UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

Sistema para la Automatización del Manejo de Información de Oracle de México

T E S I S

QUE PARA OBTENER EL TITULO DE INGENIERO EN COMPUTACION

P R E S E N T A N

AMADOR CORONA AGUSTIN ALBERTO
HERNANDEZ REYES MARGARITA
MONTERRUBIO GARCIA ANGEL
RAMIREZ TREJO JORGE

Director de Tesis:
M. I. LAURO SANTIAGO C.

MEXICO, D. F. 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

I. INTRODUCCION

Oracle Corporation liberó la primera base de datos relacional del mundo hace ya más de una década. Hoy en día, Oracle es un importante proveedor de soluciones integradas para la administración de datos. Los productos y servicios Oracle se venden en más de 85 países, lo que ha llevado a la organización a tener un crecimiento sin precedente en la industria informática.

En once de sus trece años de vida, Oracle ha tenido un crecimiento superior al cien por ciento; los resultados de 1989 hicieron que Oracle se convirtiera en el proveedor de *software* para bases de datos más grande del mundo y en la tercera empresa vendedora de *software* más grande del mundo.

Oracle Corporation estableció en el mercado mexicano una operación directa a través de una subsidiaria, Oracle de México.

Oracle de México es una empresa dedicada a la distribución y venta de *software* y servicios para bases de datos relacionales.

A tres años de haber iniciado operaciones en nuestro país, Oracle se ha definido como un líder en el mercado mexicano al contar con más de 100 instalaciones en minicomputadoras y *mainframes*, sin contar con la gran cantidad de *software* instalado en computadoras personales.

Oracle de México ofrece servicios tales como:

Centro Educacional. Reconocida y autorizada por Oracle Corporation, el área educacional ofrece una amplia variedad de cursos.

Soporte Técnico. Creada para apoyar a los clientes al ofrecer servicios de apoyo técnico, solución de problemas, creación de prototipos, etc.

Consultoría. Ofrece el servicio de personal experimentado en el uso de productos Oracle y en la implementación de sistemas y soluciones en general, apoyándose en tecnologías propias como *Oracle Case* y *Case*Method*.

Oracle se ha difundido rápidamente por todas las áreas comerciales, académicas y de investigación de México, en un mercado sumamente amplio.

A continuación se detalla brevemente cada una de las áreas de las que está compuesto actualmente Oracle de México *(ver figura I.1)*.

DIRECCION DE FINANZAS Y ADMINISTRACION

La función de esta dirección es la de realizar toda la operación administrativa de la empresa, tal como Administración de los contratos de compra-venta de licencias y servicios en general, Recursos Humanos, Cuentas por Cobrar, Cuentas por Pagar, Contabilidad, Tesorería, Activos, etc.

DIRECCION COMERCIAL

Esta dirección está integrada por dos Gerencias Regionales, México D.F. y Monterrey. Cada Gerencia Regional cuenta con un equipo de vendedores cuya función es la venta y promoción de los productos y servicios Oracle.

DIRECCION TECNICA

El área técnica es la encargada de apoyar a los clientes en la solución de problemas, actualización de productos, creación de prototipos, soporte técnico pre-venta y post-venta, así como una amplia variedad de cursos impartidos por el área educacional.

DIRECCION DE CONSULTORIA

La Dirección de Consultoría se encarga del desarrollo de sistemas a través del uso de las herramientas Oracle de cuarta generación, así como de auxiliar al cliente en el uso óptimo de los productos Oracle.

Actualmente cada una de las diferentes áreas de Oracle de México maneja su información por separado, es decir, cada departamento se encarga del manejo de sus datos de diferente forma, por ejemplo la Dirección de Finanzas, que se encarga del manejo de los contratos de los clientes, lleva la relación de estos contratos en archiveros y folders, teniendo que hacer manualmente todo el proceso de búsqueda y verificación de los mismos cada vez que alguna de las otras direcciones solicita algún dato contenido en dichos contratos.

ORGANIGRAMA DE LA EMPRESA ORACLE DE MEXICO

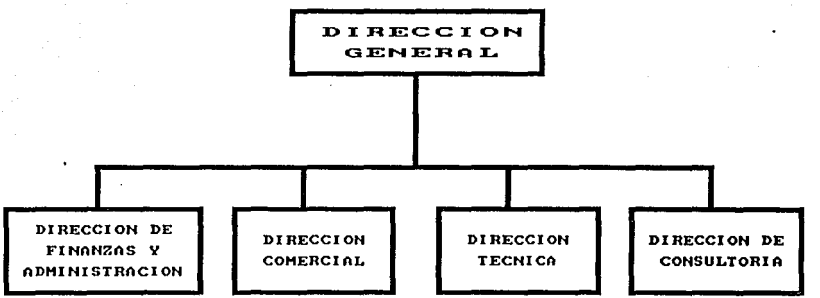


FIGURA: I.1

Así como la dirección de Finanzas lleva el control de su propia información, las otras direcciones organizan de manera diferente parte de esa misma información, haciendo que ésta sea redundante e inconsistente.

Por otra parte, el área de Dirección Técnica necesita llevar un control de las requisiciones de soporte técnico por parte de los clientes, con el fin de darles un seguimiento eficiente para satisfacerlas. Actualmente se reciben un promedio de 200 requisiciones al mes, las cuales deben registrarse para consultas posteriores en requisiciones similares. Se deben obtener diversas estadísticas que reflejen los movimientos del área técnica.

La Dirección Técnica también es la encargada de la administración de los cursos impartidos por el área Educacional, la cual necesita conocer información referente a la cantidad de cursos impartidos a los clientes y al personal en general.

Con el fin de solucionar los problemas antes mencionados, el objetivo del presente trabajo es desarrollar un Sistema de Información Automatizado que permita mantener la información que manejan las diferentes Direcciones de la empresa Oracle de México de una manera compartida, consistente, no redundante y que cuente con políticas de seguridad e *interfaces* que faciliten el acceso a la información.

Para cumplir el objetivo de este trabajo se lleva a cabo una investigación a fondo de las necesidades de cada una de las diferentes direcciones y se hace un estudio completo tanto de las metodologías de análisis y herramientas *CASE* de Oracle y en general de sus productos de desarrollo, así como de sus similares en el mercado.

Se emplean para el desarrollo del sistema las herramientas y metodología *CASE* de Oracle y los recursos propios de la empresa.

Otro de los objetivos que debe cumplir el sistema a desarrollar será el de servir como presentación a clientes de la misma empresa, por lo cual se deberá usar en su desarrollo las versiones más recientes, tanto de la herramienta *CASE* como de las herramientas de desarrollo Oracle, con las que se cuente al momento de implementar el sistema.

Este trabajo beneficiará a la empresa en los siguientes aspectos :

- Se tendrá una sola base datos con toda la información común a las diferentes áreas, evitando la redundancia e inconsistencia en los datos y facilitará el control y manejo de la misma.
- Se empleará como una presentación del uso de los productos Oracle en el desarrollo de sistemas mostrando la aplicación de diversos productos en cada fase.
- Permitirá ofrecer un mejor servicio a clientes, dado que se tendrá un control automatizado de la información.

En el presente trabajo se desarrollan los siguientes temas:

CAPITULO 1: METODOLOGIAS DE DESARROLLO DE SISTEMAS Y ETAPA DE ANALISIS

En este capítulo se presenta un panorama general de las estrategias de desarrollo e implementación de sistemas, se lleva a cabo un estudio profundo de la metodología CASE de desarrollo y se realiza un estudio comparativo entre las metodologías de desarrollo estudiadas y la metodología CASE.

Se cubre la fase de estrategia de sistemas de acuerdo a la metodología CASE antes mecionada. En la fase de estrategia se realizan encuestas a las áreas de la compañía involucradas en el sistema que se implantará, determinando las necesidades de información de cada una de ellas. A partir de las necesidades encontradas, se generan los diagramas funcionales de cada área y el modelo entidad-relación que representa las relaciones de información existentes en todo el sistema, cubriéndose así la fase de análisis.

