuales se seleccionó el más adecuado.

ORACLE

Resultó ser una buena opción para realizar el diseño de la base de datos con este tipo de manejador. Pero debido al costo y la necesidad de ampliar la memoria del equipo de la empresa (ORACLE requiere de 8 MB de RAM), se descarto ésta posibilidad.

INFORMIX-4GL

Se determinó usar este software, ya que fue el que más se adapto a las características del equipo de cómputo de la empresa (se dispone de 4 MB de RAM por terminal) además, se tiene la facilidad de contar con él y desde luego el costo de adquisición es más accesible.

SOFTWARE:

El Manejador de Base de Datos a utilizar INFORMIX-4GL, proporciona todas las herramientas necesarias para la creación de sofisticadas aplicaciones de bases de datos, un ambiente óptimo para desarrollo de aplicaciones, además de proporcionar un alto performance.

INFORMIX-4GL combina declaraciones procedurales y no-procedurales lo que permite mayor flexibilidad para la creación de programas. Con el uso de instrucciones como: OPEN, WINDOW, MENÚ, DISPLAY, INPUT, FORM, etc. permite optimizar el número de líneas de código. Esto es, menos código destinado a menos tiempo de compilación. Las características de INFORMIX-4GL se pueden resumir en lo siguiente:

Manejo de Ventanas.
Creación de menús.
Lectura de datos de formas de pantalla.
Uso de SQL para manipular la base de datos.
Manejo de pantalla de ayuda.
Creación de reportes.
Búsqueda de información de condiciones especificas por el usuario.
Detección de teclas de función y de control que oprime el usuario.
Definición de atributos para la información que se lee y escribe a la pantalla.

HARDWARE:

Como equipo mínimo necesario se requiere una computadora con un procesador 80386 o superior, un disco duro de 80 Mb mínimo recomendable y además que tenga 4 Mb en RAM.

2.6 GENERACIÓN DEL DOCUMENTO DE ANÁLISIS.

El diseño e implementación del sistema de información usando un lenguaje de cuarta generación permitirá automatizar estudios socioeconómicos y ayudar a la toma de decisiones de forma rápida y oportuna.

Las funciones principales que realizará el sistema a diseñar son:

ORDENES.	Esta función permitirá llevar el control de las empresas solicitantes, así como de las personas a las que se les realizará el estudio (personas investigadas).
ESTUDIOS.	Esta función permitirá ingresar nuevos estudios y la consulta de los mismos. ingresar nuevos estudios y la consulta de los mismos.
CATÁLOGO DE EMPRESAS.	Este catálogo contendrá toda la información necesaria para la solicitud de información, así como la elaboración de estudios posteriores.
CATÁLOGO DE CLIENTES.	Este catálogo permitirá controlar a todos los clientes que solicitan los estudios.
CATÁLOGO DE INVESTIGADORES.	Este catálogo controlará a todos los investigadores contratados.
CONTABLE.	Esta función permitirá llevar el control de los ingresos de investigadores en relación a sus estudios realizados, así como realizar la factura correspondiente. Además de permitir realizar reportes generales a detalle de las empresas y de los investigadores.

La comunicación del usuario con el sistema, se llevará a cabo a través de menús desplegados en pantalla.

Para tener acceso al sistema se tendrá que tener una clave de acceso, al igual para el módulo ***contable.***

El diseño de estas funciones se realizará con el DBMS INFORMIX-4GL, el cual se encuentra disponible en la empresa.

Se requiere además del siguiente equipo de computo como mínimo: una computadora PC con microprocesador 80386/33 Mhz o superior, disco duro de 80 Mb mínimo y 4 Mb en memoria RAM.

CAPÍTULO 3.- DISEÑO DETALLADO DEL PROYECTO

Una vez establecidos los requerimientos del sistema de software, el diseño es la siguiente etapa en el desarrollo del sistema.

Un buen diseño es la clave de ingeniería software. Un sistema de software bien diseñado es fácil de aplicar y mantener, además de ser comprensible y confiable. Los sistemas mal diseñados, aunque puedan funcionar, pueden ser caros de mantener , difíciles de probar y poco confiables. La etapa de diseño es, por tanto, la parte más importante del proceso de desarrollo de software.

Un diseño mantenible implica minimizar el costo de los cambios del sistema, y eso significa que el diseño tiene que ser comprensible y que las modificaciones deben tener un efecto local. Ambas cosas se logran si el diseño del software es muy coherente y poco acoplado. La ventaja de los sistemas con mucha coherencia y poco acoplamiento es que cualquier unidad de programa se puede remplazar por una unidad equivalente con pocos o ningún cambio en las otras unidades del sistema. Esto es importante a la hora de refinar los diseños. Tener unidades poco acopladas significa que el diseñador posee la opción de cambiar de opinión sobre el diseño de la unidad sin que haya efectos negativos en el resto del sistema.

El diseño efectivo del software se logró mejorar utilizando la metodología funcional descendente, en la cual se considera al sistema desde un punto de vista funcional. Iniciando con una visión de alto nivel y refinándola de manera progresiva hasta llegar a un diseño mas detallado.

En este capítulo se realizará el diseño detallado del proyecto que comprende la descripción detallada de las funciones, la importancia de la seguridad y protección del sistema, el modelado de datos, diseño de la base de datos y la especificación de programas.

3.1 DESCRIPCIÓN DETALLADA DE FUNCIONES

En este punto se lleva a cabo la descripción detallada de los requerimientos del sistema. La especificación de los detalles algorítmicos, el diseño de pantallas y reportes definitivos, son las actividades realizadas.

A) DIAGRAMAS DE FLUJO.

Para mayor información de estos diagramas, referirse al punto 2.5 de Diseño Técnico General.

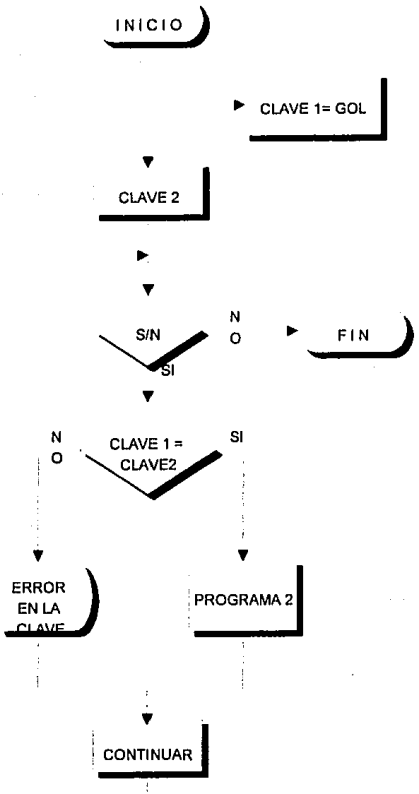


Figura 3.1 Diagrama de flujo del programa funcional

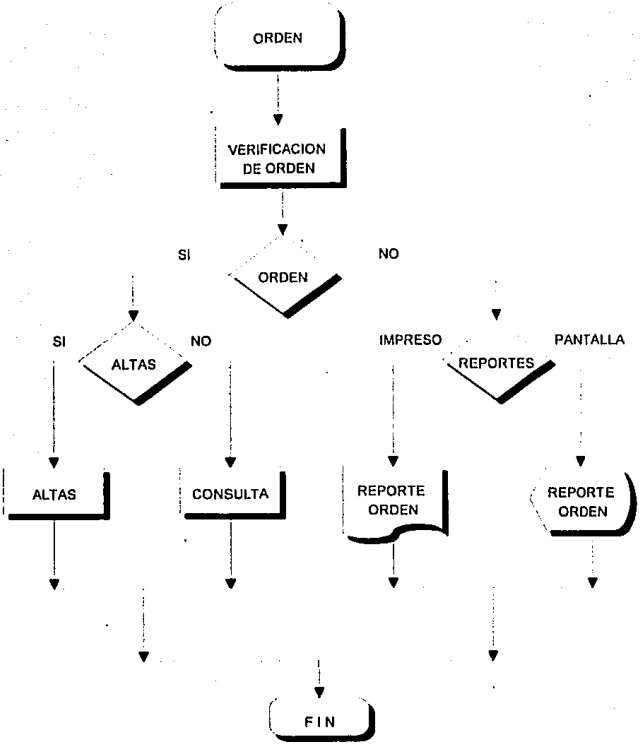


Figura 3.2 Ordenes de estudio

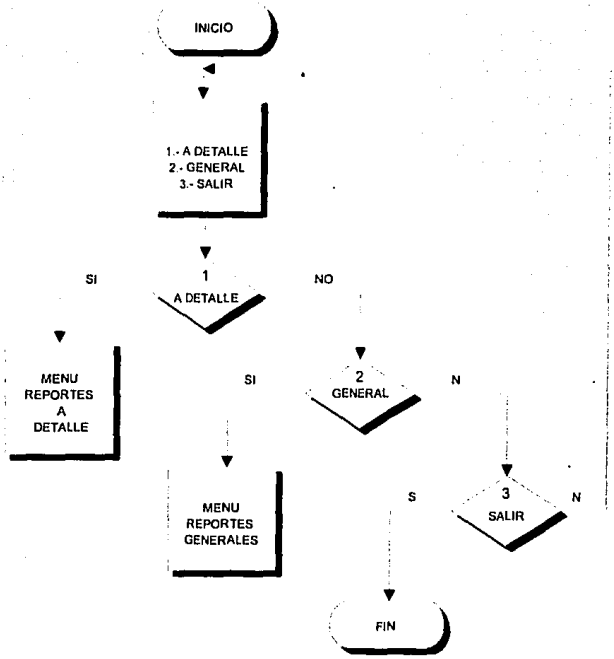


Figura 3.3 Menú de reportes Nómina

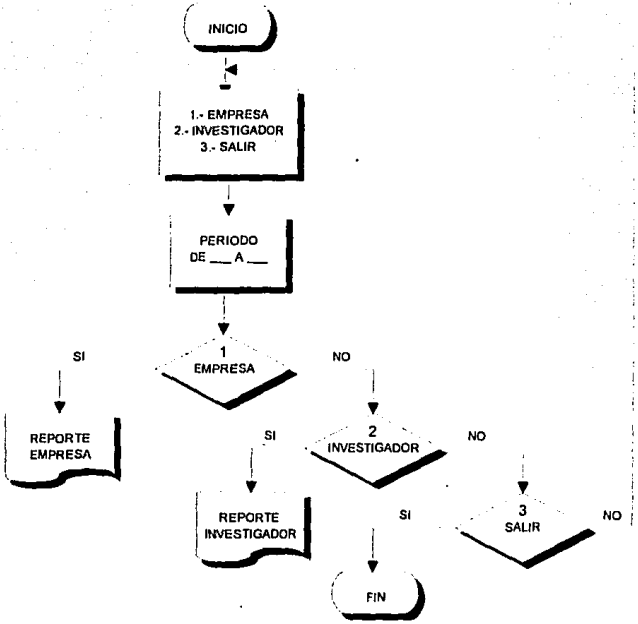


Figura 3.4 Menú de Reportes Generales

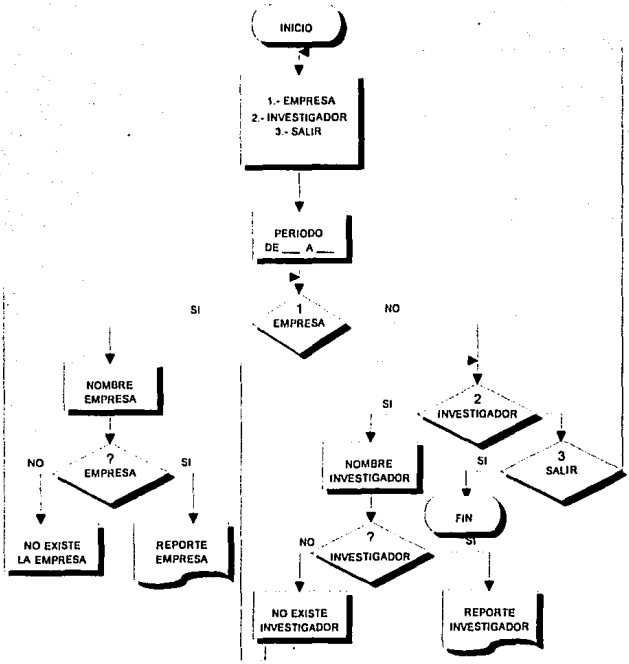


Figura 3.5 Menú de Reportes a Detalle

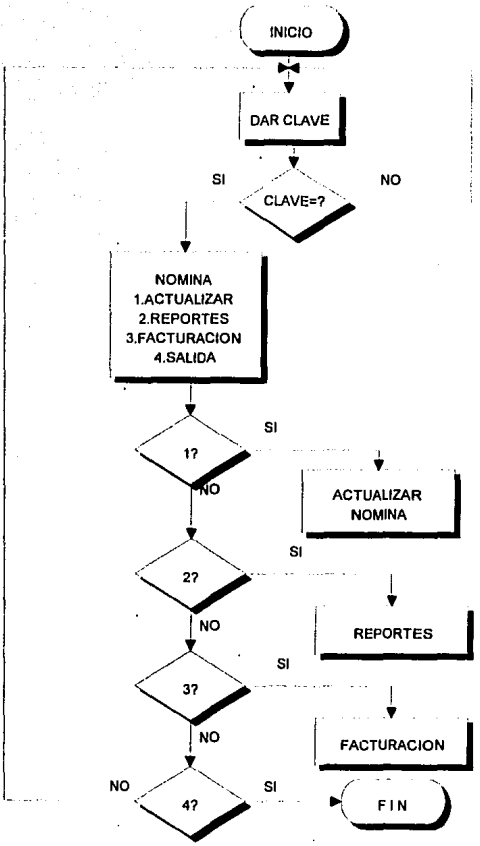


Figura 3.6 Menú de Nómina

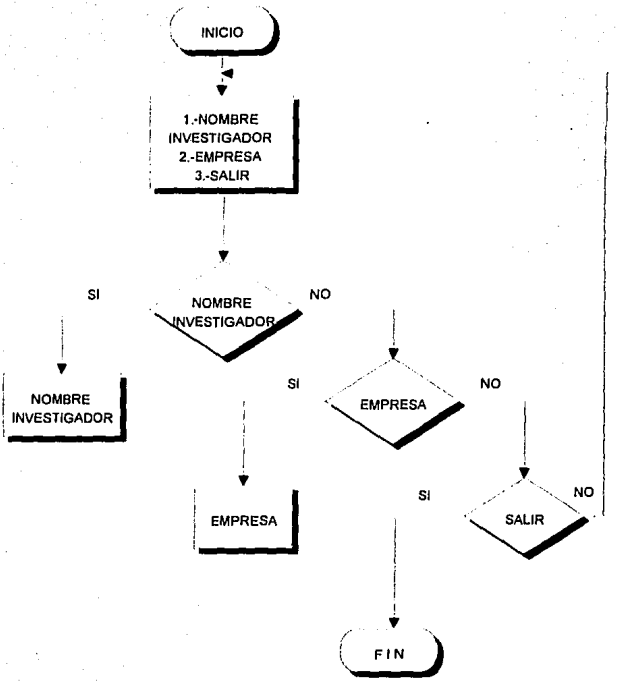


Figura 3.7 Diagrama de Flujo Principal de Consultas de Estudios

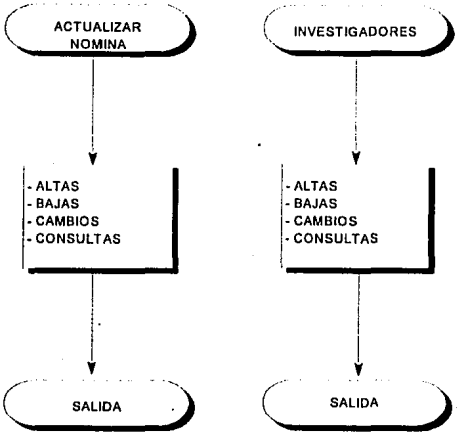


Figura 3.8 Menú de Nómina y de Investigadores

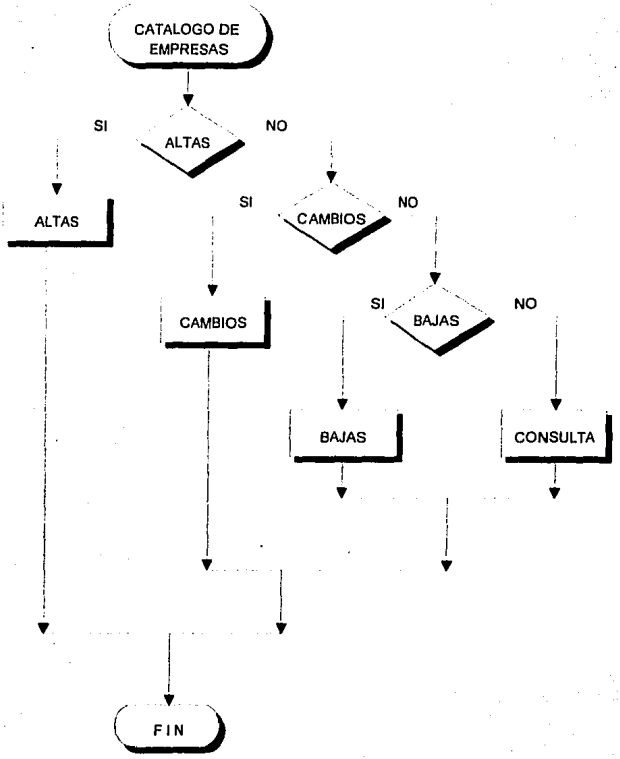


Figura 3.9 Catálogo de Empresas

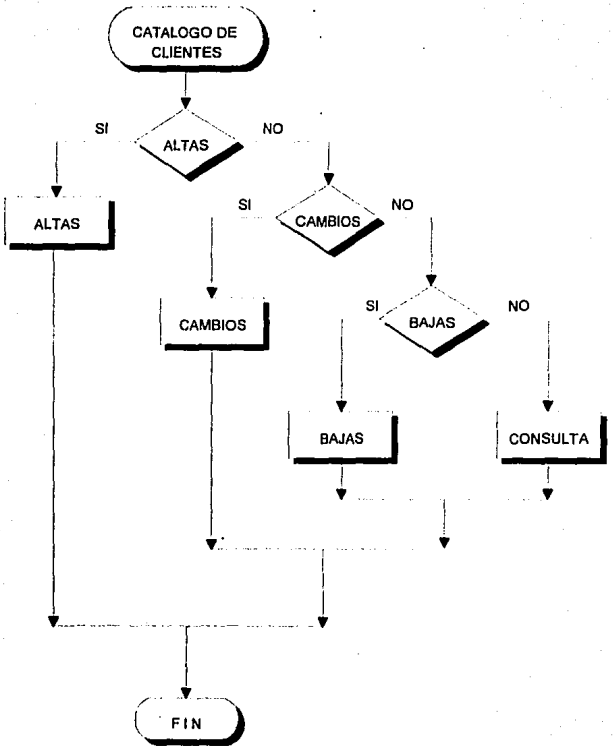


Figura 3.10 Catálogo de Clientes

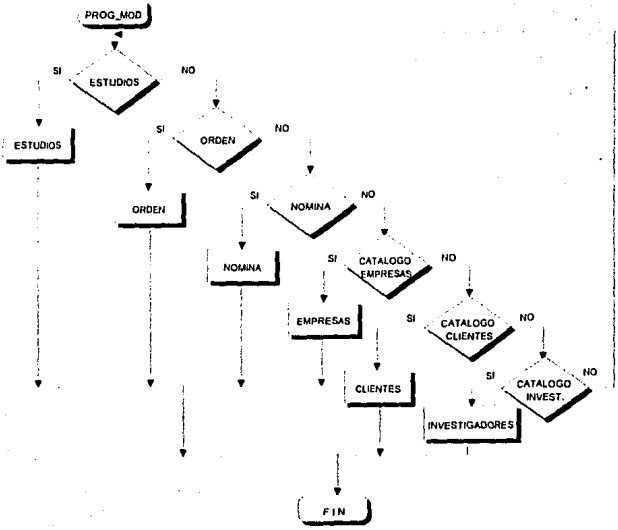


Figura 3.11 Diagrama de Flujo Principal

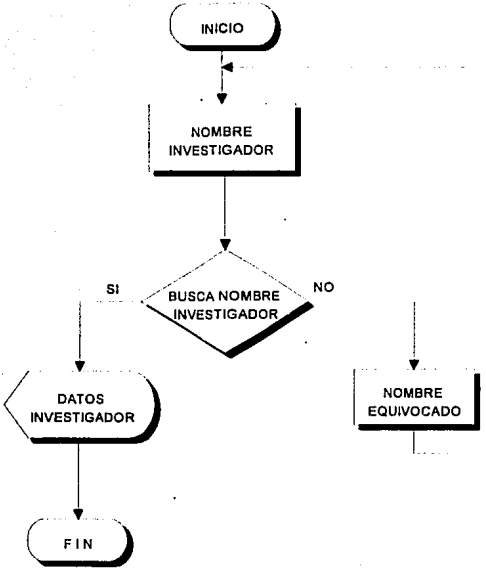


Figura 3.12 Diagrama de Flujo de Consultas por Investigador

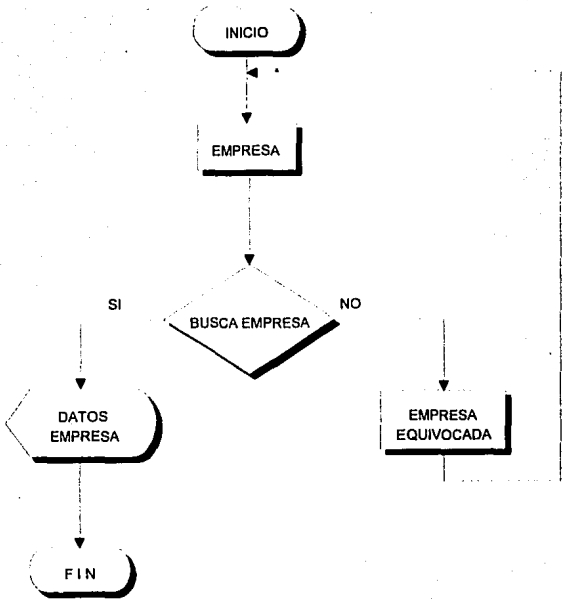


Figura 3.13 Diagrama de Flujo de Consultas por Empresa

B) FORMATO DEFINITIVO DE PANTALLAS.

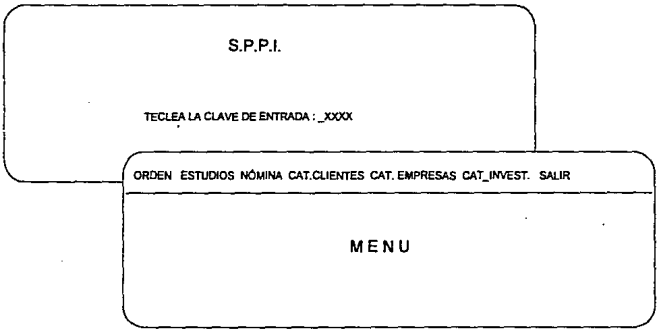


Figura 3.14 Pantallas del menú principal

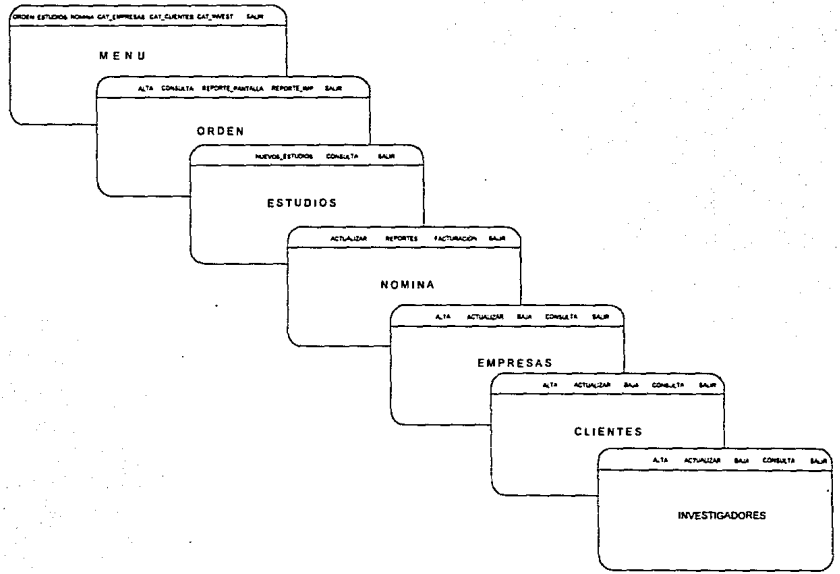


Figura 3.15 Pantallas del menú principal

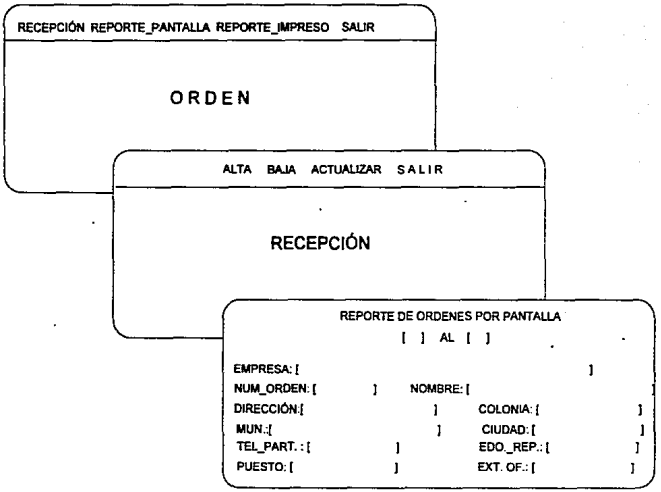


Figura 3.16 Pantallas del menú de orden.

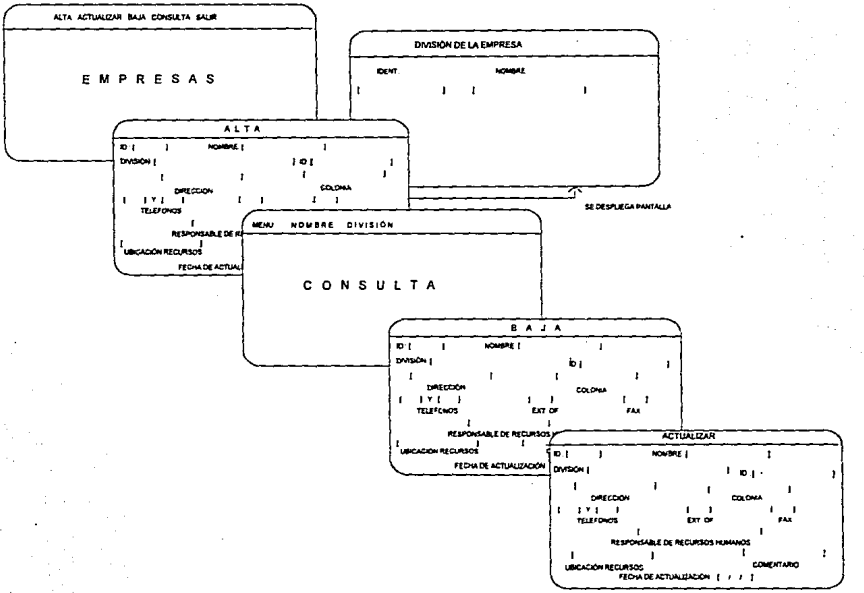


Figura 3.17 Pantallas de consulta de empresas

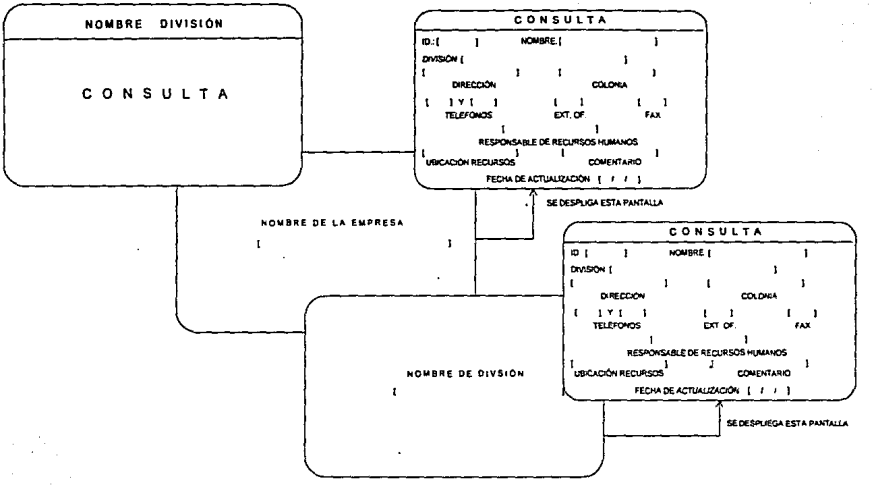


Figura 3.18 Pantallas de consulta de empresas

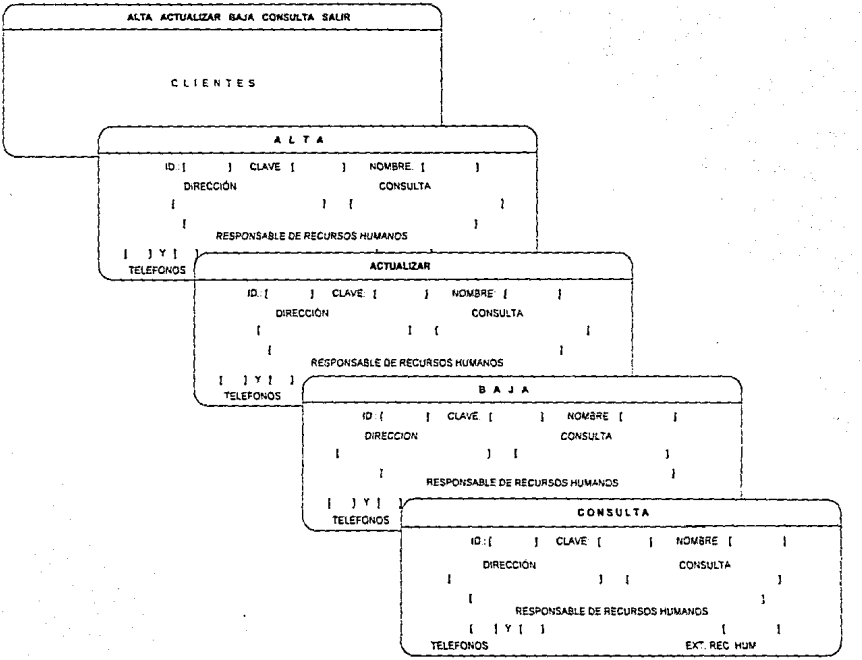


Figura 3.19 Menu de pantallas de cliente

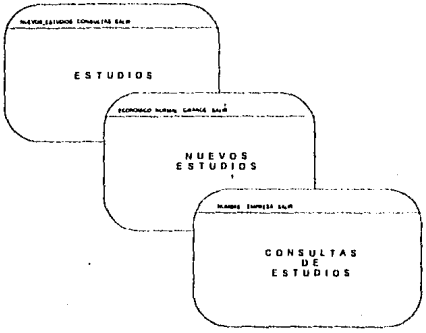


Figura 3.20 Pantallas del menu de estudios.

C) REPORTE DE IMPRESIÓN.

FECHA : __/__/__

REPORTE DE ORDEN EMPRESA

EMPRESA: |

Nombre: |
Calle: | Colonia: |
Municipio: | Ciudad: |
Tel. Part.: | Ext. Of.: |
Puesto: |

Nombre: |
Calle: | Colonia: |
Municipio: | Ciudad: |
Tel. Part.: | Ext. Of.: |
Puesto: |

Nombre: |
Calle: | Colonia: |
Municipio: | Ciudad: |
Tel. Part.: | Ext. Of.: |
Puesto: |

Nombre: |
Calle: | Colonia: |
Municipio: | Ciudad: |
Tel. Part.: | Ext. Of.: |
Puesto: |

Nombre: |
Calle: | Colonia: |
Municipio: | Ciudad: |
Tel. Part.: | Ext. Of.: |
Puesto: |

REPORTE IMPRESO DE ORDENES POR EMPRESA

3.2 SEGURIDAD Y MEDIDAS DE PROTECCIÓN.

La seguridad tiene el significado de proporcionar acceso seguro a los datos, comenzando con la protección al acceso del sistema.

La protección de los datos deberá estar orientada a negar el acceso a personas que no tengan derecho a ellos, garantizando con ello la privacía de datos personales, y al mismo tiempo que cada persona aplique adecuadamente su privilegio de acceso. Pero esto también quiere decir que los propietarios de bases de datos también tienen la responsabilidad de proteger los datos que se les han confiado. Esta responsabilidad involucra la operación confiable del equipo de hardware, además de que los datos estén protegidos contra todo riesgo.

A continuación se listan algunos ejemplos de estos riesgos, así como algunas sentencias en INFORMIX para su protección y respaldo:

A) LISTA DE PELIGROS POTENCIALES

- Acceso a información confidencial.
- Captura de tipos de datos incorrectos.
- Descompostura del equipo.
- Alteración de los programas.
- Fallas eléctricas.
- Alteración accidental de la estructura de la base de datos.

B) MEDIDAS DE PROTECCIÓN DE LOS DATOS

Cuando una base de datos es creada, solamente el administrador de la base de datos (DBA) tiene todos los permisos para accesar la base de datos. Otros usuarios pueden tener acceso a la base de datos de acuerdo a los permisos otorgados por el DBA y solamente el DBA tiene estos privilegios aunque otros usuarios podrían tener otorgado el permiso de trabajar en su totalidad con la base de datos de acuerdo a los criterios de la empresa y del DBA.

Para que el DBA pueda lograr este control tiene que otorgar ciertos atributos permisos y esto se puede hacer en INFORMIX con la función Grant. Si se le quiere otorgar o permiso a todos los usuarios de manejar la base de datos sería:

Grant all to public

También las restrieciones pueden hacerse para un cierto grupo de usuarios como se ve en la siguiente línea:

Grant all to Jose, Mary

Los permisos para accesar la base de datos pueden ser restringidos por ciertas relaciones, por ejemplo otorgarle el permiso para adherir, borrar o modificar los datos:

Grant alter on cd to Sam

Los usuarios también pueden tener permiso de cambiar valores de ciertos campos:

Grant update (fnombre, lnombre, compañia, ciudad) on customer to Eduardo

Estos permisos pueden ser eliminados con la instrucción Revoke, ejemplo:

Revoke all on orders from public

Este tipo de instrucciones son importantes para tener la seguridad de que nuestra información no pueda ser alterada por ningún usuario que no sea el indicado para el manejo de la información permitiéndonos evitar cualquier tipo de corrupciones en nuestra base de datos.