CAPITULO 2: HERRAMIENTAS CASE DE DESARROLLO Y ETAPA DE DISEÑO

Se presenta un panorama general de las herramientas CASE de desarrollo de los fabricantes más conocidos y se realiza un análisis comparativo entre estas herramientas CASE y las herramientas CASE de Oracle, las cuales están basadas cien por ciento en la metodología CASE*Method.

Se emplean las herramientas CASE Oracle para generar el diseño lógico de la aplicación. Se presentan los resultados obtenidos despues de llevar a cabo los pasos necesarios para cubrir al 100% la fase de diseño de acuerdo a la herramienta empleada.

CAPITULO 3: HERRAMIENTAS CUARTA GENERACION Y ETAPA DE IMPLEMENTACION

En este capítulo se lleva a cabo un estudio de las herramientas de cuarta generación disponibles para la implantación del sistema, se describen las características de las herramientas más viables para el cumplimiento de los objetivos y se realiza un estudio comparativo entre dichas herramientas.

Se realiza un estudio de los ambientes de desarrollo en los cuales podría llevarse a cabo la implementación del sistema.

Se genera la estructura física de la base de datos y los elementos necesarios para empezar a contruir la aplicación y describen los elementos que conforman dicha base de datos.

Haciendo uso de las herramientas de desarrollo Oracle más recientes se construyen los módulos que integran la aplicación.

CAPITULO 4: ETAPA DE TRANSICION

Se probará que el sistema implementado funciona correctamente y se realizará una evaluación del producto final. Se generarán los manuales de usuario necesarios para la operación del sistema y se elaborá un plan de capacitación de usuarios finales.

CAPITULO 5: ETAPA DE PRODUCCION

El sistema implementado es puesto en producción y se determinan las estrategias a seguir para su eventual mantenimiento, se generan especificaciones técnicas del sistema.

CAPITULO 6: CONCLUSIONES

En esta parte se describen los resultados obtenidos en la implantación de sistema y las ventajas o desventajas que se presentaron con las metodologías y herramientas seleccionadas para el desarrollo del sistema. Se analiza hasta qué punto se cumplieron los objetivos propuestos.

1. METODOLOGIAS DE DESARROLLO DE SISTEMAS Y ETAPA DE ANALISIS

METODOLOGIAS DE DESARROLLO DE SISTEMAS Y ETAPA DE ANALISIS

Este capítulo se puede dividir en dos partes principales: una en la cual se dan los antecedentes teóricos que nos servirán de base para la implementación del sistema, y otra donde se tratan los aspectos propios del desarrollo del sistema.

Así, en este capítulo se presentan los siguientes temas:

1.1 Estrategias de desarrollo e implementación de sistemas

1.2 Requerimientos de las diferentes partes de la compañía

1.3 Resultados obtenidos en este capítulo

El primer tema es la parte de investigación de este capítulo. En este tema se explica lo relativo a los métodos empleados en el análisis y desarrollo de *software*. Se describen las características de algunos de los métodos existentes.

De acuerdo con los objetivos de este trabajo de tesis emplearemos para el desarrollo del sistema la metodología CASE, no obstante se llevará a cabo el estudio comparativo entre las diversas metodologías estudiadas.

Los temas 1.2 y 1.3 forman la segunda parte del capítulo. En el tema 1.2 se detallan los requerimientos de información y procedimientos que se siguen en cada una de las diferentes áreas de la compañía, con esto se cumple la primera fase de la metodología CASE. Así mismo, se especifica cuáles de estos procedimientos se automatizarán y pasarán a formar parte del sistema.

Finalmente, en el tema 1.3 se muestran los datos que se obtienen como resultado de la utilización de la metodología seleccionada y la información adquirida en el tema 1.2. Estos datos serán utilizados como entrada a las siguientes etapas del desarrollo del sistema.

1.1 ESTRATEGIAS DE DESARROLLO E IMPLEMENTACION DE SISTEMAS

La ingeniería de *software* es un conjunto de actividades cuyo objetivo es el desarrollo de *software* más confiable, menos costoso, y que funcione eficientemente sobre máquinas reales. Abarca un conjunto de tres elementos:

- métodos
- herramientas
- procedimientos

Los métodos abarcan la planificación y estimación de proyectos, análisis de los requerimientos del sistema y del *software*, diseño de las estructuras de datos, arquitectura de programas y procedimientos algorítmicos, prueba y mantenimiento.

Las herramientas de la ingeniería de *software* suministran un soporte automático o semiautomático para los métodos. Hoy en día, existen herramientas para soportar cada uno de los diversos métodos.

Los procedimientos unen a los métodos y herramientas y facilitan un desarrollo racional y oportuno del *software* de computadora. Definen la secuencia en la que se aplican los métodos, los controles que ayudan a asegurar la calidad y coordinación de los cambios en el *software*.

Las herramientas serán analizadas a detalle en el capítulo siguiente, por otra parte dado que los procedimientos vienen inherentes tanto a la metodología como a las herramientas electas, estos no serán desarrollados con tanta profundidad.

METODOS ESTRUCTURADOS

Los métodos estructurados consisten de un conjunto de reglas, tareas, y técnicas, para ayudar en el desarrollo de sistemas en uno o más de sus estados (estrategia, análisis, diseño, etc).

Los métodos estructurados empezaron a ser utilizados en círculos académicos a finales de los 60's, y alcanzaron popularidad en el ámbito comercial a mediados de los 70's. Estos métodos ofrecieron una solución a los entonces existentes estándares de desarrollo no estructurados y carentes de diseño.

Dentro de los métodos estructurados se encontraban las gráficas estructuradas de Yourdon, diagramas de Warnier/Orr, diagramas de Jackson, la técnica estructurada de Chen para el diseño estructurado de datos, diagramas de entidad relación, etc.

A mediados de los 70's, James Martin dio inicio al concepto de ciclo de vida para unificar y controlar todo el proceso de desarrollo de un sistema. Este método llamado Ingeniería de Información (IE) en el que se sigue un enfoque *top-down*, se identifican las necesidades de información durante la etapa de análisis con el fin de delimitar los alcances del sistema. Las metodologías *CASE (Computer Aided System Engineering)* fueron construidas siguiendo este enfoque.

Las Herramientas *CASE* empezaron a ser desarrolladas a principio de los 80's para soportar y automatizar algunas de las tareas encontradas en los métodos estructurados de desarrollo.

Algunos de los métodos más conocidos para el desarrollo de sistemas dentro de la ingeniaría de *software* se detallan a continuación.

Método Warnier/Orr

Warnier/Orr no es propiamente una metodología para la construcción de sistemas, existen los diagramas Warnier/Orr , el método de Warnier (lógica de estructura de datos, construcción de sistemas lógica, y construcción lógica de programas) y el método de Orr (Desarrollo de sistemas estructurados) están unidos pero estrictamente hablando no existe una metodología Warnier/Orr.

Es muy común que se confunda la metodología con los diagramas, tal vez esto sea porque los diagramas son la parte más visible de la mayoría de las metodologías; sin embargo, una metodología es mucho más que sólo diagramas y reglas de sintaxis.

Dentro del contexto de ingeniería de *software* un método es un procedimiento o una técnica que realiza una porción significante del ciclo de vida de un sistema. Una metodología es una colección de métodos basados en una filosofía común que juntos cubren parte o todo el ciclo de vida de un sistema.

El nombre correcto de lo que mucha gente llama metodología Warnier/Orr es: Desarrollo Estructurado de Sistemas de Datos (DSSD Data-Structured Systems Development) y es el resultado del trabajo conjunto de mucha gente.

En 1972, el artículo de Terry Baker "Chief Programmer Team Operations" en el semanario de sistemas IBM, reunió varias ideas de programación estructurada, diseño *Top-Down* e implementación, esto tuvo un gran impacto en el campo de la computación. Fue entonces cuando empezó la revolución estructurada en los Estados Unidos.

A principios de los 70's Ken Orr descubre que la estructura jerárquica de los programas es reflejada en la estructura jerárquica de los datos y fue por ese tiempo que Jean Warnier había hecho el mismo descubrimiento, pero además había construido una metodología sistemática alrededor de este descubrimiento.

La programación estructurada de los datos significa que pueden ser construidas soluciones predeciblemente correctas para una amplia gama de problemas de programación. La técnica de datos estructurados fue extendida para resolver programas complejos y arbitrarios.

Se desarrolló un esquema en el cual las entradas físicas son convertidas a entradas lógicas; las entradas lógicas se convierten en salidas lógicas; finalmente, las salidas lógicas son convertidas en salidas físicas.

Con este proceso de diseño de programas comienza la estrategia del diseño orientado a objetivos, éste inicia con la estructura de salida y trabaja hacia atrás, inicia con la salida lógica, entrada lógica y finalmente entrada física.

Poco a poco DSSD se movió de la metodología de diseño de programas a una metodología de diseño de sistemas. Después la metodología fue expandida para abarcar el diseño de bases de datos, definición de requerimientos y finalmente la planeación y arquitectura de los sistemas.

A nivel conceptual DSSD aún contiene rasgos característicos a nivel de programación. Por ejemplo, aún se enfoca (en su fase de diseño) a trabajos hacia atrás desde las salidas. DSSD trabaja hacia atrás hasta la base datos lógica y luego hacia las entradas. La base de datos lógica se transforma en una base de datos relacional y normalizada *(ver figura 1.1.1)*.

Mientras que una definición completa de los requerimientos (salidas más algoritmos) es un excelente punto de partida en el diseño, no es propiamente el punto desde el cual empezar la definición de los requerimientos. Así a través de los años, DSSD se

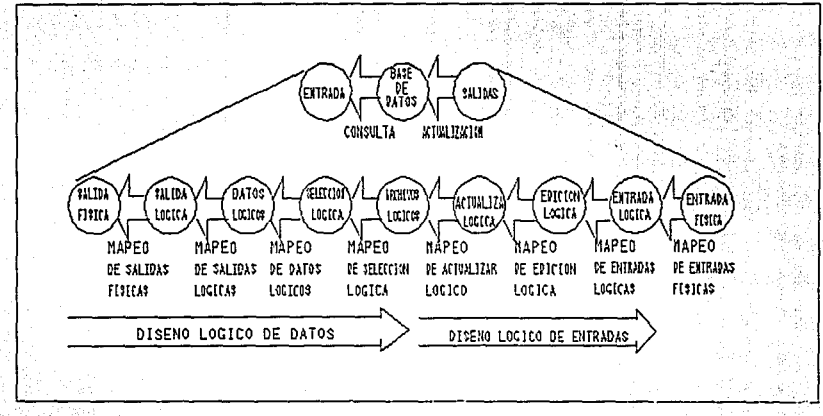


Figura 1.1.1 Trabajo hacia atrás desde la salidas del DSSD

ha extendido hasta cubrir primero el contexto, luego las funciones y finalmente los resultados del sistema.

Se hicieron necesarias un número de herramientas para facilitar este proceso. Los diagramas de Entidades ayudan a definir el contexto del sistema, y los diagramas de línea de montaje (una forma modificada de los diagramas Warnier/Orr) ayudan a definir el flujo funcional del sistema.

Las metodologías estructuradas de datos tienen ventaja sobre muchas metodologías orientadas a procesos, dado que son más rigurosas y por tanto proporcionan una mejor base para la verdadera integración a través de todo el ciclo de vida de un sistema. DSSD ha sido usado exitosamente en un gran rango de sistemas de *software*, desde sistemas comerciales en línea hasta sistemas de control de tiempo real. Cientos de personas lo han aprendido y cientos de sistemas han sido desarrollados con este método.

DSSD es un acercamiento a la ingeniería de *software* que ha proporcionado un marco de trabajo estable para incorporar nuevas tecnologías conforme éstas surgen. Por ejemplo, se han incorporado los prototipos de diseño en línea y en tiempo real sin sacrificar lo ya realizado.

Método Gane / Sarson

Este método está basado en el modelado lógico el cual consiste en tomar las ideas vagas necesarias acerca de los requerimientos y convertirlas en definiciones precisas tan pronto como sea posible, este proceso puede realizarse rápidamente si se cuenta con una técnica gráfica que nos permita tener la esencia del sistema sin ir al problema de la implementación física, con esto pudiera hacerse por ejemplo un prototipo.

El modelado lógico puede verse como un proceso de 7 pasos:

1.- Desarrollo de un Diagrama de Flujo de Datos (DFD) general que describa lo que ocurre en cada área de la empresa. Para simplificar los diagramas se emplean solamente cuatro símbolos que producen un diagrama mostrando la naturaleza lógica de cualquier sistema de información a cualquier nivel deseado de detalle. Los DFD nos muestran los límites de una área dentro del sistema y de la empresa misma, los diagramas no son técnicos, es decir son fáciles de entender por cualquier persona familiarizada con el área descrita, nos muestran datos almacenados y los procesos que transforman esos datos.

2.- Derivar una lista de los datos que serán almacenados en cada almacén de datos, la lista puede afinarse revisando los datos de entrada/salida al sistema y determinando que representa cada dato.

3.- Revisar lo que el análisis entidad-relación pueda decirme acerca de la estructura de los datos. Obtener las Entidades y crear un diagrama con las entidades que se han definido, después buscar las relaciones existentes entre las entidades y dibujar líneas que muestren estas relaciones.

Por cada par de entidades se tiene una relación entre sus elementos, la relación puede ser uno a uno, muchos a uno o muchos a muchos. Estos diagramas nos dan mucha información acerca del sistema, mostrándonos todas las relaciones que existen entre las entidades involucradas *(ver figura 1.1.2)*.

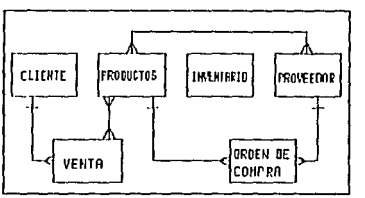


Figura 1.1.2 Todas las relaciones identificables entre entidades

4.- Emplear la información de los datos para describir un modelo que haga una liga entre dos tablas bidimensionales. Las tablas deberán estar normalizadas.

5.- Redibujar el DFD para reflejar una vista más precisa del sistema de datos como resultado del análisis del modelo entidad-relación y de la normalización.

6.- Particionar este modelo lógico de procesos y datos dentro de unidades de procedimientos, separar los procedimientos que puedan ser ejecutados manual o automáticamente.

7.- Especificar el detalle de cada unidad de procedimiento que se requiere para implementar el sistema: Extraer la unidad del DFD y mostrar donde queda dentro del sistema, detallar las tablas accesadas por esa unidad, mostrar un esquema de pantallas y reportes involucradas en la unidad y detallar la lógica y los procedimientos que serán implementados.

Una vez definidos los procedimientos se puede decidir si será un prototipo o se debe implementar directamente en algún lenguaje.

Los pasos 6 y 7 no son estrictamente hablando un modelado lógico dado que tratan de convertir el modelo lógico en modelo físico, sin embargo son parte del flujo natural a través del proceso que comienza en la definición de sistema y termina con el diseño físico.

Método Entidad-Relación

Es una metodología estructurada que puede sistemáticamente convertir los requerimientos del usuario en una base de datos bien diseñada. No es propiamente una metodología sino una aproximación.

Este método se basa en lo siguiente:

- Desarrollar un diagrama entidad-relación (ERD Entity Relational Diagram). Este paso identifica Entidades , Relaciones y atributos asociados además de la llave primaria de cada tipo de entidad.

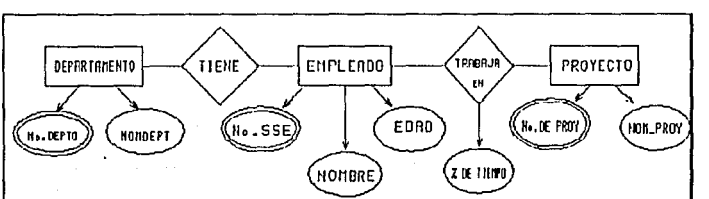


Figura 1.1.3
Diagrama Entidad Relación

Una entidad es una cosa, un concepto, una organización o un evento de interés para la organización que se está modelando. Una Entidad Tipo es una clasificación de entidades que satisfacen ciertos criterios. Una relación es una interacción entre entidades. Una relación tipo es una clasificación de relaciones basada en cierto criterio. Usualmente los sustantivos corresponden a las entidades mientras que los verbos corresponden a las relaciones.