C) PROCEDIMIENTOS PARA PROCESOS DE RESPALDOS

El proceso de respaldo es una transacción que consta de una secuencia de una o más sentencias SQL que juntas forman una unidad de trabajo. Cada sentencia de una transacción efectúa una parte de una tarea, pero todas ellas son necesarias para completar la tarea. La agrupación de las sentencias en una sola transacción indica al DBMS que la secuencia de sentencias entera debe ser ejecutada atómicamente; todas las sentencias deben completarse para que la base de datos esté en un solo estado consistente.

He aquí algunos ejemplos de transacciones típicas para la base de datos.

- Añadir un dato. Para aceptar el pedido de un cliente, el programa de entrada de pedido debería: a) consultar la tabla para asegurar que el dato está en stock; b) insertar el dato en dicha tabla; c) actualizar la tabla de acuerdo a los datos existentes; y d) actualización de cada una de las tablas que estén relacionadas con dicha información o dato.

- Cancelar un dato. Para cancelar un dato, el programa debería: a) eliminar el dato de la tabla en uso; b) actualizar la tabla en uso de acuerdo a los datos existentes; y c) actualizar las tablas en cuyo caso se solicite el dato o información en uso.

En cada uno de estos casos se requiere una secuencia de cuatro o cinco acciones, cada una formada por una sola sentencia SQL, para manejar la única transacción *lógica*.

El concepto de transacción es crítico para los programas que actualizan una base de datos ya que asegura la integridad de la base de datos. Un DBMS como INFORMIX basado en SQL efectúa este compromiso referente a las sentencias de una transacción:

Las sentencias de una transacción se ejecutarán como una unidad atómica de trabajo en la base de datos. O todas las sentencias son ejecutadas con éxito, o ninguna de las sentencias es ejecutada.

El DBMS es responsable de mantener este compromiso incluso si el programa de aplicación aborta o se produce un fallo hardware a mitad de la transacción, tal como se muestra en la Figura 3.21 en cada caso, el DBMS debe asegurarse que cuando se complete una recuperación del fallo, la base de datos nunca refleje una <<transacción parcial>>.

En un lenguaje basado en SQL como INFORMIX-4GL existen dos sentencias de procesamiento de transacciones o respaldo de información como lo son COMMIT y ROLLBACK.

- La sentencia COMMIT señala el final correcto de una transacción. Informa al DBMS que la transacción está completa; todas las sentencias que forman la transacción han sido ejecutadas, y la base de datos es autoconsistente.

La sentencia ROLLBACK señala el final sin éxito de una transacción. Informa al DBMS que el usuario no desea completar la transacción. En vez de ello, el DBMS debe deshacer los cambios efectuados en la base de datos durante la transacción. En efecto, el DBMS restaurara la base de datos a su estado antes de que la transacción comenzara.

1. Una sentencia COMMIT finaliza la transacción con éxito, haciendo que los cambios a la base de datos sean permanentes. Una nueva transacción comienza inmediatamente después de la sentencia COMMIT.
2. Una sentencia ROLLBACK aborta la transacción, deshaciendo las modificaciones que haya efectuado a la base de datos. Una nueva transacción comienza inmediatamente después de la sentencia ROLLBACK.
3. La terminación de un programa con éxito (para SQL programado) también finaliza la transacción correctamente, igual que si se hubiera efectuado una sentencia COMMIT. Puesto que el programa está finalizado, no hay ninguna nueva transacción que comenzar.

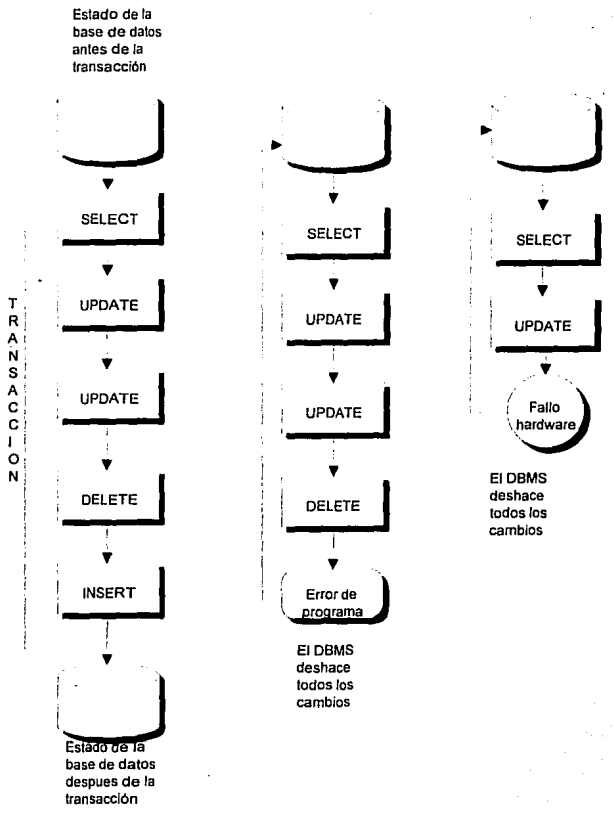


Figura 3.21 Concepto de Transacción

4. La terminación anormal del programa (para SQL programado) también aborta la transacción, del mismo modo que si hubiera ejecutado una sentencia ROLLBACK puesto que el programa está finalizado, no hay ninguna nueva transacción que comenzar.

La figura 3.22 muestra algunas transacciones típicas que ilustran estas cuatro condiciones. Observe que el usuario o programa está siempre en una transacción. No se requiere ninguna acción explícita para comenzar una transacción; comienza automáticamente con la primera sentencia SQL o inmediatamente después que la transacción procedente acaba.

En conclusión INFORMIX-4GL nos permite realizar transacciones o respaldos con las sentencias COMMIT y ROLLBACK los cuales son clave para recuperar y dar mantenimiento a una base de datos después de un fallo del sistema; sólo las transacciones que estaban complementadas en el momento del fallo permanecen en la base de datos recuperada.

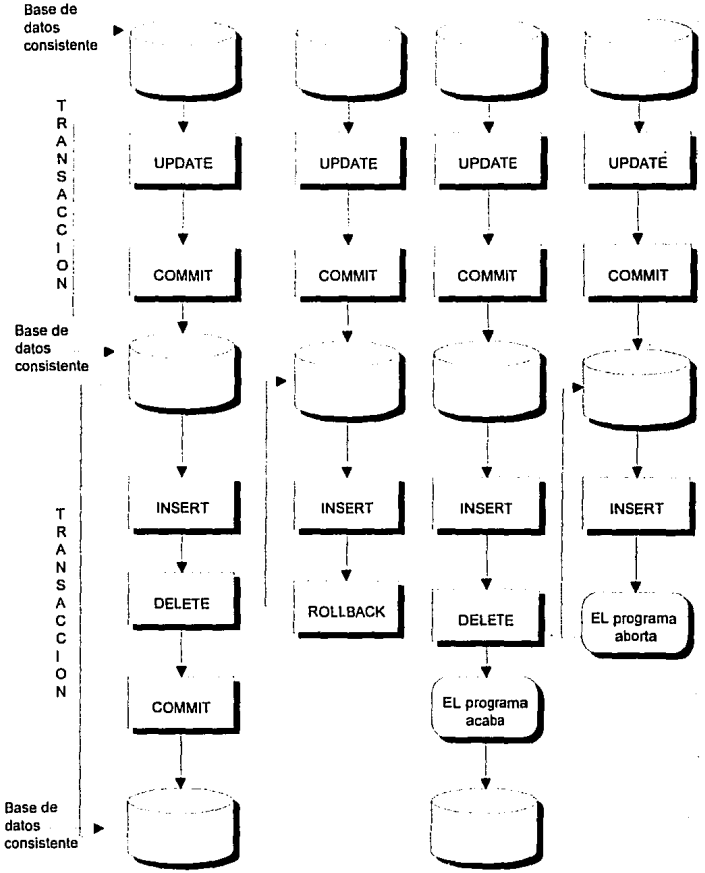


Figura 3.22 Transacciones complementadas COMMIT y vueltas atrás ROLLBACK

3.3 MODELADO DE DATOS

Actualmente una de las actividades más importantes en la mayoría de las organizaciones es la administración de los datos. En la medida en que nos movemos hacia una sociedad más orientada a la información, el problema es como determinar la organización de los datos para maximizar su utilidad. Sin embargo, mucha gente dedicada al procesamiento de datos no ha entendido completamente el problema de como organizar los datos para utilizar a su máxima capacidad los sistemas de base de datos.

MODELOS DE DATOS: Para definir la estructura de una base de datos es necesario definir el concepto de modelo de datos. Un modelo de datos es una representación abstracta (una descripción) de los datos por medio de sus entidades, eventos, actividades y sus asociaciones dentro de una organización. Modelos lógicos basados en objetos se utilizan para describir los datos en los niveles conceptuales y de vista. Son flexibles y permiten especificar claramente las limitantes de los datos. Uno ejemplo es: el modelo entidad - relación. Este modelo es el representativo de la clase de los modelos lógicos basados en objetos. Ha tenido bastante aceptación como modelo de datos apropiado para el diseño de base de datos y se utiliza ampliamente en la práctica. Se utiliza para describir los datos en los niveles conceptuales y de vista. Sirve para especificar la estructura lógica general de la base de datos.

METODOLOGÍA ENTIDAD-RELACIÓN: Es una herramienta para que los analistas y diseñadores de sistemas puedan desarrollar un modelo lógico de la base de datos de su organización. Los enfoques convencionales para el diseño lógico de base de datos normalmente solo tiene una fase: convertir la información de los objetos del mundo real directamente al esquema del usuario. La metodología ER para el diseño lógico de base de datos consiste de dos fases principales:

1) Definir el esquema empresarial usando diagramas ER.
2) Traducir el esquema empresarial al esquema del usuario.

Las ventajas son:

1) La división de las funciones y el trabajo en dos fases hace que el diseño de la base de datos sea más simple y más organizada.

2) El esquema empresarial es fácil de diseñar ya que no se encuentra restringida por las capacidades del DBMS y es independiente del almacenamiento y de consideraciones de eficiencia.

3) El esquema empresarial es más estable que el esquema del usuario. Si se desea cambiar de un DBMS a otro, el esquema empresarial será el mismo, ya que este es independiente del DBMS que se use.

4) El esquema empresarial expresado por un diagrama entidad-relación es más fácil de entender por gente sin conocimientos de computación.

El modelo de datos entidad -relación se basa en una percepción de un mundo real que consiste en un conjunto de objetos básicos llamados entidad y relaciones. El diseñador de la base de datos es el responsable de seleccionar las relaciones relevantes. también debe especificar el tipo de asociación de las relaciones (uno a uno, uno a muchos, muchos a muchos).

La conectividad de una relación específica es el tipo de asociación de las ocurrencias de las entidades de la relación. Los valores de la conectividad son "uno" o "muchos". El número real asociado con el término muchos es llamado la cardinalidad de la conectividad. Los tipos básicos de conectividad son los siguientes: uno a uno, uno a muchos y muchos a muchos.

Propiedades de las tablas

1.- Cada columna contiene valores relativos al mismo atributo, y cada valor de una columna de la tabla debe ser siempre un solo valor.

2.- Cada columna tiene un nombre distinto (nombre del atributo), y el orden de las columnas no es importante.

3.- Cada renglón es distinto; esto es, un renglón no puede duplicarse en otro para un grupo de columnas seleccionadas como llave.

4.- La secuencia de los renglones no es importante.

5.- Todos los valores no llave deben ser totalmente dependientes de toda la llave.

6.- Cada atributo no llave debe depender solo de la llave de la relación, no de ninguna otra no llave.

PASOS PARA DESARROLLAR UN DIAGRAMA ENTIDAD-RELACIÓN

1) Determinar las entidades junto con sus identificadores asociados. Todas las entidades deben tener al menos un identificador que identifique de manera única a cada entidad.

2) Determinar las relaciones entre las entidades. Cada relación debe tener el identificador de cada una de las entidades asociadas, y las relaciones deben tener al menos dos entidades asociadas, excepto las relaciones recursivas que solo tienen una.

3) Determinar la cardinalidad de la asociación entre cada entidad y relación, la información de la cola inferior de la cardinalidad es opcional.

CONVERSIÓN DEL MODELO ENTIDAD-RELACIÓN AL MODELO RELACIONAL

El modelo ER es una mejora semánticamente más rica del modelo relacional fundamentada en conceptos relacionales. Es por esta riqueza semántica, que el modelo ER se ha adoptado en virtualmente todas las herramientas CASE para el diseño de base de datos. Otro hecho importante es que la ANSI eligió al modelo de datos Entidad-Relación como el estándar para los Sistemas Diccionarios de Recursos de información. El modelo relacional únicamente soporta la definición de un tipo de objeto, la tabla o relación; el modelo ER soporta la definición de dos tipos de objetos, entidades y relaciones.

De una manera muy simple se puede decir que la entidad del modelo Entidad- Relación corresponde a las tablas del modelo relacional, y que las relaciones del modelo ER, si tiene campos, también corresponden a tablas del modelo relacional.

PASOS DE CONVERSIÓN:

EN UNA RELACIÓN DE "uno a muchos" el identificador de la entidad correspondiente a la cardinalidad " uno" pasa a ser llave extranjera de la tabla correspondiente a la entidad con cardinalidad "muchos".

EN UNA RELACIÓN DE "muchos a muchos" es necesario incluir una tabla intermedia que corresponda a la relación. Esta tabla contendrá los identificadores de las dos entidades y los campos propios de la relación.

EN UNA RELACIÓN DE "muchos a uno" cuando se llega a tener una relación de grado 3,4,.., la relación se identifica con los identificadores de cada una de las entidades asociadas. Es por esto que cada entidad corresponde a una tabla, lo mismo que la relación, junto con sus campos si tuviera.

Utilizando el Upper CASE (ORACLE) se obtuvo la documentación gráfica necesaria. Esta documentación generada permitirá establecer un prototipo que sirva de base a los analistas para el diseño de la programación.

3.4 Diseño de la Base de Datos

Existe una serie de procesos requeridos para poder llegar a la etapa de generación de una aplicación de sistemas.

a) Se requiere comprender el modelo de negocios.

b) Se requiere tomar decisiones sobre los recursos de implementación .

c) El modelo conceptual de datos debe ser trasladado físicamente a una Base de Datos funcional.

Asumiendo que el resultado final es un sistema y que el punto de partida son algunos aspectos identificables del mundo real, es necesario definir los procesos a través de los cuales se va a llevar de un punto a otro. Estas son las operaciones que CASE soporta y automatiza.

Como ya se mencionó en párrafos anteriores se requiere de habilidad de interpretación de las necesidades de los usuarios para llevar estos requerimientos a un modelo. CASE hace estas tareas más productivas gracias al Case Designer y al Case Directory.

EL MODELO DEL NEGOCIO (DATOS Y FUNCIONES)

Los resultados de este análisis pueden dividirse en un modelo de datos y en un modelo de funciones, los cuales deben ser firmemente asociados a cierto nivel a fin de ser representaciones de un mismo esquema .

DISEÑO DE LA BASE DE DATOS

Los diseños para objetos de la base de datos y programas de módulos serán codificados por los Generadores de CASE, estos comandos generan la definición de datos para la Base de Datos y código de lenguaje de cuarta generación para producir componentes operacionales del sistema.

CASE lleva a cabo estos procesos mediante sus diferentes niveles: *UPPER CASE* , *MIDDLE CASE Y LOWER CASE.* Información más exacta de este tema se puede consultar en el manual de CASE.

Con la utilización del CASE de ORACLE se obtuvo el diseño de la Base de datos a partir de lo siguiente:

Con la interfaz gráfica de CASE DESIGNER se obtuvo el diagrama Entidad-Relación. Con CASE GENERATOR y tomando las definiciones del diccionario de datos se generó la base de datos. Una descripción detallada de ésta se dará más adelante.

NORMALIZACIÓN DE DATOS

La normalización es una técnica para analizar paso a paso las asociaciones entre los datos. Aquí se expone como una manera de verificar que de la conversión del modelo ER al modelo relacional se obtiene un esquema en tercera forma normal.

PASOS DE LA NORMALIZACIÓN: Una vez que se tiene el esquema relacional, y se revisa que no haya ninguna relación no normalizada, esto es, con grupos repetitivos. Se eliminan todos los grupos repetitivos de esta relación, obteniendo un conjunto de relaciones en primera forma normal (1NF). Se eliminan la dependencias funcionales parciales para obtener relaciones en segunda forma normal (2NF). finalmente, se eliminan las dependencias transitivas, creando una tercera forma normal (3NF).

Primera forma: Es una relación que contiene solo valores elementales (o simples) en la intersección de cada renglón y columna. Así, una relación normalizada no tiene grupos repetidos. Para normalizar una relación que contiene un solo grupo repetitivo, se elimina el grupo repetitivo y se forman dos nuevas relaciones.

Segunda forma: Para eliminar las anomalías de la primera forma normal, se deben eliminar las dependencias parciales. Una relación esta en segunda forma normal si esta en primera forma normal y se han eliminado las dependencias parciales. Para convertir una relación con dependencias parciales a la segunda forma normal, se crean dos nuevas relaciones, una con los atributos totalmente dependientes de la llave primaria y otra con los atributos dependientes de una de las partes de esa llave.

Tercera forma: Una relación esta en tercera forma normal (3NF) si esta en segunda forma normal y no tiene dependencias transitivas. Esto es, cada atributo no llave depende totalmente de la llave primaria y no hay dependencias transitivas.