El siguiente paso es identificar la cardinalidad de los tipos de relaciones, es decir con cuantos elementos de una entidad B se puede relacionar un elemento de una entidad A y viceversa. Se identifican después las propiedades (atributos) de cada Entidad-Relación y se expresaran gráficamente con círculos (o elipses). Las llaves primarias son identificadas con un doble círculo.

- Convertir los diagramas entidad-relación en archivos convencionales y estructuras de bases de datos. Existe un conjunto de reglas para hacer esto.

- Desarrollar los programas de aplicación basados en los archivos y estructuras de la base de datos. Si se está usando un Sistema Manejador de Bases de Datos (DBMS) es posible escribir en este momento un programa en SQL (System Query Language) para expresar la consulta a los datos.

El modelado Entidad-Relación puede ser usado no sólo como una herramienta de diseño sino como la base para el modelado en sistemas manejadores de bases de datos.

Método Yourdon

El método Yourdon es una colección genérica de ideas de ingeniería de *software* desarrolladas por muchas personas quienes han trabajado para la compañía Yourdon. Las ideas de todas estas personas se reunieron y se les dió el nombre de técnicas estructuradas: programación estructurada, diseño estructurado, y análisis estructurado, con esto Ed Yourdon intenta enlazar su metodología a las demás etapas del proceso de desarrollo de un sistema, para crear una metodología que compita con los métodos que sí cubren completamente el ciclo de vida de un sistema, es decir, los métodos que inician con la etapa de estrategia y análisis, diseño lógico y físico hasta la implementación, transición y producción.

Debido a la continua influencia de nuevas ideas de gente nueva, el método Yourdon está en continuo desarrollo, de tal forma que actualmente el método Yourdon es muy diferente del de hace 10 años, ahora se han incorporado las mejores ideas de diseño orientado a objetos y de análisis.

Este método consta de dos partes: herramientas y técnicas. Las herramientas son una variedad de diagramas gráficos usados para modelar los requerimientos y la arquitectura de un sistema de información. El más familiar de estos diagramas es el diagrama de flujo de datos (DFD), este diagrama ha sido extendido para soportar sistemas de tiempo-real. Los DFD incluyen control de flujo y control de procesos (*ver figura 1.1.4*).

Este método incluye además diagramas entidad-relación (ERD's) y diagramas de transición de estados (STD's State Transition Diagrams). Para terminar de definir los requerimientos de un sistema pueden emplearse las cartas de estructura para mostrar la organización de los módulos que serán implementados.

Para complementar la descripción del sistema es necesario un soporte textual adicional, un diccionario de datos que nos describa cómo está compuesto cada elemento y un conjunto de especificaciones de procesos que nos describa el comportamiento de los procesos en cada nivel.

La técnica del método Yourdon consta de guías que nos llevan de la nada a un modelo del sistema bien organizado. Esta técnica consiste en particionar (*top-down*) el sistema en funciones. Actualmente el método usa una técnica conocida como partición de eventos, inicia dibujando un diagrama de contexto el cual permite definir los alcances del sistema y las interfaces con fuentes externas. Después de las entrevistas al usuario se escriben una serie de eventos que ocurren en el medio ambiente externo y a las cuales el sistema responde, esto nos permite dibujar un primer diagrama de flujo de datos. Por ejemplo, si un sistema consta de 100 eventos tendrá 100 burbujas en el DFD, dado lo anterior es posible particionar los eventos en varios y unirlos en un solo proceso, esta técnica es parecida al diseño orientado a objetos.

El Diseño Estructurado Yourdon consta de 4 pasos:

1.- Diagramas de Flujo de Datos

Muestra los sistemas como un conjunto de procesos que transforman datos. Los diagramas de flujo de datos nos muestran la

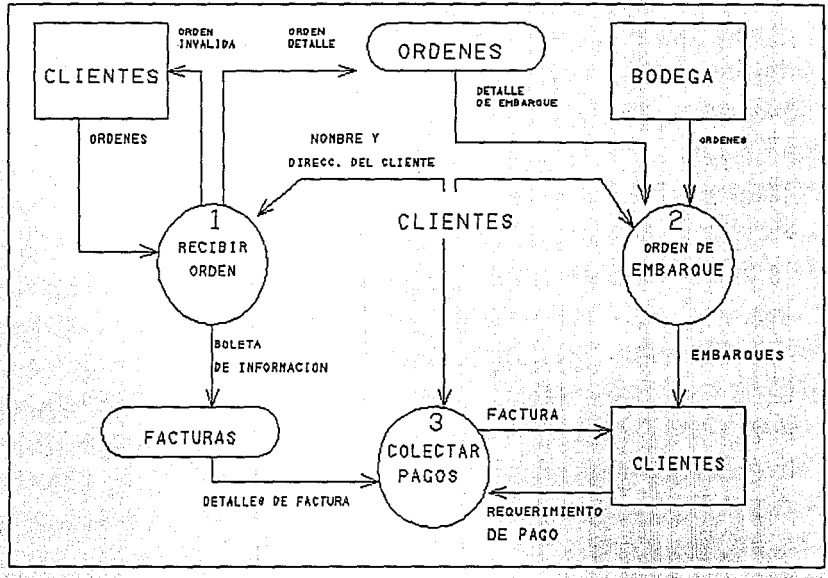


Figura 1.1.4 Diagrama de Flujo de Datos

relación entre los procesos y el flujo de información entre ellos. Los requerimientos en tiempo real son mostrados en los Diagramas de Flujo de datos vía el control de flujo y el control de procesos. Los comportamientos dependientes del tiempo son mostrados con diagramas de transición de estados y los datos involucrados son modelados de una forma simple en modelos de entidad.

2.- Cartas de Estructura

Las carta de estructura son una jerarquía de unidades procedurales que documentan el control del programa, el flujo de información entre programas y la lógica del programa. Se emplean las técnicas de Análisis de Transformación y de Análisis de Transacción.

3.- Evaluación del Diseño

En este paso se mide la calidad del diseño, para ello son empleados los criterios de acoplamiento y cohesión.

4.- Preparación para la Implementación

La lógica diseñada del programa es empaquetada en unidades de implementación física, llamadas unidades de carga.

Método de Ingeniería de Información (IEM)

El método de Ingeniería de Información sigue un enfoque *top-down*, desarrollando modelos en cada una de las etapas del ciclo de vida que cubre (estrategia, análisis y diseño), forzando el desarrollo de sistemas al manejo de metas y planes de estrategias.

IEM ofrece modelos de alto nivel para la representación total de una empresa durante la etapa de estrategia y diagramas de estructura de datos detallados (entidades, relaciones, atributos), descomposición funcional, flujo de datos y diagramas de dependencia, y definición de reportes y pantallas durante el "análisis de las necesidades de la empresa". Diagramas mostrando detalladamente la lógica del programa, incluyendo el manejo procedural y criterios de selección durante subsecuentes diseños y construcciones de sistemas. Matrices para el mapeo de las especificaciones del análisis y del diseño, para el manejo de metas, conjuntando los requerimientos de datos y procesos, y

asegurando una terminación e integración de éstos a nivel de aplicación.

Método DeMarco

DeMarco es un método de análisis estructurado orientado a procesos. Es utilizado para definir los requerimientos de un sistema en las etapas de análisis y diseño. Durante la etapa de análisis, DeMarco obtiene todo el conjunto de información manejada por los procesos actuales y los nuevos requerimientos de ésta y los utiliza para obtener la descripción de la funcionalidad del sistema a construir. Utiliza diagramas de flujo de datos como técnica de representación.

Pasos que sigue:

1.- Construir el modelo físico actual (DFDs).