A) DESCRIPCIÓN DE LA ESTRUCTURA DE LA BASE DE DATOS

Una empresa puede generar una o muchas ordenes y una orden debe ser generada por una empresa.

Una orden puede investigar a uno o muchos investigados y un investigado debe ser investigado a petición de una orden.

Una orden debe contener un tipo de estado y un tipo de estado puede estar en una o muchas ordenes.

Un estudio debe ser realizado a un investigado y un investigado puede amparar un estudio.

Un estudio es hecho por un investigador y un investigador puede ser asignado a uno o muchos estudios.

Un estudio puede ser facturado con una factura y una factura debe contener uno o muchos estudios.

Un estudio puede tener una o muchas acciones y una acción debe pertenecer a un sólo estudio.

Una acción debe tener un tipo de acción y un tipo de acción puede estar en una o muchas acciones.

Una acción puede hacer una o muchas llamadas y una llamada debe ser hecha por una acción.

Un estudio puede contener una o muchas referencias y una referencia es parte de un estudio.

Una referencia debe tener una compañía y una compañía puede ser referenciada por una o muchas referencias.

Una compañía puede tener una o muchas divisiones y una división debe pertenecer a una compañía. Ver figura 3.22 y 3.23

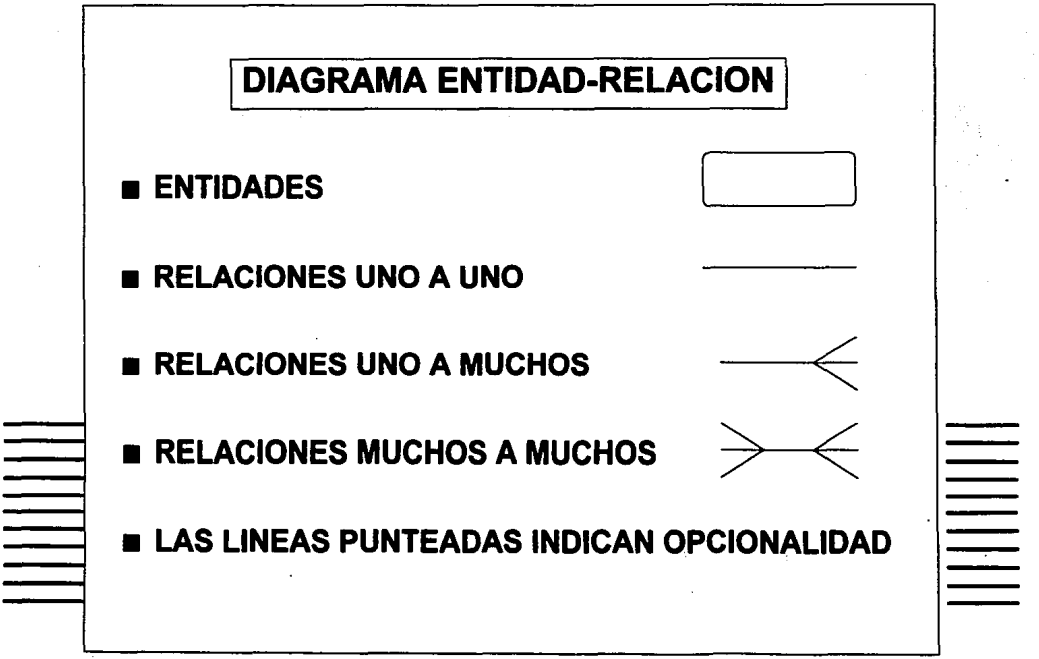


Figura 3.22 Elementos del diagrama entidad-relación.

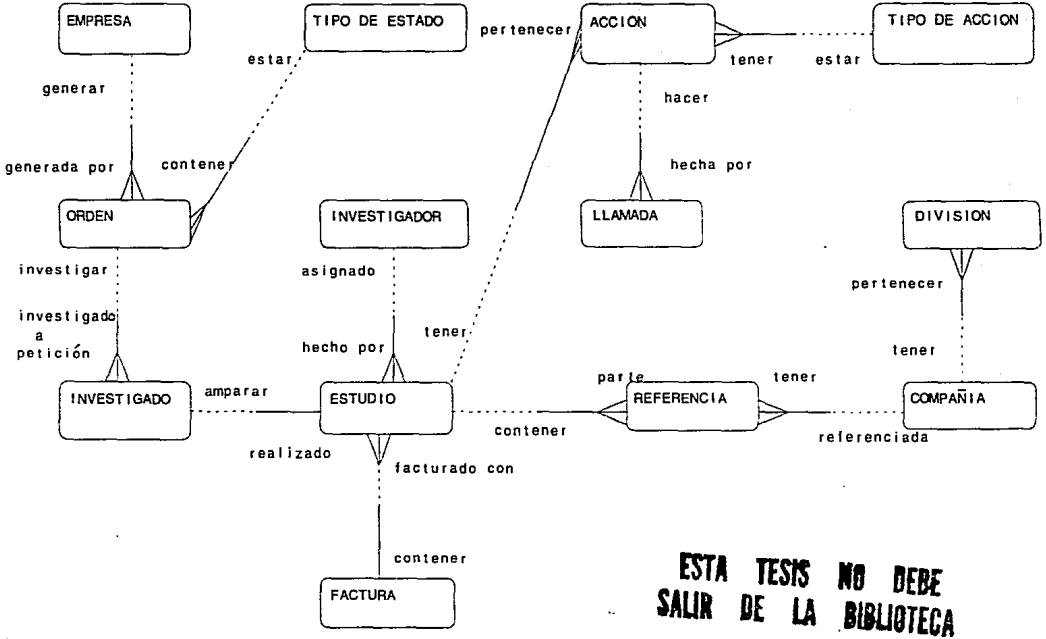


Figura 3.23 Diagrama entidad-relación.

B) DESCRIPCION DE LOS CAMPOS

NOMBRE	TIPO	DESCRIPCION
	ACCIONES	
ACC_NUM_RECIBO	SMALLINT	Número de recibo de dicha acción.
ACC_EST_ORD_NUM	SMALLINT	Número de orden del estudio de la acción.
ACC_TAC_CLAVE_ACC	CHAR(1)	Clave de la acción de acuerdo al tipo de acción de que se trate.
ACC_FECHA_ENTREGA	DATE	Fecha de entrega de dicha acción.
ACC_NUM_LLAMADAS	CHAR(2)	Número de llamadas con respecto a dicha acción.
	COMPAÑIAS	
COM_NOM	CHAR(35)	Nombre de la compañía.
COM_ID	SMALLINT	Identificador de la compañía.
COM_DIRECCION	CHAR(30)	Dirección de la compañía.
COM_COLONIA	CHAR(30)	Colonia en la cual se encuentra ubicada la compañía.
COM_INFORMADOR	CHAR(35)	Nombre del informador de dicha compañía.
COM_TEL1	CHAR(10)	Telefono de la compañía.
COM_TEL2	CHAR(10)	Telefono de la compañía.
COM_EXT_R_H	CHAR(4)	Extensión de la dirección de Recursos Humanos de la compañía.
COM_FAX	CHAR(10)	Número de fax de la compañía.
COM_COMENTARIO	CHAR(45)	Comentario de la compañía.
COM_UBIC_R_H	CHAR(30)	Ubicación de la dirección de Recursos Humanos.
COM_FECHA_INGRESO	DATE	Fecha de ingreso de la compañía.
	DIVISIONES	
DIV_ID	SMALLINT	Identificador de la división.
DIV_COM_ID	SMALLINT	Identificador de la relación división compañía.
DIV_NOM	CHAR(20)	Nombre de la división.
	EMPRESAS	
EMP_NOM	CHAR(35)	Nombre de la empresa.
EMP_ID	SMALLINT	Identificador de la empresa.
EMP_CLAVE	CHAR(4)	Clave de la empresa.
EMP_RFC	CHAR(15)	Registro Federal de Contribuyentes de la empresa.
EMP_DIRECCION	CHAR(30)	Dirección de la empresa.
EMP_COLONIA	CHAR(30)	Colonia donde se ubica la empresa.
EMP_NOM_SOL	CHAR(35)	Nombre de la empresa.
EMP_TEL1	CHAR(7)	Telefono de la empresa.
EMP_TEL2	CHAR(7)	Telefono de la empresa.
EMP_EXT_R_H	CHAR(4)	Numero de extensión de Recursos Humanos
EMP_INGRESO	DATE	Fecha de ingreso de la empresa.

ESTUDIOS		
EST_FECHA_CAPTURA	DATE	Fecha de captura del estudio.
EST_FECHA_VISITA	DATE	Fecha en que se realizo la visita para el estudio.
EST_FECHA_ENTREGA	DATE	Fecha de entrega del estudio.
EST_IDO_NOMBRE	CHAR(35)	Nombre del investigado en dicho estudio
EST_INV_ID_INVEST	SMALLINT	Identificador de la relación investigado y estudio.
EST_ORD_NUM	SMALLINT	Número de orden de dicho estudio.
EST_FAC_NUM	SMALLINT	Número de Factura de dicho estudio.
EST_NUM_DISCO	CHAR(10)	Número de disco donde se encuentra el estudio.
FACTURAS		
FAC_NUM	SMALLINT	Número de factura.
FAC_EMP_NOM	CHAR(35)	Nombre de la empresa de la factura.
FAC_FECHA_EMISION	DATE	Fecha en que se realizo la factura.
FAC_MONTO	MONEY(5,0)	cantidad monetaria por la cual es valida la factura.
INVESTIGADORES		
INV_ID_INVEST	SMALLINT	Identificador del investigador.
INV_COLONIA	CHAR(30)	Colonia en la cual se encuentra ubicada la dirección del investigador.
INV_DIRECCION	CHAR(30)	Dirección donde radica el investigador.
INV_NOM_INVEST	CHAR(35)	Nombre del investigador.
INV_HORARIO_LIBRE	CHAR(10)	Horario en el que se encuentra disponible el investigador.
INV_TELEFONO	CHAR(7)	Telefono del investigador.
INV_PROFESION	CHAR(15)	Profesión del investigador.
INV_COMENTARIO	CHAR(30)	Comentario hacerca del investigador.
INVESTIGADOS		
IDO_NOM	CHAR(35)	Nombre de la persona que se le esta elaborando el estudio.
IDO_ORD_NUM	SMALLINT	Número de orden de dicho investigado.
IDO_COLONIA	CHAR(30)	Colonia donde se encuentra hubicada la dirección del investigado.
IDO_DIRECCION	CHAR(30)	Dirección de dicho investigado.
IDO_CIUDAD	CHAR(20)	Ciudad en la que radica.
IDO_TEL_PART	CHAR(10)	Telefono particular del investigado si es que lo tiene.
IDO_PUESTO	CHAR(25)	Puesto que ocupara o ocupa el investigado.
IDO_ESTADO_REP	CHAR(15)	Estado del investigado.
IDO_EXT_OF	CHAR(4)	Número de extensión de la oficina.
IDO_MUNICIPIO	CHAR(20)	Municipio donde se localiza.

LLAMADAS		
LLA_EMP_LLAMADA	CHAR(35)	Llamadas de la empresa.
LLA_ESTADO_REP	CHAR(30)	Estado de dicha llamada.
LLA_TEL	CHAR(10)	Telefono de la llamada.
LLA_FECHA_LLAMADA	DATE	Fecha en que se hizo la llamada.
LLA_ACC_NUM_RECIBO	SMALLINT	Número de recibo de la acción de acuerdo a la llamda.
LLA_EST_ORD_NUM	SMALLINT	Número de orden del estudio de acuerdo a la llamada.
LLA_TAC_CLAVE_ACC	CHAR(1)	Clave de la acción dependiendo del tipo de acción que se trate y de la llamada.
ORDENES		
ORD_NUM	SMALLINT	Número de orden.
ORD_FECHA_ENTREGA	DATE	Fecha de entrega de la orden.
ORD_EMP_ID	SMALLINT	Identificador de acuerdo a la relación orden empresa.
ORD_TES_TIPO	CHAR(1)	Tipo de orden de acuerdo al tipo de estudio que se trate.
ORD_FECHA_PETICION	DATE	Fecha en que se solicito dicha orden.
REFERENCIAS		
REF_COM_ID	SMALLINT	Identificador de la relación compañia y referncias de esta.
REF_ORD_NUM	SMALLINT	Número de orden de dicha referencia.
REF_COMENTARIO	CHAR(30)	Comentario de acuerdo a la referencia.
TIPOS DE ACCIONES		
TAC_CLAVE_ACC	CHAR(1)	Clave correspondiente a una acción de acuerdo al tipo de esta.
TAC_DESCRIPCION	CHAR(15)	Descripción del tipo de acción.
TAC_COSTO_ACC	MONEY(3,0)	Costo de la acción dependiendo de que tipo sea.
TIPOS DE ESTUDIOS		
TES_TIPO	CHAR(1)	Tipo de estudio.
TES_DESCRIPCION	CHAR(20)	Descripción del tipo de estudio.
TES_COSTO	MONEY(4,0)	Costo del estudio dependiendo de que tipo sea dicho estudio.