2.- A partir del modelo físico, construir el modelo lógico actual (DFDs).

3.- Construir el modelo lógico del nuevo sistema a desarrollar incluyendo DFDs, definiciones del diccionario de datos, y especificaciones de los nuevos procesos.

4.- Crear una familia de nuevos modelos físicos (DFD).

5.- Obtener el costo y tiempos estimados para cada modelo.

6.- Seleccionar el modelo más adecuado.

Método Jackson

Jackson es una técnica de diseño orientada a datos, la cual se basa en las estructuras de datos requeridas para obtener la estructura de los procesos. Empieza asumiendo que la etapa de análisis ha sido terminada.

En este método el sistema es visto como un conjunto de procesos, los cuales se adecuan al manejo de las estructuras de datos, las cuales se consideran estáticas. Utiliza básicamente dos técnicas de diagramación: diagramas de redes, en los cuales se muestra el flujo de información a través de los procesos, y

diagramas arborescentes, en los cuales se muestran las relaciones jerárquicas entre procesos y datos.

Pasos que sigue.

1.- Dibujar los diagramas de estructura de árbol mostrando el flujo de datos entre cada proceso.

2.- Tomar todas las estructuras anteriores y formar un solo programa.

3.- Mostrar las operaciones necesarias para obtener la información de salida de la información de entrada y asignar cada una de éstas a los componentes del programa (procesos).

4.- Transcribir el programa a un texto estructurado, agregando tanto lógica de selección como ciclos.

CASE*Method

CASE*Method es una metodología que guía los proyectos hasta su terminación, a través de la aplicación de técnicas rigurosas para el desarrollo de sistemas. CASE*Method permite aprovechar los beneficios de la tecnología *CASE* dando un marco de trabajo estructurado para guiar y controlar el desarrollo productivo a través de todo el ciclo de vida.

Esta Metodología abarca el análisis estratégico, análisis detallado y diseño relacional de sistemas, permite llevar el control del sistema en todas las etapas de su ciclo de vida de una manera eficiente, ordenada y con la certeza de alcanzar los objetivos del sistema.

Durante las etapas de estrategia y análisis del ciclo de desarrollo del sistema, CASE*Method establece una comunicación más efectiva con los usuarios a través de técnicas, entrevistas y diagramación. Al entender los requerimientos de los usuarios desde el principio, se evitan revisiones costosas posteriores durante el desarrollo de sistemas.

Una vez identificados los requerimientos CASE*Method define el camino óptimo para completar estas tareas rápidamente y con precisión, basándose en la teoría relacional y extensos controles de calidad.

El método *CASE* es un método *top-down*, en el la gente está antes que los documentos, es un método de autochequeo, la implementacion es independiente de los requerimientos, está diseñado para el cambio, se implementa en forma metódica en todo tiempo.

CASE*Method soporta todas las etapas del ciclo de vida de un sistema.

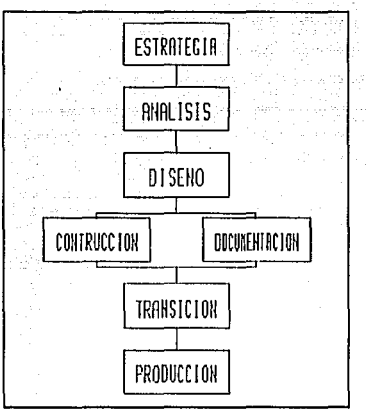


Figura 1.1.5
CASE*Method Enfoque estructurado
para el desarrollo de sistemas

CASE*Method comienza con el principio del ciclo de vida, involucrando a usuarios y administradores en la especificación de los objetivos, requerimientos y prioridades que se necesitan para desarrollar el sistema.

Mientras se preparan e incrementan modelos detallados, pueden verificarse y confirmarse con los usuarios, creando prototipos

conceptuales del sistema completo. Utilizando el enfoque *top-down*, combinado con técnicas *bottom-up* para el chequeo cruzado, CASE*Method incrementa la calidad en cada etapa.

Después de completar el análisis, se crea el diseño físico y lógico para un desempeño y flexibilidad óptimos. La documentación e implementación del sistema ocurren paralelamente, mientras se construyen los sistemas y se realizan pruebas de unidad. La transición incluye el entrenamiento del usuario, la conversión de datos, y las pruebas del sistema, basado en una planeación anterior durante el análisis y diseño.

Etapa Estratégica

- Establecer los objetivos, prioridades y restricciones del negocio.
- Determinar los Modelos de Negocio.
- Establecer el compromiso administrativo, ganado a través de involucrar usuarios clave y secciones interactivas con retroalimentación.
- Realizar el plan de desarrollo y arquitectura del sistema.

Etapa de Análisis

- Especificación de los modelos detallados de la información necesaria para el sistema, vía los diagramas Entidad-Relación.
- Crear los modelos de los requerimientos funcionales utilizando técnicas de descomposición.
- Realizar diagramas de flujo de datos y matrices de chequeo cruzado.
- Realizar el plan de transición incluyendo la conversión y el entrenamiento.
- Definir la frontera del sistema.

Etapa de Diseño

- Establecer la arquitectura detallada del sistema.

- Transferir las especificaciones conceptuales a diseños físicos y lógicos.

- Realizar el diseño relacional de bases de datos incluyendo consideraciones de rendimiento y tamaño.

- De las especificaciones funcionales, realizar el diseño de módulos para sistema de aplicación.

- Realizar el diseño para la transición y la arquitectura de pruebas.

- Utilizar las matrices que ilustran referencias cruzadas de los componentes del sistema, para control de cambios.

Etapa de Construcción

- Creación de la base de datos relacional.

- Implementación de los sistemas de aplicación.

- Pruebas de unidad y revisiones del usuario.

- Aseguramiento de la calidad y aceptación.

Etapa de Documentación

- Documentación del sistema.

- Material de entrenamiento, operaciones y guías del usuario.

Etapa de Transición

- Conversión de los datos.

- Ejecutar las pruebas de simulación paralelas.

- Ejecutar las pruebas del sistema para verificar la funcionalidad, rendimiento, utilidad, recuperación, integridad y seguridad.

- Aceptación del usuario.

- Convertir el sistema para que sea completamente operacional.

Etapa de producción

- Realizar las pruebas de integridad.
- Ejecutar los procedimientos de soporte y monitoreo del sistema.
- Registro de los requerimientos.
- Cambio en los procedimientos de control.
- Realizar revisiones regulares.

ANALISIS COMPARATIVO DE LAS METODOLOGIAS DE DESARROLLO

Método Warnier/Orr (DSSD)

DSSD es un método enfocado a datos el cual contempla únicamente la etapa de diseño. Deriva las estructuras del programa de las estructuras de los datos siguiendo una estrategia *bottom-up*, pone atención en las salidas requeridas para obtener las estructuras del programa y las estructuras de los datos de entrada.

Pasos que sigue:

1.- Fase de planeación del proyecto: un plan de proyecto es desarrollado, conjuntando las metas de la empresa con un sistema particular de objetivos en un nivel de diseño lógico.

2.- Fase de definición de requerimientos: se definen en un nivel lógico y después en uno físico los requerimientos de usuarios y sistema, además de las necesidades de *hardware*. Se desarrollan diagramas de entidades (Diagramas de Flujo de Datos) describiendo las entidades y las transacciones que pasan a través de ellas. Las salidas de los procesos son mostrados utilizando diagramas de Warnier-Orr y un diccionario de datos.

3.- Fase de diseño: Se hace un diseño lógico de la base de datos normalizada y un diseño lógico de procesos, pasando después al diseño físico.

Ventajas

- Ofrece a los diseñadores pasos sencillos a seguir, usando una sola técnica de diagramación desde el principio hasta el final.
- Warnier/Orr es bueno para sistemas pequeños orientados a reportes.

Desventajas

- DSSD utiliza las salidas del sistema para obtener los requerimientos del mismo. Esta estrategia *bottom-up* puede causar un manejo incompleto o redundante tanto de datos como de procesos.
- DSSD sólo puede contemplar estructuras de datos jerárquicas, por lo que no se puede utilizar en el diseño de bases de datos relacionales ni distribuidas.

Método de Gane and Sarson

Gane and Sarson es un método *bottom-up* que contempla solamente las etapas de análisis y diseño.

Pasos que sigue:

1.- Construir el modelo lógico del sistema actual (DFDs).

2.- Construir el modelo lógico del sistema a desarrollar incluyendo especificaciones estructuradas (DFDs), un diccionario de datos y la nueva especificación de los procesos.

3.- Realizar un análisis entidad-relación.

4.- Diseñar las tablas de la base de datos que soporten al diseño lógico.

5.- Crear un nuevo modelo físico del sistema que consiste en redibujar el DFD para que muestre de una forma mas precisa los datos del sistema como resultado del análisis entidad-relación y de la normalización.

6.- Dividir el DFD en un conjunto de procesos.

7.- Especificar los detalles de implementación de cada uno de los procesos.

Ventajas

- Las convenciones que se utilizan para los diagramas de flujo de datos son las más rigurosas de todos los métodos.

- Los diagramas son fáciles de entender ya que sólo constan de cuatro símbolos.

Desventajas

- Es un método en el que puede omitir el análisis conceptual ya que empieza con una vista lógica del sistema existente con lo que no se podría satisfacer completamente las necesidades de los usuarios.

- Tiene un alcance limitado, ya que contempla de forma inadecuada las etapas siguientes a la etapa de diseño.

Método Yourdon

El Diseño estructurado Yourdon es un método que cubre la etapa del diseño durante el ciclo de vida de una sistema, fue desarrollado por Ed Yourdon a principios de los 70's. Es altamente recomendado para soluciones basadas en lenguajes de tercera generación y se enfoca al diseño de sistemas en tiempo real.

A finales de 1988 una revista de *software* publicó que el 30.2% de las empresas en Estados Unidos (la mayoría de gobierno, con necesidades de tiempo-real) emplean para el desarrollo de sus proyectos el método Yourdon, mientras que el 29.2% (la mayoría empresas comerciales) emplean métodos *top-down*.

El método Yourdon examina los sistemas actuales, obteniendo nuevas soluciones de los viejos sistemas. Este método es capaz de obtener soluciones para las necesidades de defensa y en general de cualquiera aplicación en tiempo-real.

Yourdon está bien adaptado para la etapa de diseño de los sistemas que requieren de procesamiento en tiempo real y se

requiere de una etapa de análisis extensa. Este método puede dirigir una funcionalidad compleja con solamente una lógica procedural limitada (3GL).

Ventajas

- Es ideal para aplicaciones de tiempo real.
- Tiene un diccionario de datos para la captura de detalles tales como atributos e identificadores únicos.
- No depende de los datos, dado que se concentra en la lógica para los programas.
- Yourdon es lo mejor para sistemas pequeños con requerimientos de procesamiento procedural y sistema de archivos simple.

Desventajas

- No ofrece una técnica propia para el modelado de entidades o diseño de datos, pero recomienda el uso de la técnica Chen de Entidad Relación.

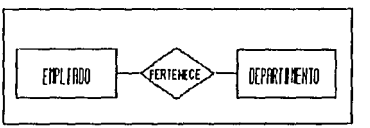


Figura 1.1.6 Modelo Entidad Relación de Chen

Esta técnica es muy superficial ya que no nos da mucha información acerca de las relaciones entre entidades. Esta forma de representación no es muy rigurosa y la normalización es imposible.

- No se crea un modelo de requerimientos de información conceptual.

- Sus diagramas no se prestan para describir los requerimientos del sistema, éstos son vistos únicamente como un mecanismo de procesos anticipados del sistema.

- No está diseñado para alternativas futuras, sus DFD's no están bien relacionados con las otras técnicas empleadas por Yourdon. No es posible saber en esta representación, donde toma lugar el proceso dentro de la organización. Yourdon no define el alcance de cada diagrama de flujo de datos.

- Es un conjunto de técnicas que no están bien acopladas. Las cartas de Estructura no corresponden a los Diagrams Entidad-Relación. No hay funciones que verifiquen el modelo de datos ni es posible verificar la lógica funcional del programa.

- El alcance limitado de Yourdon es su mayor desventaja ya que pasa por alto la etapa de análisis y posteriormente las etapas de desarrollo y se concentra en el diseño de programas procedurales.

Método de Ingeniería de Información (IEM)

El método de Ingeniería de Información cubre las etapas de estrategia, análisis y diseño del ciclo de vida de un sistema.

IEM ofrece diagramas de estructura de datos detallados (entidades, relaciones, atributos), descomposición funcional, flujo de datos y diagramas de dependencia, y definición de reportes y pantallas.

Ventajas

- Es un método que no está enfocado a procesos ni a datos, obtiene los requerimientos de información en un contexto balanceado entre necesidades de usuario y planes estratégicos.

- Los prototipos de entidades y modelado de funciones son hechas en conjunción con el usuario, para asegurar la satisfacción de éste.

Desventajas

- IEM no contempla el entrenamiento a usuarios, documentación, ni diseño en tiempo real. Las etapas de construcción y producción son tratadas en forma mínima.

- Los modelos de IEM son difíciles de leer.

Método DeMarco

DeMarco es un método de análisis estructurado orientado a procesos. Al igual que el método Gane and Sarson es utilizado sólamente en las etapas de análisis y diseño.

Ventajas

- DeMarco empieza el análisis utilizando los sistemas ya existentes, lo cual ayuda a los analistas y diseñadores a tener un punto de inicio.

- El modelado del sistema es iniciado con facilidad.

Desventajas

- DeMarco sólo contempla las etapas de análisis y diseño.

- Tiene un enfoque *bottom-up*, en el cual se diseña el nuevo sistema tomando las especificaciones de uno ya existente, teniendo como problema la posible pérdida de los nuevos requerimientos.

Método Jackson

Jackson es una técnica de diseño orientada a datos, la cual se basa en las estructuras de datos requeridas para obtener la estructura de los procesos, el sistema es visto como un conjunto de procesos. Utiliza básicamente dos técnicas de diagramación: diagramas de redes y diagramas arborescentes.

Ventajas

- Es útil cuando se tienen pequeñas necesidades de programación.

- Es capaz de contemplar requerimientos tanto a nivel sistema como a nivel proceso (módulo).

Desventajas

- Esta limitado al manejo secuencial de registros y al procesamiento en *batch* por lo que ignora la tecnología de bases de datos.

- Solo contempla la etapa de diseño.

CASE*Method

CASE*Method el un método que contempla todas las etapas del ciclo de vida. Mediante la utilización de este método se pueden conocer las necesidades de los usuarios, sin que éstas tengan que ser especificadas por sistemas existentes. Utiliza técnicas *bottom-up* como herramienta para asegurar la exactitud y terminación de una tarea.

Ventajas

- CASE*Method es flexible, ya que se puede adaptar con facilidad a proyectos de cualquier tamaño y complejidad.

- Enfatiza la utilización de referencias cruzadas con el propósito de detectar posibles errores, usando para ello matrices y diagramas de flujo de datos durante las etapas de análisis y diseño.

- Utiliza entidades conceptuales y el modelado de funciones para mostrar los requerimientos esenciales de la empresa, lo que se traduce a un diseño sin errores, es decir, se puede saber desde un inicio si el sistema va a satisfacer completamente las necesidades del usuario.

- Soporta el manejo de prototipos de interfaces con el usuario (pantallas, reportes, etc) durante las etapas de diseño y construcción.

- Sus técnicas de diagramación son fáciles de utilizar y entender.

- Ofrece sugerencias para la elaboración de documentación del sistema.

Desventajas

- No se puede aplicar el desarrollo de sistemas de tiempo real.
- Necesita de una técnica para especificar a detalle el diseño de los módulos.

Observaciones Finales

Después del estudio y análisis comparativo de diversas metodologías de desarrollo de sistemas, hemos podido observar que existen una gran diversidad de métodos para el desarrollo de sistemas.