3.5 ESPECIFICACIÓN DE PROGRAMAS

A continuación se muestra un ejemplo de la programación utilizada para el desarrollo de la subrutina ORDENES:

```
   FUNCTION Alta_Recepcion()
 DEFINE reg_orden  RECORD LIKE ordenes.*,
              resp, continuar,sigue
              CHAR(1), trab_orden
              RECORD LIKE ordenes.* ,
              num, num_serial
              SMALLINT,
reg_ido RECORD LIKE investigados.*,
trab_ido RECORD LIKE investigados.*,
nombre LIKE investigados.ido_nom

              OPTIONS PROMPT LINE 1


CLEAR SCREEN
OPEN FORM f_entord  FROM  "F_ENTORD"
OPEN FORM f_orden1 FROM "F_ORDEN1"
OPEN FORM f_orid  FROM "F_ORID"
OPEN FORM f_ido  FROM "F_IDO"
LET continuar = "S"
WHILE (continuar = "S" OR continuar ="s")
OPEN WINDOW vent  AT 8,6 WITH FORM "f_entord" attribute(border,yellow)
DISPLAY "O   R   D   E   N" AT 3,27
CLEAR FORM
LET num_serial = 0
INITIALIZE reg_orden.* TO NULL
INITIALIZE reg_ido.* TO NULL
PROMPT "MENSAGE: DESEAS SALIR <S>?" FOR CHAR resp
IF (resp = "S" OR resp ="s")
   THEN
      EXIT WHILE
   ELSE
      DISPLAY "" AT 1,2
      INPUT BY NAME reg_ido.ido_nom
      SELECT * INTO trab_ido.*  FROM  investigados
         WHERE ido_nom = reg_ido.ido_nom
         IF STATUS = NOTFOUND
```

```
THEN
   DISPLAY "NUEVA ORDEN" AT 1,2
   sleep 3
   CLOSE WINDOW vent
   OPEN WINDOW vent1  AT 2,3 WITH FORM "f_orden1"
   attribute(border,yellow) DISPLAY "INFORMACION DE LA ORDEN"
   AT  2,20
   SELECT MAX(ord_num) INTO num_serial FROM ordenes
    IF num_serial IS NULL
         THEN
            LET reg_orden.ord_num = 1
     ELSE
            LET reg_orden.ord_num = num_serial + 1
     END IF
    DISPLAY BY NAME reg_ido.ido_nom
    DISPLAY BY NAME reg_orden.ord_num
    INPUT BY NAME reg_orden.ord_emp_id THRU
     reg_orden.ord_tes_dias,reg_orden.ord_comentario
     LET reg_orden.ord_fecha_entrega = reg_orden.ord_fecha_peticion
     + reg_orden.ord_tes_dias + 1  DISPLAY BY NAME
    reg_orden.ord_fecha_entrega
    DISPLAY "" AT 1,3
    OPEN WINDOW vent2 AT 10,35 WITH FORM "f_ido" attribute(border)
    #DISPLAY "DATOS DEL INVESTIGADO" AT 1,8
    LET reg_ido.ido_ord_num = reg_orden.ord_num
    DISPLAY BY NAME reg_ido.ido_nom
    DISPLAY BY NAME reg_ido.ido_ord_num
    INPUT BY NAME reg_ido.ido_direccion THRU reg_ido.ido_puesto

ELSE
    DISPLAY  "EXISTE LA ORDEN" AT 1,2
    sleep 3
    CLOSE WINDOW vent
    SELECT * INTO trab_orden.*
    FROM ordenes WHERE  ord_num =
    trab_ido.ido_ord_num  DISPLAY ""
    AT 4,30
    OPEN WINDOW vent3 AT 7,3 WITH FORM "f_orid" attribute(border,yellow)
    DISPLAY "INFORMACION DE LA ORDEN" AT 1,24
    DISPLAY BY NAME trab_ido.ido_nom
    DISPLAY BY NAME trab_orden.*
    DISPLAY "Favor de Consultar la Orden" AT
    13,48 SLEEP 8
    CLOSE WINDOW vent3
    DISPLAY "" AT 4,30
```

```
                DISPLAY "" AT 10,10
                EXIT  WHILE
            END IF
PROMPT "ALTA DE LA ORDEN (S/N) "
FOR CHAR resp IF (resp = "S" OR resp ="s")
THEN
INSERT INTO ordenes VALUES(reg_orden.*)
INSERT INTO investigados
VALUES(reg_ido.*) DISPLAY "Orden dada
de alta" AT 15,26
sleep 3
CLOSE WINDOW vent2
DISPLAY "" AT  2,12
CLOSE WINDOW vent1
END IF
DISPLAY "" at 1,3
DISPLAY "" AT 7,8
END IF
END WHILE
CLOSE  FORM f_ent
CLOSE  FORM f_orden1
CLOSE  FORM f_orid
CLOSE  FORM f_ido
CLEAR SCREEN
END FUNCTION


#PROGRAMA CONSULTA DE ORDEN
FUNCTION  consulta_Recepcion()
 DEFINE reg_orden  RECORD LIKE
            ordenes.*, resp,
            continuar,sigue,pasar CHAR(1),
            trab_orden RECORD LIKE
            ordenes.* , num, num_serial
            SMALLINT,
            reg_ido RECORD LIKE
            investigados.*, trab_ido
            RECORD LIKE investigados.*,
            nombre LIKE
            investigados.ido_nom

              OPTIONS PROMPT LINE 1


            CLEAR SCREEN
```

```
OPEN FORM f_entord  FROM  "F_ENTORD"
OPEN FORM f_orden FROM "F_ORDEN"
OPEN FORM f_orid1  FROM "F_ORID1"
OPEN FORM f_ido  FROM "F_IDO"
LET continuar = "S"
OPEN WINDOW vent  AT 8,6 WITH FORM "f_entord" attribute(border,yellow)
WHILE (continuar = "S" OR continuar ="s")
CLEAR ido_nom
DISPLAY "O   R   D   E   N" AT 3,27
LET num_serial = 0
INITIALIZE reg_orden.* TO NULL
INITIALIZE reg_ido.* TO NULL
PROMPT "MENSAGE: DESEAS SALIR (S/N)?"  FOR
CHAR  resp
IF (resp = "S" OR resp ="s")
   THEN
      EXIT WHILE
   ELSE
      DISPLAY "" AT 1,2
      INPUT BY NAME reg_ido.ido_nom
 SELECT * INTO trab_ido.*  FROM  investigados
        WHERE ido_nom = reg_ido.ido_nom
        IF STATUS = NOTFOUND
          THEN
            DISPLAY "NO EXISTE LA ORDEN"
            AT 1,2     sleep 3

                    ELSE
             DISPLAY  "EXISTE LA ORDEN"
             AT 1,2    sleep 3

             SELECT * INTO trab_orden.*
             FROM ordenes WHERE
                ord_num = trab_ido.ido_ord_num
             DISPLAY "" AT 4,30
             OPEN WINDOW vent3 AT 2,2 WITH FORM "f_orid1" attribute(border,yellow)
             DISPLAY "INFORMACION DE LA ORDEN" AT 1,24
             DISPLAY BY NAME trab_ido.ido_nom
             DISPLAY BY NAME trab_orden.*
             SLEEP 5
            PROMPT "Deseas ver datos del investigado (S/N)" FOR CHAR pasar
             IF (pasar ="s" OR pasar="S")
                THEN
                   OPEN WINDOW vent2 AT 7,23 WITH FORM "f_ido" attribute(border)
                   DISPLAY "DATOS DEL INVESTIGADO" AT 1,8
```

```
                    DISPLAY BY NAME trab_ido.ido_ord_num THRU trab_ido.ido_puesto
                    PROMPT "Para salir presiona (S/s) " FOR CHAR resp
                      IF (resp = "S" OR resp ="s")
                       THEN
                        CLOSE WINDOW vent2
                        DISPLAY "" AT 2,12
                        CLOSE WINDOW vent3
                      END IF
                  ELSE
                      CLOSE WINDOW vent3
                END IF
          END IF
     END IF
END WHILE
CLOSE FORM f_entord
CLOSE FORM f_orden
CLOSE FORM f_orid1
CLOSE FORM f_ido
END FUNCTION

#PROGRMA DE ACTUALIZACION
DE ORDEN FUNCTION
   actualizar_Recepcion()
  DEFINE reg_orden RECORD LIKE
              ordenes.*,
          resp, continuar,sigue,paso,paso1
          CHAR(1), trab_orden RECORD LIKE
          ordenes.* ,
       num, num_serial SMALLINT,
          reg_ido RECORD LIKE
          investigados.*, trab_ido
          RECORD LIKE investigados.*,
          nombre LIKE
          investigados.ido_nom

                    OPTIONS
          PROMPT LINE 1


          CLEAR SCREEN
OPEN FORM f_entord FROM "F_ENTORD"      OPEN
FORM f_orden1 FROM "F_ORDEN1"
OPEN FORM f_orid FROM "F_ORID"
OPEN FORM f_ido FROM "F_IDO"
LET continuar = "S"
```

```
OPEN WINDOW vent  AT 8,6 WITH FORM "f_entord" attribute(border,yellow)
WHILE (continuar = "S" OR continuar ="s")

#OPEN WINDOW vent  AT 8,6 WITH FORM "f_entord" attribute(border,yellow)
DISPLAY "O     R  D  E  N" AT 3,27
LET num_serial = 0
INITIALIZE reg_orden.* TO NULL
INITIALIZE reg_ido.* TO NULL
PROMPT "MENSAGE: DESEAS SALIR (S/N)?"  FOR CHAR  resp
IF (resp = "S" OR resp ="s")
   THEN
      EXIT WHILE
   ELSE
      DISPLAY "" AT 1,2
      INPUT BY NAME reg_ido.ido_nom
      SELECT * INTO trab_ido.*  FROM  investigados
         WHERE ido_nom = reg_ido.ido_nom
         IF STATUS = NOTFOUND
          THEN
             DISPLAY "NO EXISTE ORDEN" AT 1,2
             SLEEP 3

           CLEAR ido_nom
          ELSE
             DISPLAY "EXISTE LA ORDEN" AT 1,2
             SLEEP 3

             OPEN WINDOW vent1  AT 2,3 WITH FORM "f_orden1" attribute(border,yellow)
             DISPLAY "INFORMACION DE LA ORDEN" AT  2,20
             SELECT * INTO trab_orden.*
             FROM ordenes WHERE  ord_num =
             trab_ido.ido_ord_num  DISPLAY BY
             NAME reg_ido.ido_nom
INPUT BY NAME trab_orden.ord_num THRU trab_orden.ord_comentario
 WITHOUT DEFAULTS      PROMPT "Se actualiza la informacion (S/N)" FOR CHAR paso
             IF (paso ="S" OR paso="s")
              THEN
                LET  trab_orden.ord_fecha_entrega = trab_orden.ord_fecha_peticion +
                 trab_orden.ord_tes_dias DISPLAY trab_orden.ord_fecha_entrega
                                  UPDATE ordenes SET * = trab_orden.* WHERE ord_num =
                                                    trab_ido.ido_ord_num
                DISPLAY "Información actualizada"  AT 12,42
                SLEEP 3
               END IF
                DISPLAY "" AT 1,3
```

```
                OPEN WINDOW vent2 AT 7,23 WITH FORM "f_ido" attribute(border)
                INPUT BY NAME trab_ido.ido_nom THRU trab_ido.ido_puesto
               WITHOUT DEFAULTS
                PROMPT "Se actualiza la informacion (S/N)" FOR CHAR paso1
                IF (paso1 ="S" OR paso1="s")
                   THEN
                      UPDATE investigados SET * = trab_ido.* WHERE ido_ord_num =
                      trab_orden.ord_num   DISPLAY "Información actualizada"  AT 18,33
                      SLEEP 4
                      CLOSE WINDOW vent2
                      CLOSE WINDOW vent1
                    ELSE
                      CLOSE WINDOW vent2
                      CLOSE WINDOW vent1
                  END IF
              END IF
         END IF
END WHILE
CLOSE  FORM f_ent
CLOSE  FORM f_orden
CLOSE  FORM f_orid
CLOSE  FORM f_ido
CLEAR SCREEN
END FUNCTION
```

CAPÍTULO 4 IMPLANTACIÓN DEL SISTEMA

La etapa final en el desarrollo de un sistema es la implantación, esta etapa marca el momento de verificar si el sistema desarrollado funciona realmente. Por lo que, una fase importante antes de que sea implantado el sistema es la prueba del mismo con el objetivo de descubrir errores de procesamiento.

El desarrollo de sistemas de software envuelve un serie de actividades de producción en las que las posibilidades de que aparezca la falibilidad humana son enormes. Los errores pueden empezar a darse desde el primer momento del proceso en el que los objetivos se pueden especificar en forma errónea o imperfecta, así como (los errores que aparecen) en los posteriores pasos de diseño y desarrollo. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidad.

La prueba de software es un elemento crítico para la garantía de calidad del software y representa un último repaso de las especificaciones, del diseño y de la codificación.

La prueba presenta una interesante anomalía para el ingeniero de software. Durante las fases anteriores de definición y de desarrollo, el ingeniero intenta construir el software partiendo de un control abstracto y llegando a una implementación tangible. A continuación llega la prueba. El ingeniero crea una serie de casos de prueba que intentan *demoler* el software que ha sido construido. De hecho, la prueba es uno de los pasos de la ingeniería del software que se puede ver (por lo menos, psicológicamente) como destructivo en lugar de constructivo.

La gente que desarrolla software es por naturaleza constructiva. La prueba requiere que se descarten ideas preconcebidas sobre la "corrección" del software que se acaba de desarrollar y superar cualquier conflicto de intereses que aparezcan cuando se descubren errores.

La IMPLANTACIÓN del Sistema para Estudios Socioeconómicos comprende la realización de pruebas y la instalación del sistema. Una vez que los resultados de cada prueba son satisfactorios, el sistema esta listo para ser usado por los usuarios por lo que se procede a la instalación del sistema en la máquina destino para ser liberado a producción .

4.1 PRUEBAS DEL SOFTWARE

En este tema se analizarán los fundamentos de las pruebas del software y las técnicas de diseño de casos de prueba.

El objetivo fundamental de la prueba del software es descubrir errores. Para conseguir este objetivo se planifican y se ejecutan una serie de pasos: *Pruebas de unidad, de integración, de validación y del sistema.* Las pruebas de unidad y de integración se centran en la verificación funcional de cada módulo y en la incorporación de los módulos en una estructura de programa. La prueba de validación demuestra el seguimiento de los requerimientos del software, y la prueba del sistema valida el software una vez que ha sido incorporado en un sistema mayor.

La prueba de software se basa en los siguiente alcances:

1.- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.
2.- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
3.- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Las metas anteriores suponen un cambio dramático del punto de vista. Nos quitan la idea que normalmente se tiene de que una prueba tiene éxito si no descubre errores. El objetivo es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

Si la prueba se lleva a cabo con éxito (de acuerdo con el objetivo anteriormente establecido), descubrirá errores en el software. Como ventaja secundaria, la prueba demuestra hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requerimientos de rendimiento. Además, los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software y de alguna manera indican la calidad del software como un todo.

Ahora analizaremos las técnicas para el diseño de casos de prueba, tomando como objetivo principal el derivar un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos en el software. Para llevar a cabo este objetivo, se usan dos categorías diferentes de técnicas de diseño de casos de prueba: *prueba de la caja blanca y prueba de la caja negra.*

- *Pruebas de la caja blanca.* Estas pruebas se centran en la estructura de control del programa. Se derivan casos de prueba que aseguren que durante la prueba se han ejecutado por lo menos una vez todas las sentencias del programa y que se ejercitan todas las condiciones lógicas. La prueba del camino básico, una técnica de la caja blanca, hace uso de grafos de programa (o matrices de grafo) para derivar el conjunto de caminos linealmente independientes que aseguren la cobertura. La prueba de bucles complementa a otras técnicas de la caja blanca proporcionando un procedimiento para ejercitar bucles de distintos grados de complejidad.

- *Pruebas de la caja negra.* Son diseñadas para validar los requerimientos funcionales sin fijarse en el funcionamiento interno de un programa. Las técnicas de prueba de la caja negra se centran en el dominio de información de un programa de forma que se proporcione una cobertura completa de prueba. La partición equivalente divide el dominio de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. En análisis de valores límite prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables. Los grafos de causa-efecto se incluyen en una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. Finalmente, la prueba de validación de datos asegura que los datos interactivos (conducidos por órdenes) son procesados correctamente.

4.2 INSTALACIÓN DEL SISTEMA

El proceso de instalación del sistema se coloca como la fase de transición entre la creación del producto del software desarrollado y la operación normal de éste, inmerso en las actividades de la empresa. Hasta este punto del proyecto de software, el producto se encuentra totalmente desarrollado y cumple con todas las especificaciones que le dieron origen, sólo falta insertarlo en el entorno operativo de la compañía e iniciar operaciones.

El proceso de instalación del sistema es tan importante, para el éxito del sistema, como las etapas para la creación del mismo, ya que un inicio de operaciones deficiente o mal implementado se puede traducir en rotundo fracaso inicial del producto de software. Se deben considerar todos los factores que influyen en la operación de un sistema como son: el factor humano (capacitación de los usuarios), el factor procedimental (implementar procedimientos administrativos acordes con la operación del sistema), el factor hardware (terminales, cableado, comunicaciones, etc.), y en su caso, el factor de conversión de sistemas (migración de información entre el sistema anterior y el nuevo), etc.

Por último, al concluir la etapa de pruebas con la instalación se llevó a cabo la liberación del sistema a producción, lo cuál se traduce como la operación del sistema por los usuarios y la integración a las actividades propias de la empresa.

CONCLUSIONES

Con la elaboración de este trabajo se buscó dar una descripción de la metodología que marco el plan a seguir en el desarrollo del sistema, basándonos principalmente en su ciclo de vida. El usar una metodología nos permitió realizar un desarrollo que garantiza el cumplimiento de los requisitos de calidad.

Es importante reconocer la importancia de trabajar en equipo y también reconocer que la Ingeniería es una carrera de carácter **interdisciplinario**, porque el hecho de trabajar y comunicarnos óptimamente con personas como programadores, analistas, capturistas, y personal de otras áreas, resulto de vital importancia para el buen desarrollo de este trabajo y de todos en general.

Con respecto a las etapas del desarrollo, tenemos lo siguiente:

Etapa de Análisis:

Normalmente a esta etapa no se le concede mucha importancia, ya sea por ser la primera o por falta de experiencia, mas sin embargo los errores que cometamos aquí van a repercutir gravemente y de una forma exponencial dependiendo en que momento sean corregidos, es decir a medida que mas tiempo tardemos en encontrar un error, mas vamos ir avanzando y mas va a ser lo que tenemos que modificar.

El objetivo principal del análisis es recabar toda la información necesaria para poder determinar claramente el objetivo y los alcances del sistema. Para hacer esto no basta con la información de las diferentes metodologías, si no que ademas se requiere un conjunto de actividades y actitudes que no están documentadas, como el que haya una buena simpatía entre el usuario y el desarrollador, que se tenga paciencia para repetir el mismo tema en diferentes juntas para lograr una retroalimentación adecuada y poder definir exactamente que es lo que se quiere y como debe llevarse a cabo, en las entrevistas acudir con las personas adecuadas, el que bajo ninguna circunstancia el desarrollador se salga de si, y el mantener siempre una buena imagen ante el cliente, cuidando el uso de la palabra, el vestido, el cumplimiento de lo que se ofrece y hasta la puntualidad con la que se labora. Este conjunto de conceptos entran bajo el rubro de la experiencia, partimos del hecho que nadie nace sabiendo, pero gracias a un buen trabajo en equipo y la comunicación entre sus partes, pudimos aprovechar la experiencia y capacidad de cada uno de nosotros.

Etapa de Diseño:

Una vez que se ya se tenían los requerimientos del sistema el siguiente paso fue el definir con que herramienta se desarrollaría, que recursos se necesitaban, en que tiempo se realizaría y cuales serían sus costos. En cuanto a la herramienta a utilizar se selecciono Informix por su precio y por su funcionalidad, mas sin embargo en el momento de la programación el esfuerzo fue mayor que el previsto por no contar con versiones recientes. En cuanto al desarrollo de la base de datos avanzamos rápidamente ya que se llevo a cabo mediante una herramienta generadora de sistemas llamada case, esta nos permitió plasmar gráficamente los objetos a controlar en el sistema (entidades) y su forma de interactuar entre si (relaciones), con esta información automáticamente se normalizo la base de datos y se genero el script de partida para la creación de las tablas de la base de datos.

Etapa de Construcción:

La construcción de formas, reportes y menús, se llevo a cabo por módulos y los fuimos acoplando a medida que se construían, tuvimos algunos problemas con el paso de parámetros de fecha, pero estos fueron resueltos dividiendo la variable.

Finalmente queremos hacer patente la responsabilidad y satisfacción que trae consigo la creación de un sistema integral para una empresa de estudios socioeconómicos, ya que ahora cuentan con una herramienta adecuada para su operación diaria y un medio eficiente para la toma de decisiones.

INICIALMENTE:

1.- Establecer una aplicación. Una aplicación diferente debe ser dada de alta para cada desarrollo de sistema.

NOTA (1): El título de la aplicación de esta pantalla aparecerá en la esquina superior izquierda de cada forma que es generada en esta aplicación usando CASE*Generator.

Método de Acceso:
1. Menú Pull-Down CASE *Generator Administration
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Application Access Control Menú
4. Seleccionar Application System Definition Screen

NOTA (2): Para fijar la aplicación Preferida (Prefered Application), entrar a cualquier pantalla de CASE*Dictionary y posicionar el cursor en el campo del nombre de la aplicación en el primer bloque. En ese momento, presionar la tecla de función {Block Menú} o {User Preference}. Esto desplegará una caja de texto que permite especificar la aplicación preferida.

ETAPA DE ESTRATEGIA:

1.- Desarrollar un Diagrama Funcional Jerárquico de Alto Nivel.

Método de Acceso:
1. Menú Pull-Down de Techniques de CASE*Designer
2. Seleccionar Function Hierarchy Diagrammer

2.- Desarrollar un Modelo Entidad-Relación.

Método de Acceso:
1. Menú Pull-Down de Techniques de CASE*Designer
2. Seleccionar Entity Relationship Diagram

3.- Crear Definición de Unidades de Negocios. Las unidades de negocios son comúnmente asociadas con sistemas distribuidos donde las diferentes funciones de los negocios son ejecutadas por diferentes organizaciones con la compañía o son realizadas en diferentes localidades.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Strategy Menú
4. Seleccionar Business Unit Definition Screen

4.- Validar relaciones de Entidades. Consultar las relaciones para asegurarse de que son correctas y que son comprensibles en la asociación entre dos entidades.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Strategy Menú
4. Seleccionar Relationship Definition Screen

5.- Empezar definiciones de términos: Definir cualquier término que es único para la empresa o que puede ser usado ambiguamente dentro de la corporación.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Strategy Menú
4. Seleccionar Relationship Definition Screen

6.- Análisis de Unidades de Negocios. Definir localización, problemas y asociar estos elementos a Unidades de Negocios.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Strategy Menú
4. Seleccionar Bussiness Unit Analysis Menú
5. Seleccionar Data Entry Screen of Choice
6. Reportar a través de User Extensibility Report Menú

7.- Análisis de Dirección de Negocios. Definir objetivos, factores críticos de éxito, claves de rendimiento y asociar estos elementos a unidades de negocios.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Strategy Menú
4. Seleccionar Bussiness Unit Analysis Menú
5. Seleccionar Data Entry Screen of Choice
6. Reportar a través de User Extensibility Report Menú Option.

8.- Revisar y Actualizar Información de soporte de estrategia. Revisar todas las pantallas dentro del menú de estrategia. Proporcionar información como sea requerida.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Strategy Menú
4. Cualquier pantalla de datos para información adicional y de soporte.

9.- Generar documentación de soporte de la Etapa de Estrategia. Correr los reportes de estrategia. Proporcionar información como sea requerida.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Strategy Menú
4. Seleccionar Strategy Reports Menú
5. Cualquier reporte que sea requerido.

Etapa de Análisis:

1.- Refinar la descomposición de la Jerarquía de Funciones.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar Entity Relationship Diagrammer

2.- Finalizar el Modelo Entidad-relación.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar Entity Relationship Diagrammer

3. Crear definición de funciones. Agregar información de frecuencia de funciones, notas, funciones lógicas, y algoritmos de negocios. Recordar que las palabras son muy importantes porque mapearán las funciones y entidades cuando se genere la matriz candidata de funciones/entidades. Tratar de referenciar la entidad que es usada por la función en la descripción de la función.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Function Requirements Menú
5. Seleccionar Function Definition Screen

4.- Definición de entidades. Agregar información acerca de entidades tal como volúmenes estimados, sinónimos y descripciones. Si se agregan muchos sinónimos es necesario asegurarse de que sean dados de alta en registros por separado (por ejemplo, un sinónimo por línea)

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Function Requirements Menú
5. Seleccionar Entity/attribute Definition Screen

5.- Definición de Dominios. Si los dominios son conocidos con anticipación, agregarlos ahora. Si no, éstos pueden ser creados cuando un atributo es asignado a los dominios en el siguiente paso.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Data Requirements Menú
5. Seleccionar Domain Definition Screen.

6.- Crear atributos para cada entidad. Agregar todos los atributos para cada entidad, indicar el nombre de atributo, si el atributo es parte de un dominio, si nulos son permitidos, tipo de dato, longitud, si el atributo es parte de un identificador único y notas sobre el atributo.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Data Requirements Menú
5. Seleccionar Entity/attribute Definition Screen.

7.- Dar Información Adicional de Atributos: Si se requiere, se puede adicionar más detalles de atributos como valores válidos o rango de valores.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Data Requirements Menú
5. Seleccionar Attribute definition Screen

8. Validar/Refinar Identificadores Únicos. Esta pantalla permite consultar y/o definir cualquier atributo que es parte del identificador único de cada entidad y permite definir cualquier identificador único que proviene de una relación. En otras palabras, para las terminaciones de relaciones de "Muchos" o de 1: M, cualquier llave foránea que podría ser parte de la llave primaria necesita ser definida por medio del nombre de la asociación. Usar la tecla de lista de valores para desplegar los atributos o las relaciones que deben ser parte del Identificador Único.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Data Requirements Menú.
5. Seleccionar Unique Identifier Definition Screen

9. Generar la Matriz Candidata de Función/Entidad. Esta utilería empata las funciones con las entidades que dichas funciones utilizan a través de SQL* Text Retrieval o de empatamiento de palabras.

Método de Acceso:
1. Menú Pull-DownTools de CASE* Designer
2. Seleccionar la opción de CASE* Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Utilities Menú
5. Generar la Matriz con Generate Candidate Function/Entity Matrix

10. Validar/Refinar Entidades usadas por cada Función Elemental. Verificar los resultados de la matriz Función/Entidad y determinar los usos apropiados de la entidad (por ejemplo, Cómo usa una función a la entidad, crea, selecciona, actualiza, borra, almacena histórico u otra función).

NOTA: Asociación de Funciones con Entidades y establecimiento de usos deben ser solamente para FUNCIONES ELEMENTALES. Asociaciones de Funciones a Entidades para funciones de alto nivel deberán ser borradas.

Método de Acceso: 1. Menú Pull-DownTools de CASE* Designer
2. Seleccionar Matrix Diagrammer
ó
1. Menú Pull-DownTools de CASE* Designer
2. Seleccionar la opción de CASE* Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Function Requirements Menú
5. Seleccionar Function Definition Screen (Página dos)

11. Crear la Matriz Función/Atributo. Esta utilería verá a las funciones que tienen entidades asociadas y jalará en los atributos para cada entidad listada, asignando el mismo uso (insert, retrieve, update, nullify) como la entidad.

Método de Acceso: 1. Menú Pull-DownTools de CASE* Designer
2. Seleccionar la opción de CASE* Designer
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Utilities Menú
5. Crear la Matriz con Create Function/Attribute Matrix

12. Validar/Refinar Atributos usados por cada Función Elemental. Determinar atributos válidos usados por cada función y verificar su uso. Confirmar que ninguna llave primaria o campo mandatorio indique "Y" en la columna "Nullify".

Método de Acceso: 1. Menú Pull-Down Techniques de CASE*Designer
2. Seleccionar Matrix Diagrammer
ó
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Function Requirements
5. Seleccionar Function Definition Screen (Página Dos)

13. Ejecutar el Cross-Checking. Checar el cruce de Funciones/Entidades y/o Funciones/atributos o cualquier otro cruce crítico que proporcione confirmación útil para completar el Análisis.

Método de Acceso: 1. Menú Pull-Down Techniques de CASE*Designer
2. Seleccionar Matrix Diagrammer
o
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Dictionary
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Function Requirements Menú
5. Seleccionar Function definition Screen (Página Dos)

14. Ejecutar el Cross-Checking. Checar el cruce de Funciones/Entidades y/o Funciones/atributos o cualquier otro cruce crítico que proporcione confirmación útil para completar al Análisis.

Método de Acceso: 1. Menú Pull-Down Techniques de CASE*Designer
2. Seleccionar Matrix Diagrammer

15. Desarrollar el Diagrama de Flujo de Datos. Los DFD's pueden ser usados gráficamente verificando el cruce de funciones y uso de entidades.

Método de Acceso: 1. Menú Pull-Down Techniques de CASE*Designer
2. Seleccionar Matrix Diagrammer

16. Revisar y Actualizar Información de soporte de Análisis. Revisar todas las pantallas con el Menú de Análisis. Proporcionar la información como sea requerida.

Método de Acceso:
1. Menú Pull-DownTools de CASE*Desingner
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Analysis Menú
4. Todos los Menúes de Análisis
5. Cualquier Pantalla para información adicional y de soporte.

17. Generar la Documentación de Soporte de la Etapa de Análisis. Correr los reportes como sean requeridos para la documentación y entendimiento del sistema.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer.
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Analysis Menú
4. Seleccionar Analysis Data Reports Menú
5. Cualquier reporte de interés
   Nota: La opción de Strategy & Análisis Quality Checks Menú puede ser utilizada

Etapa de Diseño.

1.- Mapear las Entidades a Tablas. Presionar Commit para indicar que una entidad debe convertirse en tabla.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Data Design Utilities Menú
5. Seleccionar Fastpath Table Mapping Screen

2.- Confirmar y Actualizar la Definición de Tablas: Agregar detalles acerca de las tablas como título de despliegue y descripción. Lo más importante es decidir si el prefijo de la columna será usado. Si el prefijo es usado, verificar su nombre para ver si es adecuado. Si el prefijo no es usado, es necesario limpiar el campo.

Método de Acceso:
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar DataBase Design Menú
5. Seleccionar Table Definition Screen (Página uno solamente).

3.- Crear el Diseño de la Base de Datos por Default: Seleccionar "V" para validar el diseño pero no cargar el diseño a las tablas de CASE; "I" para insertar el diseño a las tablas de CASE, o "R" para reemplazar características existentes a las columnas o para agregar nuevas columnas. Se preguntará cuáles tablas se desean para el diseño -usar % para todas las tablas.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Default Database Design

4.- Agregar información de secuencias. Definir cualquier secuencia que pueda ser usada automáticamente para generar valores de llaves primarias. Las secuencias deberán ser definidas a través de esta pantalla antes de que las columnas sean asignadas. Información como el nombre de la secuencia, valores inicial y máximo, e incremento de secuencia podrán ser definidos.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Sequence definition Screen

5.- Confirmar y Actualizar información de detalle de Columnas:
Se agrega información de secuencias, descripción de columnas, valores de default, prompts de columnas, longitud de despliegue, reglas de despliegue de secuencias y columnas.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Default Database Design Menú
5. Seleccionar Table Definition Screen

6.- Agregar información de Validación y Derivación de Columnas:
Se agrega información de reglas de validación de columnas, reglas de derivación y se documentan rangos de columnas.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Default Database Design Menú
5. Seleccionar Tables, Columns and Keys Menú
6. Seleccionar Column Validation and Derivation Screen

7.- Agregar información de Columnas. Se agrega información de constraints de columnas como llaves primarias y llaves foráneas y reglas de integridad referencial.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Default Database Design Menú
5. Seleccionar Tables, Columns and Keys Menú
6.- Seleccionar Table Key Constraints Definition Screen

8.- Crear el Diseño de Índices por Default: Esta utilería crea índices únicos de cada llave primaria y un índice no-único en cada llave foránea. Es necesario asegurarse de que se han definido por completo las llaves primarias y foráneas antes de correr esta utilería.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Data Design Utilities Menú
5.- Seleccionar Default Index Design

9.- Confirmar y Verificar las definiciones de índices: Revisar índices, confirmar orden de columnas y adicionar índices necesarios.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Database design Menú
5.- Seleccionar default Index Design

10.- Definir opciones de DBA: Si se desea se puede almacenar información de DBA en las tablas de CASE, tales como bases de datos, grupos de usuarios, tablespaces, rollback, segments, storages y/o archivos físicos.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar DBA Options Menú
5. Cualquier pantalla de entrada

11.- Crear el Diseño por default de Módulos. Correr el script que toma la definición de funciones y su asociación a entidades/atributos para determinar los módulos por default del sistema. Si una función solo tiene uso de "Select" y su tiempo de respuesta requerido es "Immediate", entonces se creará un módulo tipo pantalla. Si la función tiene cualquier otro uso además de "Select" y la respuesta requerida es "Overnight", entonces se creará un Módulo tipo Utilería.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Module Design Utilities Menú
5. Seleccionar Default Module Specification

12.- Aceptar los Módulos del Sistema. Aceptar los módulos que serán incluidos en el sistema presionando commit.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Module Design Utilities Menú
5. Seleccionar Candidate Module Specification Acceptance Screen

13.- Agregar información de control de Módulos: Agregar información acerca de cómo serán asignados los módulos al proyecto.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Module Control Menú
5. Seleccionar Module planning Screen

14.- Agregar información de factores de peso para cada Módulo:
Agregar información sobre cuánto tiempo se asignará a cada tarea del módulo.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Module Control Menú

5. Seleccionar Module Weigthing factor Screen

15.- Validar definición de módulos. Verificar que cada función elemental esté implementada por un módulo. Se puede agregar información sobre descripción de Módulos, notas de desarrollo, etc.

NOTA: Verificar cómo va a ser generado cada módulo porque eso determina la estructura del menú.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Module Design Menú
5. Seleccionar Module Definition Screen

16.- Verificar/Agregar uso de Módulos: Verificar cómo cada módulo va a usar las tablas. NOTA: Con la opción de Default Module design esta pantalla será llenada. A continuación se presentan maneras para llenar esta pantalla si no se ha corrido esta opción.

1) Copy From Module. Copia cualquier información acerca de la tabla y columnas que el módulo usa.
2) Copy From Function. Copia la información de matrices funciones a entidades/atributos que se agregaron durante el análisis. Considerar que las llaves foráneas no se copiarán ya que no son consideradas como atributos.
3) Agregar el nombre de la tabla asociada manualmente. Para agregar columnas, presionar *list of values* y seleccionar las columnas. Si todas las columnas serán incluidas, presionar lista de valores nuevamente.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Module Design Menú
5. Seleccionar Module Data Usage (Detailed) screen.

17.- Crear el Diseño de Menú del Sistema por default: Correr el scrip que determina el Menú del sistema por default basado en la asignación de funciones a unidades de negocios.

NOTA: Todos los módulos deberán ser definidos a su tiempo. Esta utilería utiliza la definición de los módulos y la asociación a unidades de negocios.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Crear la especificación con Create Menú System

Specification

18.- Aceptar los menúes del sistema. Aceptar los menúes que serán incluidos en el sistema presionando commit.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Module Design Utilities Menú
5. Seleccionar Candidate Menú Specification Acceptance Screen

19.- Establecer las preferencias del usuario a nivel módulo.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Module Design Utilities Menú
5. Seleccionar User Preferences Screen

20.- Ejecutar el Cross-Checking

Método de Acceso.
1. Menú Pull-Down Techniques de CASE*Designer
2. Seleccionar Matrix Diagrammer

21.- Revisar y actualizar información de soporte de la etapa de Diseño: Revisar todas las pantallas del Menú de Diseño.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Database Design Menú

22.- Generar documentación de soporte de la Etapa de Diseño: Correr los reportes como sean requeridos para la documentación y entendimiento del sistema.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Design Menú
4. Seleccionar Database Design Reports Menú

Etapa de Implementación.

1.- Crear usuarios y/o objetos de DBA: El comando generador de DBA crea objetos de usuarios como tablas, vistas, índices, secuencias y clusters, y también crea objetos de DBA como tablespaces, rollback segments y grants. Esta utilería crea comandos DDL deseados.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar DDL Command Generator

2.- Correr utilerías de CASE*Generator. Textos de ayuda, valores de referencia, y tablas de control. Esta pantalla permite especificar cuáles módulos o tablas correrán esta utilería

* El texto de Ayuda contiene todos los textos de ayuda para las aplicaciones en una tabla de la base de datos.
* Los valores de Referencias contienen todos los valores válidos para cada columna y/o dominio en una tabla de la base de datos.
* Las Tablas de Control crean secuencias para todos los sistemas generados para llaves primarias.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator Utilities Menú
6. Seleccionar CASE*Generator for Reference Tables

3.- Establecer preferencias para SQL*Forms. Establecer todas las preferencias para triggers deseados. Generalmente se usan valores de default.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for SQL*Forms Menú
6. Seleccionar User Preferences Screen

4.- Generar SQL*Forms. Este paso debe ser corrido para cada módulo de SQL*Forms.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for SQL*Forms Menú
6. Seleccionar CASE*Generator for SQL*Forms

5.- Establecer preferencias de Reportes. Establecer los triggers deseados para la generación de reportes. Se usarán valores de default.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for Reports
6. Seleccionar User Preferences Screen

6.- Generar reportes. Este paso debe ser corrido para cada módulo tipo reporte.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for Reports Menú
6. Seleccionar CASE*Generator for Reports

7.- Establecer preferencias para Menúes. Establecer los triggers deseados para los menúes.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for SQL*Menú
6. Seleccionar User Preferences Screen.

8.- Generar SQL*Menú: Este paso deberá ser corrido para cada módulo tipo menú.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for SQL*Menú Menú
6. Seleccionar CASE*Generator for SQL*Menú

9.- Establecer preferencias de la aplicación.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for Applications Menú
6. Seleccionar User Preferences Screen

10.- Generar aplicaciones: Este paso debe ser corrido para cada aplicación que se desee.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for Applications Menú
6. Seleccionar CASE*Generator for applications.

11.- Generar documentación de soporte para la etapa de Implementación.
Correr los reportes de implementación necesarios para el entendimiento del sistema.

Método de Acceso.
1. Menú Pull-Down Tools de CASE*Designer
2. Seleccionar la opción de CASE*Designer
3. Seleccionar Implementation Menú
4. Seleccionar CASE*Generator Main Menú
5. Seleccionar CASE*Generator for Reports Menú
6. Cualquier reporte de interés

APÉNDICE B GLOSARIO DE CONCEPTOS

DEFINICIONES

ACOPLAMIENTO Y COHESIÓN: La modificación de un bloque común de datos o de control puede requerir de modificaciones en todas las rutinas que se encuentren acopladas a ese bloque; por otro lado, si los módulos se comunican solamente por los parámetros y si las interfases entre módulos permanecen constantes, los detalles internos de los módulos pueden ser modificados sin tener que modificar la rutina que usan los módulos modificados.