Sin embargo y de acuerdo al estudio realizado, se observa que algunas metodologías ya han quedado obsoletas, pues se enfocan al desarrollo en lenguajes de tercera generación o bien son para sistemas de base de datos jerárquicas, muchas de éstas metodologías cubren sólo una parte del ciclo de vida de un sistema, es decir son incompletas, la mayoría de ellas se enfocan al diseño haciendo a un lado una de las partes más importantes como lo es el análisis. Algunos métodos emplean para sus modelos diagramas que son muy difíciles de entender, por lo cual se requiere de un entrenamiento especial para aplicarlos, otros métodos por el contrario emplean diagramas tan simples que por lo mismo dichos diagramas resultan incompletos.

Por otro lado basamos en la tendencia actual del mercado de *Software* hemos encontrado lo siguiente: una revista de *Software* publicó que hay un interés creciente en los métodos de ciclo de vida completo y *top-down*. De la misma manera en que la programación estructurada reemplazó al código no estructurado de los 70's así también en los 80's se muestra una adopción gradual de las técnicas estructuradas que cubren completamente el ciclo de vida de un sistema.

Las características del sistema que se desea implantar, se adaptan completamente a un modelo de bases de datos relacional y hemos podido observar con el estudio de las metodologías, que muchas de ellas no fueron pensadas para este tipo de implementación.

Se necesita implantar un sistema de información compartida, que sea sencillo de aprender por los usuarios finales y que incluya una documentación adecuada, además de que contemple modificaciones futuras conforme la empresa crezca.

Para el caso de estudio de esta tesis se necesita un sistema de bases de datos relacional y se empleará para este desarrollo los productos basados en la metodología *CASE*.

El Método de Desarrollo de Sistemas **CASE*Method**, es una metodología completa y moderna enfocada a la tecnología de bases de datos relacional que cubre completamente el ciclo de vida de un sistema, nos brinda una estrategia de planeación de sistemas de muy alto nivel, nos permite realizar un análisis detallado del sistema. Cuenta con técnicas de referencia cruzada que aseguran que el modelo sea completo y exacto. Sus DFD's son más rigurosos que otros métodos. Se basa en la comunicación entre usuarios y desarrolladores en el diseño de aplicaciones de 4a. generación y estados de post-diseño.

El método *CASE* es un método *TOP-DOWN*, es un método de autochequeo, la implementación es independiente de los requerimientos - tanto de *hardware* como de *software*, está diseñado para el cambio, se implementa en forma metódica en todo tiempo.

1.2 REQUERIMIENTOS DE LAS DIFERENTES AREAS DE LA COMPAÑIA

Como se mencionó en la introducción a este trabajo de tesis, Oracle de México cuenta con cuatro direcciones, las cuales a la fecha hacen el manejo de su información en forma separada, todas ellas en general tienen problemas para mantener su información integrada, no redundante y consistente; sin embargo, aun cuando comparten información, las necesidades de procesamiento de los datos son diferentes para cada área. A continuación se detallan cuales son las funciones y los problemas específicos a los cuales de enfrenta cada una de estas direcciones.

DIRECCION DE FINANZAS

La dirección de Finanzas y Administración se encarga de llevar el control tanto de ingresos como de egresos de la compañía, este control incluye registro de contratos de venta del producto ya sea licencias o soporte técnico, contratos de proyectos de consultoría, contratos de servicios educacionales, manejo de cuentas por cobrar y por pagar, tanto a proveedores como a empleados, contabilidad general, impuestos, presupuestos, reportes, tesorería, y recursos humanos.

En esta área se manejan varios procedimientos, algunos de ellos en forma manual y almacenando la información en folders y archiveros. Los procedimientos manuales son:

- Reconocimiento de venta
- Control de contratos de proyecto
- Control de reservas incobrables
- Notas de crédito y cancelaciones
- Salvaguardia de activos fijos
- Política de crédito a clientes
- Contratación de Personal
- Compra o renta de activo fijo
- Cuentas de gastos a comprobar del personal de Oracle
- Control de servicios de viaje

Algunos otros procedimientos requieren de un sistema automatizado en donde se tenga registro de lo que se ha pagado o cobrado. En general, la mayoría de los procedimientos que a continuación se mencionan son manejados en una hoja de cálculo Lotus (en una microcomputadora PC con MS-DOS). Esta forma de

control no es un sistema propiamente dicho sino más bien una forma de almacenar la información que se maneja.

- Pago de gastos de viaje y representación
- Pago de cuenta de gastos semanal
- Control de caja chica y fondos fijos de caja
- Control de compras
- Gastos médicos complementarios
- Anticipos para viajes propios del negocio
- Control de cuentas por pagar a proveedores
- Control de los desembolsos en efectivo
- Pago a proveedores de bienes y servicios en Monterrey
- Procedimiento de Licencia a Prueba

Otros procedimientos manejados por el área de Finanzas si cuentan con un sistema de control automatizado, programado en la hoja de cálculo Lotus en una microcomputadora PC con MS-DOS. Este sistema está adecuado a las necesidades propias de la empresa, y se encarga de realizar los siguientes procedimientos:

- Control de Nómina
- Expedición de Cheques
- Pago de Nómina y reembolsos a empleados de Monterrey
- Procedimientos de cierres contables y anuales

Muchos de los procedimientos mencionados no necesitan de un sistema automatizado para llevarse a cabo, dado que la mayoría de éstos son solamente decisiones que deben tomarse o bien registrarse en algún archivo la posesión de algún bien.

El objetivo del presente tema de tesis es desarrollar un sistema que automatizará lo referente a los contratos de licencia, soporte técnico de los clientes, contratos de consultoría y contratos del área educacional, así como la correspondiente facturación de los mismos. Esta es una parte importante de automatizar ya que se requiere un mayor control de los contratos que se han vendido. Además, la información contenida en un contrato es importante no sólo para el área financiera sino también para la gente del área técnica y de ventas, a quienes les interesa conocer fechas de vencimiento de un contrato o el nombre y la dirección de los contactos.

Con respecto a contratos se desea saber quien es el vendedor que atiende a cierta cuenta, que tipo de contrato es, y una parte muy importante para las finanzas, es el costo al cual fue vendido cada uno de los productos estipulados en el contrato.

El procedimiento que se debe seguir para que la venta de un contrato sea reconocida es el siguiente: el vendedor será responsable de entregar antes del día último del mes , el contrato o la orden de compra firmado tanto por Oracle como por el Cliente y la carta de recepción de los productos por parte del cliente. Es necesario que en un contrato se haga el desglose necesario del reconocimiento de ingresos para el área educativa y de soporte técnico, además esta forma de desglose deberá ser autorizada por el gerente de soporte técnico y por el encargado del área educativa.

Otro tipo de contrato que necesita de un control automatizado son los contratos de licencias de prueba, en estos se debe registrar la fecha de inicio del contrato, dado que solamente se pueden tener a prueba los productos en plazos de 30, 60 y hasta 90 días. Después de esa fecha se deberá elaborar el formato de contrato correspondiente y se turnará al área de cuentas por cobrar para la elaboración de la factura.

Cada contrato deberá de especificar una lista de los productos y/o servicios comprados y el precio por separado de cada producto, las instrucciones de embarque y las condiciones de pago. Deberá especificarse asimismo si se trata de un contrato por plataforma o por lugar, es decir, en cuantas computadoras se instalarán los productos.

Un contrato puede incluir Soporte Técnico estándar y actualización, o solamente actualización. Los ingresos derivados por soporte técnico y/o actualización son reconocidos al inicio del período del soporte. Si se incluye el soporte técnico en cuota por licencia, el ingreso es reconocido de la siguiente forma: la cuota de licencia menos la parte atribuible de soporte técnico.

Otra de las actividades de la dirección finanzas es la de controlar el inventario de los productos Oracle, así como de mantener en bodega las cintas con los productos necesarios. Esta área es llamada área de compras o **"shipping"**, dentro de la dirección comercial. Esta área se encarga de solicitar los productos necesarios a Oracle Corporation y verificar que estos sean entregados en México, en el tiempo mínimo para que sean entregados al vendedor, lo cual no debiera llevarse más de 15 días después de solicitado el producto. En general, en esta área se controla la entrada y salida de los productos manuales, cintas, paquetes, revistas, etc.