La comunicación entre módulos incluye el pasaje de datos de elementos de control (como banderas, interruptores, etiquetas y nombres de procedimientos) así como las modificaciones del código de un módulo hacía otro. El grado de acoplamiento es menor para la comunicación de datos, mayor para la de conceptos de control y mucho mayor en el caso de módulos que modifican el código de otros módulos.

El acoplamiento entre módulos puede ser considerado dentro de la escala siguiente:

1. *Acoplamiento del contenido.* Ocurre cuando un módulo modifica los valores locales o las instrucciones de algún otro módulo; este puede ocurrir en programas en lenguaje ensamblador.

2. *Acoplamiento de zonas compartidas.* En este los módulos son atados en forma conjunta por medio de zonas globales para las estructuras de datos: un ejemplo, es en los programas escritos en FORTRAN.

3. *Acoplamiento de control.* Este incluye el pasaje de banderas de control, ya sea como parámetros o en forma global entre los módulos de tal forma que un módulo controle la secuencia de proceso de otro.

4. *Acoplamiento por zonas de datos.* Este es similar al de zonas compartidas excepto que los elementos globales son compartidos en forma selectiva entre las diversas rutinas que requieren de los datos.

5. *Acoplamiento de datos.* Incluye el uso de listas de parámetros para pasar a los elementos entre rutinas. La forma más deseada de acoplamiento es ciertamente una combinación de zonas de datos y de acoplamiento de datos.

La cohesión interna de un módulo se mide en términos de la fuerza de unión de los elementos dentro del módulo y esta ocurre en el siguiente orden:
1. *Cohesión coincidental.* Ocurre cuando los elementos de un módulo no tienen relación aparente entre cada uno de ellos, esto se da cuando se crea un módulo de un grupo de instrucciones no relacionadas que aparecen en diversos módulos.

2. *Cohesión lógica.* Implica algunas relaciones entre los elementos de un módulo, como por ejemplo en uno que desempeñe todas las funciones de entrada y salida.

3. *Cohesión temporal.* Presentan muchas de las desventajas de los lógicamente unidos; sin embargo se encuentran arriba en la escala de la cohesión debido a que todos los elementos son ejecutados en un momento dado sin requerir de ningún parámetro o lógica alguna para determinar que elemento debe ejecutarse. Por ejemplo, un módulo dedicado a la inicialización de un sistema o programa.

4. *Cohesión en la comunicación.* Se refieren al mismo conjunto de datos de entrada o salida; por ejemplo la instrucción imprima y perfore el archivo de salida.

5. *Cohesión secuencial.* Ocurre cuando la salida de un elemento es la entrada para el siguiente; por ejemplo la instrucción lea la siguiente transacción y actualice el archivo maestro.

6. *Cohesión funcional.* Representa un tipo fuerte, y por ende deseable de amarre de los elementos de un módulo debido a que los elementos se encuentran relacionados al desempeño de una sola función. Ejemplo calcule la raíz cuadrada, obtenga un número aleatorio y escriba el registro en el archivo de salida.

7. *Cohesión informacional.* Ocurre cuando el módulo contiene una estructura de datos compleja, así como varias rutinas que manejan dicha estructura; cada rutina del módulo presenta unión funcional; esta cohesión es la realización total de abstracción de los datos. Esta cohesión es similar a la cohesión en la comunicación, sin embargo, difieren en que la comunicación implica que todo el código en el módulo sea ejecutada en cada llamada al mismo, por su parte, la cohesión informacional requiere que solamente el segmento con cohesión funcional sea ejecutado al ser llamado el módulo.

En resumen, la meta de la modularización de un sistema de programación por el uso de los criterios del acoplamiento y la cohesión es la de producir sistemas que tengan acoplamiento de zonas de datos y acoplamiento de datos entre los módulos y además que cuenten con cohesión funcional e informacional en los elementos de cada módulo

CARDINALIDAD: Número de tuplas de una relación.

CONECTIVIDAD: Una relación especifica el tipo de asociación de las ocurrencias de las entidades de la relación. Los valores de la conectividad son "uno" o "muchos". El número real asociado con el término muchos es llamado la cardinalidad de la conectividad. Los tipos básicos de conectividad son los siguientes: uno a uno, uno a muchos y muchos a muchos. Conceptos básicos del modelo relacional.

CRITERIOS DE MODULACIÓN: El diseño arquitectónico tiene como meta producir sistemas modulares de programación bien estructurados, cada módulo de programación tiene las siguientes características:

1. Los módulos contienen instrucciones, lógica de proceso y estructuras de datos.

2. Los módulos pueden ser compilados aparte y almacenados en una biblioteca.

3. Los módulos pueden quedar incluidos dentro de un programa

4. Los segmentos de un módulo pueden ser utilizados al invocar un nombre con algunos parámetros.

5. Los módulos pueden usar a otros módulos.

Entre los criterios de modularización se incluye el criterio convencional en que cada módulo junto con sus correspondientes submódulos corresponden a un paso del proceso en la secuencia de ejecución; así mismo, el criterio del ocultamiento de la información, en que cada módulo oculta a otros módulos una decisión difícil o modificable del diseño; al criterio de la abstracción de los datos, en que cada módulo oculta los detalles de representación de una estructura de datos importante debajo de las funciones que accesan y modifican dicha estructura; a los niveles de abstracción, en que los módulos y las colecciones de los mismos proporcionan una jerarquía de servicios más complejos; el acoplamiento y la cohesión, por medio de los cuales un sistema se estructura para maximizar los elementos de cohesión de un sistema y minimizar el acoplamiento entre módulos, así como, a la modelación de problemas por medio de la cual la estructura modular de un sistema se ajusta a la estructura del problema a resolver.

DISEÑO DESCENDENTE: El diseño descendente se basa en la noción de que la estructura del problema debe determinar la estructura de la solución del software. Esta metodología utiliza la característica humana más fundamental para la solución de problemas: la abstracción.

Según el diccionario, la abstracción es el "proceso de eliminar de una idea sus acompañamientos concretos". La idea se considera como una entidad abstracta, sin especificar cómo se realiza esa entidad. A medida que el diseño progresa, cada componente se refina en sus propias operaciones fundamentales y el proceso continúa hasta que se formula un diseño de bajo nivel.

La formulación y descripción de un diseño de software incluyen varias etapas:

1. Estudiar y comprender el problema. Sin esta comprensión, es imposible el diseño efectivo del software.

2. Identificar las características generales de, al menos, una posible solución. En esta etapa suele ser útil identificar varias soluciones y evaluar cada una de ellas. Se debe de elegir la solución más simple posible. Es de particular importancia no permitir detalles de aplicaciones de bajo nivel, de las cuales tenga conocimiento el diseñador, que infieren en la elección de una solución.

3. Construir un diagrama de flujo de datos que muestre las transformaciones generales de los datos del sistema. Si esto parece imposible, quizá no se haya comprendido de manera adecuada el problema.

4. Mediante el diagrama de flujo de datos, construir un diagrama de estructura que muestre las unidades de programa relacionadas con la solución.

5. Describir cada abstracción utilizada en la solución mediante un lenguaje de descripción como Ada/PDL. Es probable que en las primeras etapas del diseño esto consista casi exclusivamente en una descripción en lenguaje natural.

Después de haber formulado y descrito una solución inicial de alto nivel, el proceso de solución del problema debe repetirse para cada abstracción utilizada. Este proceso de refinamiento continúa hasta que se ha preparado una especificación de bajo nivel para cada abstracción.

DOMINIO: Conjunto de todos los valores posibles para un atributo.

ENTIDAD: Es una "cosa" que se puede distinguir. En un diagrama ER las entidades se representan con un rectángulo, una entidad es el objeto principal del cual se tiene que almacenar información.

ESTRUCTURA: La estructura es una característica fundamental, el uso de una estructuración permite que un sistema grande sea definido en términos de unidades más pequeñas y manejables con una clara definición de las relaciones entre las diferentes partes del sistema.

GRADO DE LA TUPLA: Número de atributos que tiene una tupla (*n* de una *n*-tupla).

INTEGRIDAD REFERENCIAL: El valor llave debe existir en la relación asociada para integridad de la base de datos.

LLAVE PRIMARIA: El atributo para el cual no más de una tupla puede tener el mismo valor (combinación).

LLAVE CANDIDATA: Atributo o conjunto de atributos que pueden servir como llave primaria.
LLAVE SECUNDARIA: Todas aquellas llaves candidatas que no se eligieron como llave primaria.

LLAVE EXTRANJERA: Llave primaria que es llave en otra relación.

MODULARIDAD: Un módulo es una asignación de trabajo para un programador, en el sentido de que todos los sistemas modulares incorporan colecciones de abstracciones, en la que cada abstracción funcional o de datos, así como la abstracción en el control manejan un aspecto local del problema por resolver. Los sistemas modulares consisten en unidades claramente definidas y manejables con las interfases claramente definidas entre los diversos módulos, la modularidad ciertamente mejora la claridad del diseño, que a su vez facilita la instrumentación, la depuración, las pruebas, la documentación y el mantenimiento de un producto de programación.

RELACIONES: Se representa como un rombo con líneas conectando las entidades relacionadas, normalmente un verbo transitivo correspondiente a la relación. Hay muchas relaciones entre entidades y algunas son de interés para la empresa, el diseñador de la base de datos es el responsable de seleccionar las relaciones relevantes. también debe especificar el tipo de asociación de las relaciones (uno a uno, uno a muchos, muchos a muchos).

REQUERIMIENTO: Es una característica del sistema o una descripción de algo que el sistema será capaz de realizar acorde a su propósito.

TUPLA: Conjunto de valores que componen un renglón de la relación. Es equivalente a una instancia de un registro.

```
REM
REM This ORACLE V6 RDBMS command file was generated by CASE*Dictionary .
REM                          on 17-MAY-94
REM
REM For application system TESIS version 1
REM
SET SCAN OFF

REM  Objects being generated in this file are:-
REM TABLE
REM         ACCIONES
REM         COMPANIAS
REM         DIVISIONES
REM         EMPRESAS
REM         ESTUDIOS
REM         FACTURAS
REM         INVESTIGADORES
REM         INVESTIGADOS
REM         LLAMADAS
REM         ORDENES
REM         REFERENCIAS
REM         TIPOS_DE_ACIONES
REM         TIPOS_DE_ESTADOS

REM
REM    Created from Entity ACCION by CASE on 17-MAY-94
REM
PROMPT
PROMPT Creating Table ACCIONES
CREATE TABLE acciones(
 acc_número_de_recibo                NUMBER(7,0)        NOT NULL,
 acc_est_clave_del_estudio           CHAR(8)            NOT NULL,
 acc_tac_clave_de_la_accion          NUMBER(3,0)        NOT NULL,
 acc_fecha_de_entrega                DATE               NOT NULL,
 acc_número_de_llamadas              NUMBER(2,0)        NULL
)
;
```

```
COMMENT ON TABLE acciones
      IS 'Created from Entity ACCION by CASE on 17-MAY-94';

REM
REM    Created from Entity COMPANIA by CASE on 17-MAY-94
REM
PROMPT
PROMPT Creating Table COMPANIAS
CREATE TABLE companias(
 com_identificador_de_la_empres  NUMBER(4,0)                    NOT NULL,
 com_colonia                     CHAR(30)                       NOT NULL,
 com_nombre_de_la_empresa        CHAR(30)                       NOT NULL,
 com_número_telefonico_1         CHAR(10)                       NOT NULL,
 com_direccion                   CHAR(30)                       NOT NULL,
 com_informador                  CHAR(30)                       NOT NULL,
 com_fecha_de_actualizacion      DATE                           NOT NULL,
 com_ext_recursos_humanos        CHAR(10)                       NULL,
 com_comentario                  CHAR(30)                       NULL,
 com_número_de_fax               CHAR(10)                       NULL,
 com_número_telefonico_2         CHAR(10)                       NULL,
 com_ubicacion_recursos_humanos  CHAR(30)  NULL
)
;

COMMENT ON TABLE companias
      IS 'Created from Entity COMPANIA by CASE on 17-MAY-94';

REM
REM    Created from Entity DIVISION by CASE on 17-MAY-94 REM
PROMPT
PROMPT Creating Table DIVISIONES
CREATE TABLE divisiones(
 div_identificador_de_la_divisi  NUMBER(3,0)  NOT NULL,
 div_com_identificador_de_la_em  NUMBER(4,0)                    NOT NULL,
 div_nombre_de_la_division       CHAR(30)                       NOT NULL
)
;

COMMENT ON TABLE divisiones
      IS 'Created from Entity DIVISION by CASE on 17-MAY-94';

REM
REM    Created from Entity EMPRESA by CASE on 17-MAY-94 REM
PROMPT
PROMPT Creating Table EMPRESAS
```

```
CREATE TABLE empresas(
 emp_identificador_de_la_empres NUMBER(3,0)          NOT NULL,
 emp_colonia                    CHAR(30)             NOT NULL,
 emp_ext_recursos_humanos       CHAR(10)             NOT NULL,
 emp_direccion                  CHAR(30)             NOT NULL,
 emp_nombre_del_solicitante     CHAR(30)             NOT NULL,



 emp_número_de_telfono_1        CHAR(10)             NOT NULL,
 emp_nombre_de_la_empresa       CHAR(30)             NOT NULL,
 emp_número_de_telefono_2       CHAR(10)             NULL
)
;

COMMENT ON TABLE empresas
      IS 'Created from Entity EMPRESA by CASE on 17-MAY-94';

REM
REM    Created from Entity ESTUDIO by CASE on 17-MAY-94 REM
PROMPT
PROMPT Creating Table ESTUDIOS
CREATE TABLE estudios(
 est_clave_del_estudio          CHAR(8)              NOT NULL,
 est_fecha_de_captura           DATE                 NOT NULL,
 est_fecha_y_hora_de_visita     DATE                 NOT NULL,
 est_fecha_de_entrega           DATE                 NOT NULL,
 est_ido_nombre                 CHAR(30)             NOT NULL,
 est_inv_identificador_del_inve NUMBER(2,0) NOT NULL,
 est_ord_número_de_orden        NUMBER(3,0)          NOT NULL,
 est_fac_número_de_factura      NUMBER(5,0)          NULL,
 est_número_de_disco            CHAR(10)             NULL
)
;

COMMENT ON TABLE estudios
      IS 'Created from Entity ESTUDIO by CASE on 17-MAY-94';

REM
REM    Created from Entity FACTURA by CASE on 17-MAY-94 REM
PROMPT
PROMPT Creating Table FACTURAS
CREATE TABLE facturas(
 fac_número_de_factura          NUMBER(5,0)          NOT NULL,
 fac_fecha_de_emision           DATE                 NOT NULL
```

```
)
;

COMMENT ON TABLE facturas
      IS 'Created from Entity FACTURA by CASE on 17-MAY-94';

REM
REM    Created from Entity INVESTIGADOR by CASE on 17-MAY-94
REM
PROMPT
PROMPT Creating Table INVESTIGADORES
CREATE TABLE investigadores(
 inv_identificador_del_investig NUMBER(2,0) NOT NULL,
 inv_colonia                                CHAR(30)
      NOT NULL,
 inv_direccion                              CHAR(30)
      NOT NULL,
 inv_nombre_del_investigador                CHAR(30)
      NOT NULL,
 inv_horario_libre                          CHAR(10)
      NULL,
 inv_telefono                               CHAR(10)
      NULL,
 inv_profesion                              CHAR(15)
      NULL
)
;

COMMENT ON TABLE investigadores
      IS 'Created from Entity INVESTIGADOR by CASE on 17-MAY-94';

REM
REM    Created from Entity INVESTIGADO by CASE on 17-MAY-94 REM
PROMPT
PROMPT Creating Table INVESTIGADOS
CREATE TABLE investigados(
 ido_nombre                                 CHAR(30)
      NOT NULL,
 ido_ord_número_de_orden                    NUMBER(3,0)
      NOT NULL,
 ido_colonia                                CHAR(30)
      NOT NULL,
 ido_direccion                              CHAR(30)
      NOT NULL,
```

```
ido_est_clave_del_estudio                CHAR(8)
      NULL,
ido_ciudad                               CHAR(20)
      NULL,
ido_telefono_particular                  CHAR(10)
      NULL,
ido_puesto                               CHAR(25)
      NULL,
ido_estado                               CHAR(15)
      NULL,
ido_ext_de_oficina                       CHAR(10)
      NULL,
ido_municipio                            CHAR(20)
      NULL
)
;

COMMENT ON TABLE investigados
      IS 'Created from Entity INVESTIGADO by CASE on 17-MAY-94';

REM
REM    Created from Entity LLAMADA by CASE on 17-MAY-94
REM
PROMPT
PROMPT Creating Table LLAMADAS
CREATE TABLE llamadas(
pkey_!default                            CHAR(8)
      NOT NULL,
lla_est_clave_del_estudio                CHAR(8)
      NULL,
lla_tac_clave_de_la_accion               NUMBER(3,0)
      NULL,
lla_empresa_a_la_que_se_llamo            CHAR(30)
      NOT NULL,
lla_estado                               CHAR(30)
      NOT NULL,
lla_telefono                             CHAR(15)
      NOT NULL,
lla_fecha_de_la_llamada                  DATE
      NOT NULL,
lla_acc_número_de_recibo                 NUMBER(7,0)
      NULL
)
;
```

```
COMMENT ON TABLE llamadas
      IS 'Created from Entity LLAMADA by CASE on 17-MAY-94';

REM
REM    Created from Entity ORDEN by CASE on 17-MAY-94 REM
PROMPT
PROMPT Creating Table ORDENES
CREATE TABLE ordenes(
 ord_número_de_orden       NUMBER(3,0)      NOT NULL,
 ord_costo_del_estudio     NUMBER(3,0)      NOT NULL,
 ord_fecha_de_entrega      DATE             NOT NULL,
 ord_emp_identificador_de_la_em NUMBER(3,0)                    NOT NULL,
 ord_tes_clave_del_estado  CHAR(1)          NOT NULL,
 ord_fecha_de_peticion     DATE             NOT NULL
)
;

COMMENT ON TABLE ordenes
      IS 'Created from Entity ORDEN by CASE on 17-MAY-94';

REM
REM    Created from Entity REFERENCIA by CASE on 17-MAY-94
REM
PROMPT
PROMPT Creating Table REFERENCIAS
CREATE TABLE referencias(
 ref_com_identificador_de_la_em  NUMBER(4,0)                   NOT NULL,
 ref_est_clave_del_estudio  CHAR(8)          NOT NULL,
 ref_comentario                              CHAR(30)          NULL
)
;

COMMENT ON TABLE referencias
      IS 'Created from Entity REFERENCIA by CASE on 17-MAY-94';

REM
REM    Created from Entity TIPO DE ACCION by CASE on 17-MAY-94
REM
PROMPT
PROMPT Creating Table TIPOS_DE_ACIONES
CREATE TABLE tipos_de_aciones(
 tac_clave_de_la_accion    NUMBER(3,0)      NOT NULL,
 tac_descripcion                            CHAR(15)
     NOT NULL
)
```

```
;

COMMENT ON TABLE tipos_de_aciones
      IS 'Created from Entity TIPO DE ACCION by CASE on 17-MAY-94';

REM
REM     Created from Entity TIPO DE ESTADO by CASE on 17-MAY-94 REM
PROMPT
PROMPT Creating Table TIPOS_DE_ESTADOS
CREATE TABLE tipos_de_estados(
 tes_clave_del_estado        CHAR(1)              NOT NULL,
 tes_descripcion                                  CHAR(10)              NOT
NULL
)
;

COMMENT ON TABLE tipos_de_estados
      IS 'Created from Entity TIPO DE ESTADO by CASE on 17-MAY-94';

PROMPT Adding PRIMARY Constraint To ACCIONES Table

ALTER TABLE ACCIONES ADD (
        PRIMARY KEY (ACC_NÚMERO_DE_RECIBO,
                        ACC_EST_CLAVE_DEL_ESTUDIO,
                        ACC_TAC_CLAVE_DE_LA_ACCION)
        CONSTRAINT ACC_PK
)
/

PROMPT Adding PRIMARY Constraint To COMPANIAS Table

ALTER TABLE COMPANIAS ADD (
  PRIMARY KEY (COM_IDENTIFICADOR_DE_LA_EMPRES)
        CONSTRAINT COM_PK
)
/

PROMPT Adding PRIMARY Constraint To DIVISIONES Table

ALTER TABLE DIVISIONES ADD (
        PRIMARY KEY
                        (DIV_IDENTIFICADOR_DE_LA_DI
                        VISI,
                        DIV_COM_IDENTIFICADOR_DE_L
                        A_EM)
        CONSTRAINT DIV_PK
```

```
)
/

PROMPT Adding PRIMARY Constraint To EMPRESAS Table

ALTER TABLE EMPRESAS ADD (
  PRIMARY KEY (EMP_IDENTIFICADOR_DE_LA_EMPRES)
        CONSTRAINT EMP_PK
)
/

PROMPT Adding PRIMARY Constraint To ESTUDIOS Table

ALTER TABLE ESTUDIOS ADD (
        PRIMARY KEY (EST_CLAVE_DEL_ESTUDIO)
        CONSTRAINT EST_PK
)
/

PROMPT Adding PRIMARY Constraint To FACTURAS Table

ALTER TABLE FACTURAS ADD (
        PRIMARY KEY (FAC_NÚMERO_DE_FACTURA)
        CONSTRAINT FAC_PK
)
/

PROMPT Adding PRIMARY Constraint To INVESTIGADORES
Table

ALTER TABLE INVESTIGADORES ADD (
   PRIMARY KEY (INV_IDENTIFICADOR_DEL_INVESTIG)
        CONSTRAINT INV_PK
)
/

PROMPT Adding PRIMARY Constraint To INVESTIGADOS
Table

ALTER TABLE INVESTIGADOS ADD (
        PRIMARY KEY (IDO_NOMBRE,
                        IDO_ORD_NÚMERO_DE_O
                        RDEN)
        CONSTRAINT IDO_PK
)
/
```

```
PROMPT Adding PRIMARY Constraint To LLAMADAS
Table

ALTER TABLE LLAMADAS ADD (
        PRIMARY KEY (PKEY_IDEFAULT)
        CONSTRAINT LLA_PK
)
/

PROMPT Adding PRIMARY Constraint To ORDENES Table

ALTER TABLE ORDENES ADD (
        PRIMARY KEY (ORD_NÚMERO_DE_ORDEN)
        CONSTRAINT ORD_PK
)
/

PROMPT Adding PRIMARY Constraint To REFERENCIAS Table

ALTER TABLE REFERENCIAS ADD (
        PRIMARY KEY
                        (REF_COM_IDENTIFICADOR_DE_LA_
                        EM,
                        REF_EST_CLAVE_DEL_ESTUDIO)
        CONSTRAINT REF_PK
)
/

PROMPT Adding PRIMARY Constraint To TIPOS_DE_ACIONES
Table

ALTER TABLE TIPOS_DE_ACIONES ADD (
        PRIMARY KEY (TAC_CLAVE_DE_LA_ACCION)
        CONSTRAINT TAC_PK
)
/

PROMPT Adding PRIMARY Constraint To TIPOS_DE_ESTADOS
Table

ALTER TABLE TIPOS_DE_ESTADOS ADD (
    PRIMARY KEY (TES_CLAVE_DEL_ESTADO)
        CONSTRAINT TES_PK
)
/
```

```
PROMPT Adding FOREIGN Constraint To ACCIONES
Table

ALTER TABLE ACCIONES ADD (
 FOREIGN KEY (ACC_EST_CLAVE_DEL_ESTUDIO)
        REFERENCES  ESTUDIOS (
                        EST_CLAVE_DEL_ESTUDI
                        O)
        CONSTRAINT ACC_PERTENECER
)
/

PROMPT Adding FOREIGN Constraint To ACCIONES
Table

ALTER TABLE ACCIONES ADD (
FOREIGN KEY (ACC_TAC_CLAVE_DE_LA_ACCION)
        REFERENCES  TIPOS_DE_ACIONES (
                        TAC_CLAVE_DE_LA_ACCI
                        ON)
        CONSTRAINT ACC_TENER
)
/

PROMPT Adding FOREIGN Constraint To DIVISIONES
Table

ALTER TABLE DIVISIONES ADD (
        FOREIGN KEY
        (DIV_COM_IDENTIFICADOR_DE_LA_EM)
        REFERENCES  COMPANIAS (
                        COM_IDENTIFICADOR_DE_L
                        A_EMPRES)
        CONSTRAINT DIV_PERTENECER
)
/

PROMPT Adding FOREIGN Constraint To ESTUDIOS
Table

ALTER TABLE ESTUDIOS ADD (
FOREIGN KEY (EST_FAC_NÚMERO_DE_FACTURA)
        REFERENCES  FACTURAS (
                        FAC_NÚMERO_DE_FACTU
                        RA)
        CONSTRAINT EST_FACTURADO_CON
```

```
)
/

PROMPT Adding FOREIGN Constraint To ESTUDIOS
Table

ALTER TABLE ESTUDIOS ADD (
        FOREIGN KEY (EST_IDO_NOMBRE)
        REFERENCES  INVESTIGADOS (
                        IDO_NOMBRE)
        CONSTRAINT EST_REALIZADO
)
/

PROMPT Adding FOREIGN Constraint To ESTUDIOS Table

ALTER TABLE ESTUDIOS ADD (
  FOREIGN KEY (EST_INV_IDENTIFICADOR_DEL_INVE)
        REFERENCES  INVESTIGADORES (
                        INV_IDENTIFICADOR_DEL_INV
                        ESTIG)
        CONSTRAINT EST_HECHO_POR
)
/

PROMPT Adding FOREIGN Constraint To INVESTIGADOS
Table

ALTER TABLE INVESTIGADOS ADD (
    FOREIGN KEY (IDO_EST_CLAVE_DEL_ESTUDIO)
        REFERENCES  ESTUDIOS (
                        EST_CLAVE_DEL_ESTUDIO)
        CONSTRAINT IDO_AMPARAR
)
/

PROMPT Adding FOREIGN Constraint To INVESTIGADOS
Table

ALTER TABLE INVESTIGADOS ADD (
        FOREIGN KEY (IDO_ORD_NÚMERO_DE_ORDEN)
        REFERENCES  ORDENES (

                        ORD_NÚMERO_DE_ORDEN)
        CONSTRAINT IDO_INVESTIGADO_A_PE
)
```

```
/

PROMPT Adding FOREIGN Constraint To LLAMADAS
Table

ALTER TABLE LLAMADAS ADD (
 FOREIGN KEY (LLA_ACC_NÚMERO_DE_RECIBO)
        REFERENCES  ACCIONES (
                        ACC_NÚMERO_DE_RECIB
                        O)
        CONSTRAINT LLA_HECHA_POR
)
/

PROMPT Adding FOREIGN Constraint To ORDENES
Table

ALTER TABLE ORDENES ADD (
        FOREIGN KEY
        (ORD_EMP_IDENTIFICADOR_DE_LA_EM)
        REFERENCES  EMPRESAS (
                        EMP_IDENTIFICADOR_DE_LA_
        EMPRES) CONSTRAINT ORD_GENERADA_POR
)
/

PROMPT Adding FOREIGN Constraint To ORDENES Table

ALTER TABLE ORDENES ADD (
    FOREIGN KEY (ORD_TES_CLAVE_DEL_ESTADO)
        REFERENCES  TIPOS_DE_ESTADOS (
                        TES_CLAVE_DEL_ESTADO)
        CONSTRAINT ORD_CONTENER
)
/

PROMPT Adding FOREIGN Constraint To REFERENCIAS
Table

ALTER TABLE REFERENCIAS ADD (
        FOREIGN KEY
        (REF_COM_IDENTIFICADOR_DE_LA_EM)
        REFERENCES  COMPANIAS (
                        COM_IDENTIFICADOR_DE_LA
                        _EMPRES)
        CONSTRAINT REF_TENER
```

```
)
/

PROMPT Adding FOREIGN Constraint To REFERENCIAS
Table

ALTER TABLE REFERENCIAS ADD (
    FOREIGN KEY (REF_EST_CLAVE_DEL_ESTUDIO)
        REFERENCES  ESTUDIOS (
                        EST_CLAVE_D
        EL_ESTUDIO) CONSTRAINT
        REF_PARTE
)
/

REM
REM  End of command file
REM
EXIT
```

Programa que crea una base de datos para control de estudios. Esta información se adaptó al manejador 4GL.

```
MAIN
CREATE DATABASE estudios

CREATE TABLE acciones
(
ace_num_recibo          SMALLINT     NOT NULL,
ace_est_ord_num         SMALLINT     NOT NULL,
ace_tac_clave_ace       CHAR(1)      NOT NULL,
ace_fecha_entrega       DATE         NOT NULL,
ace_num_llamadas        SMALLINT
)

CREATE TABLE companias
(
com_nom                 CHAR(35)     NOT NULL,
com_id                  SMALLINT     NOT NULL,
com_direccion           CHAR(30)     NOT NULL,
com_colonia             CHAR(30)     NOT NULL,
com_informador          CHAR(35)     NOT NULL,
com_tel1                CHAR(10)     NOT NULL,
com_tel2                CHAR(10),
```

```
com_ext_r_h          CHAR(4),
com_Fax              CHAR(10),
com_comentario       CHAR(45),
com_ubic_r_h         CHAR(30),
com_fecha_ingreso    DATE        NOT NULL
)

CREATE TABLE divisiones
(
div_id               SMALLINT    NOT NULL,
div_com_id           SMALLINT    NOT NULL,
div_nom              CHAR(20)    NOT NULL
)


CREATE TABLE empresas
(
emp_nom              CHAR(35)    NOT NULL,
emp_id               SMALLINT    NOT NULL,
emp_clave            CHAR(4)     NOT NULL,
emp_rfc              CHAR(15),
emp_direccion        CHAR(30)    NOT NULL,
emp_colonia          CHAR(30)    NOT NULL,
emp_nom_sol          CHAR(35)    NOT NULL,
emp_tel1             CHAR(7)     NOT NULL,
emp_tel2             CHAR(7),
emp_ext_r_h          CHAR(4),
emp_ingreso          DATE        NOT NULL
)

CREATE TABLE estudios
(
est_fecha_captura    DATE        NOT NULL,
est_fecha_visita     DATE        NOT NULL,
est_fecha_entrega    DATE        NOT NULL,
est_ido_nombre       CHAR(35)    NOT NULL,
est_inv_id_invest    SMALLINT    NOT NULL,
est_ord_num          SMALLINT    NOT NULL,
est_fac_num          SMALLINT    NOT NULL,
est_num_disco        CHAR(10)    NOT NULL
)

CREATE TABLE facturas
(
fac_num              SMALLINT    NOT NULL,
```

```
fac_emp_nom          CHAR(35)      NOT NULL,
fac_fecha_emision    DATE          NOT NULL,
fac_Monto            MONEY(7,2)    NOT NULL
)


CREATE TABLE investigadores
(
inv_id_invest        SMALLINT      NOT NULL,
inv_colonia          CHAR(30)      NOT NULL,
inv_direccion        CHAR(30)      NOT NULL,
inv_nom_invest       CHAR(35)      NOT NULL,
inv_horario_libre    CHAR(10),
inv_telefono         CHAR(7)       NOT NULL,
inv_profesion        CHAR(15),
inv_comentario       CHAR(30)
)

CREATE TABLE investigados
(
ido_nom              CHAR(35)      NOT NULL,
ido_ord_num          SMALLINT      NOT NULL,
ido_colonia          CHAR(30)      NOT NULL,
ido_direccion        CHAR(30)      NOT NULL,
ido_ciudad           CHAR(20)      NOT NULL,
ido_tel_part         CHAR(10),
ido_puesto           CHAR(25),
ido_estado_rep       CHAR(15),
ido_ext_of           CHAR(4),
ido_municipio        CHAR(20)
)

CREATE TABLE llamadas
(
 lla_emp_llamada          CHAR(35)      NOT NULL,
 lla_estado_rep           CHAR(30)      NOT NULL,
 lla_tel                  CHAR(10)      NOT NULL,
 lla_fecha_llamada        DATE          NOT NULL,
 lla_acc_num_recibo       SMALLINT      NOT NULL,
 lla_est_ord_num          SMALLINT      NOT NULL,
 lla_tac_clave_acc        CHAR(1)       NOT NULL
)
```

```
CREATE TABLE ordenes
(
ord_num              SMALLINT       NOT NULL,
ord_fecha_entrega      DATE         NOT NULL,
ord_emp_id           SMALLINT       NOT NULL,
ord_tes_tipo         CHAR(1)      NOT NULL,
ord_fecha_peticion     DATE         NOT NULL,
ord_tes_dias         SMALLINT       NOT NULL,
ord_comentario       CHAR(20)
)

CREATE TABLE referencias
(
ref_com_id           SMALLINT       NOT NULL,
ref_ord_num          SMALLINT       NOT NULL,
ref_comentario       CHAR(30)
)

CREATE TABLE tipos_de_aciones
(
tac_clave_acc        CHAR(1)        NOT NULL,
tac_descripcion      CHAR(15),
tac_costo_acc        MONEY(5,2)     NOT NULL
)

CREATE TABLE tipos_de_estudios
(
tes_tipo             CHAR(1)        NOT NULL,
tes_descripcion        CHAR(20),
tes_costo            MONEY(4,0)     NOT NULL
)

END MAIN
```

BIBLIOGRAFÍA

* Barker, Richard / Longman, Cliff
CASE Method, (ORACLE) Function and Process Modeling
Eddison-Wesley, 1992

* Date, C. J.
Introducción a los sistemas de bases de datos
Eddison-Wesley, 1990

* Fairley, Richard
Ingeniería de Software
Mc Graw-Hill, 1988

* James, R. Groff
Using SQL
Osborne Mc Graw-Hill, 1990

* ORACLE
CASE - ORACLE
Servicios Educacionales, 1992

* P.I.C. (Programas, Ingeniería y Computadoras)
THE INFORMIX - 4GL
Product Family Manual, v 4.1, 1991

* Pressman, Roger S.
Ingeniería de Software
Mc Graw-Hill, 1990

* P.T.R (Programación en Tiempo Real)
La Metodología Entidad Relación para el Diseño Lógico de Base de Datos.
Apuntes del curso.

* Sommerville, Ian
Ingeniería de Software
Addison-Wesley Iberoamericana, 1988

- Wiederhold, Gio
  Diseño de la Base de Datos
  Mc. Graw-Hill, 1985

- Wiliam S. Davis
  Herramientas CASE, Metodología Estructurada para desarrollo de un Sistema.
  Paraninfo; 1992

- William Bates / Andrés Fortino
  dBase III Plus en redes locales
  Mc Graw-Hill Interamericana, 1989

- Rubén del Valle P.
  Introducción a Informix - 4GL
  Servicios Educacionales, 1991