DIRECCION COMERCIAL Y DE VENTAS

Es el área de la compañía encargada de vender y distribuir los productos de Oracle, herramientas y aplicaciones. Es el área encargada también de promocionar los productos, realizando estudios de mercado y de la competencia, así como llevando a cabo actividades de publicidad.

Una de las funciones principales de la dirección comercial es la de vender los productos de Oracle, ya sean licencias de uso del *software* o soporte técnico y/o actualización, así como venta de créditos educacionales.

Cada vendedor es encargado de cierto sector de clientes, los sectores se encuentran divididos de acuerdo a la función de las empresas que lo integran, así por ejemplo, tenemos el sector educacional (Universidades) o el sector financiero (Bancos), un tercer sector es PEMEX, este sector esta integrado únicamente por la Cia. Petróleos Mexicanos y constituye un sector dado la importancia que este cliente representa para la Cia. Oracle. Cuando un vendedor ha hecho un contacto, deberá formalizar la intensión de compra por parte del cliente y presentar esta formalización al director del área comercial, con la forma del contrato ya sea de licencia, o de licencia a prueba, debidamente llenada y firmada por el cliente. En esta forma de contrato se deberá de especificar los productos que se venderán y el tipo de máquina en el cual serán instalados. En el caso de que se trate de una licencia a prueba, se deberá registrar la fecha en la cual inicia el compromiso de prueba, haciendo entrega de los productos al cliente en esa misma fecha y se dará el tiempo de prueba a partir de ese día.

Si se tratara de un contrato de licencia, entonces se deberá especificar el costo de venta de cada producto y del total se hará el desglose de lo correspondiente a soporte técnico y el costo de los créditos educacionales vendidos al cliente.

Una vez que el director comercial y el director del área de postventa firman el contrato, se hace la solicitud de productos al encargado de "*shipping*" y posteriormente se hace la entrega de los productos al cliente, quien a su vez deberá firmar la carta de entrega respectiva. Cuando el contrato ha sido firmado por ambas partes involucradas, Oracle y el Cliente, éste deberá ser entregado al departamento de finanzas para la elaboración de la factura; posteriormente, cuando el cliente haya pagado, se hará el cálculo de las comisiones correspondientes al vendedor.

AREA EDUCACIONAL

El Area Educacional es la encargada de impartir cursos acerca del uso de los productos que distribuye la empresa.

El coordinador del Area Educacional debe programar cursos tomando en cuenta el número de instructores disponibles, el número de salones disponibles y sobre todo el tipo de curso, para asignar a la persona adecuada. La asignación de la persona adecuada es hecha con base en un archivo histórico, en donde se registra cuando y que cursos ha impartido cada instructor.

Los cursos son programados asignando a cada uno un instructor. Para asistir a un curso el cliente debe reservar un lugar haciendo una llamada telefónica. Una vez que el cliente ha reservado su lugar, deberá confirmar por escrito su asistencia al curso.

Si el cliente tiene créditos educacionales, se le descuenta el número de créditos equivalentes al valor del curso y se elabora una factura por el valor del curso.

Una vez que se ha confirmado el curso el cliente asiste.

Un aspecto importante es que al terminar cada uno de los cursos se efectúa una encuesta, que se aplica a los alumnos sobre la forma en que fue impartido el curso, con el objeto de mejorar la calidad de los mismos. También se elabora una constancia de participación para los alumnos.

En ocasiones el cliente desea saber en que cursos ha empleado sus créditos y cuantos tiene disponibles, el coordinador debe obtener esta información consultando el archivo histórico.

En el sistema por desarrollar se contemplarán los siguientes aspectos:

- Programación de cursos
- Recepción de una petición para asistir a los cursos (reservación)
- Confirmación de la impartición del curso
- Confirmación de la asistencia a un curso por un cliente

- Cancelación de cursos

- Generación de reportes

  . número de cursos impartidos por cada instructor
  . calificaciones para cada instructor después de impartir un curso
  . lista de asistencia a los cursos
  . encuestas de los cursos impartidos
  . generación de diplomas
  . otros

- Facturación de los cursos

- Acoplamiento del sistema encuestas a este sistema

AREA DE SOPORTE TECNICO

El Area de soporte técnico es la encargada de asistir a los clientes en caso de algún problema en el funcionamiento de los productos Oracle. Además, se encarga de mantener a los clientes actualizados, en cuanto a los productos Oracle.

El Area de Soporte Técnico representa un recurso importante para el Area educacional, ya que a menudo, los ingenieros de soporte deben auxiliar a ésta impartiendo cursos.

El área de Soporte Técnico está integrada por ingenieros que tienen a su cargo diversos productos y cada uno de ellos es especialista en los productos que se le han asignado.

Por cada producto se tienen dos especialistas cuando menos.

Para que un cliente pueda recibir atención para una solicitud de soporte técnico, es necesario que siga ciertas políticas establecidas.

Para hacer una solicitud de asistencia técnica (SAT), el cliente debe hacer una llamada telefónica a la mesa de soporte técnico. En este lugar se recibe la solicitud y de ser posible se resuelve en ese momento.

Si la solicitud de asistencia técnica requiere de un mayor conocimiento del producto, la SAT es transferida al especialista del producto para darle seguimiento hasta alcanzar una solución.

El control del rendimiento de esta área es realizada por medio de estadísticas que se generan mensualmente, en donde se especifica cuantas solicitudes de soporte técnico se atendieron por cliente, por producto, y quien atendió la solicitud entre otros aspectos.

El sistema contemplará los siguientes aspectos:

- Recepción de llamadas solicitando el servicio de Soporte Técnico
- Asignación de las llamadas a la persona experta en la solución del problema
- Seguimiento de las llamadas
- Generación de reportes
  - . número de SAT's
  - . número de solicitudes atendidas por ingeniero
  - . número de solicitudes atendidas por producto
  - . número de solicitudes atendidas por plataforma
  - . otros

1.3 RESULTADOS OBTENIDOS EN ESTE CAPITULO

Aplicando las dos primeras etapas de la metodología CASE*Method (Estrategia y Análisis) se obtuvieron los siguientes resultados.

DIAGRAMAS FUNCIONALES

Con base en la información recopilada acerca de los requerimientos de la empresa descritos anteriormente, se construyeron los diagramas funcionales mostrados en las figuras 1.3.1 a la 1.3.5 . Estos diagramas muestran de una manera jerárquica los procedimientos que se siguen en cada una de las diferentes áreas con el fin de cumplir con sus objetivos.

Un diagrama funcional o jerarquía de funciones es la simple agrupación de funciones en una estructura jerárquica, representando todas las funciones que se realizan o deberían realizarse dentro de una área de la empresa.

Un diagrama funcional modela los procedimientos de la empresa en forma jerárquica hasta que se alcanza un nivel de descomposición en el cual es posible asociar cada función con un módulo o una subrutina. Son fácilmente generados a partir de las entrevistas, esta técnica maneja diseño de módulos. El modelo de funciones a un nivel alto puede ser desarrollado durante la etapa de estrategia para soportar completamente las necesidades de la empresa.

Los Diagramas de Funciones de CASE*method están etiquetados o numerados para tener una referencia-cruzada dentro de la jerarquia y los cuales nos permiten fácilmente identificar donde se inicia una función, lo que se requiere cumplir para llegar a ella y como se ajusta dentro de toda la organización . Cada función puede a su vez ser descompuesta en otras funciones lo cual es representado por puntos suspensivos (...). No es necesaria ni se tiene implícita ninguna secuencia o procedimiento en esta técnica.

La siguiente tabla muestra la relación entre las figuras y los diagramas funcionales correspondientes a cada uno de los diferentes módulos que componen nuestro sistema.

Figura	Módulos
1.3.1	Finanzas
1.3.2	Ventas y Administración
1.3.3	Educacional
1.3.4	Soporte Técnico
1.3.5	Actualizaciones

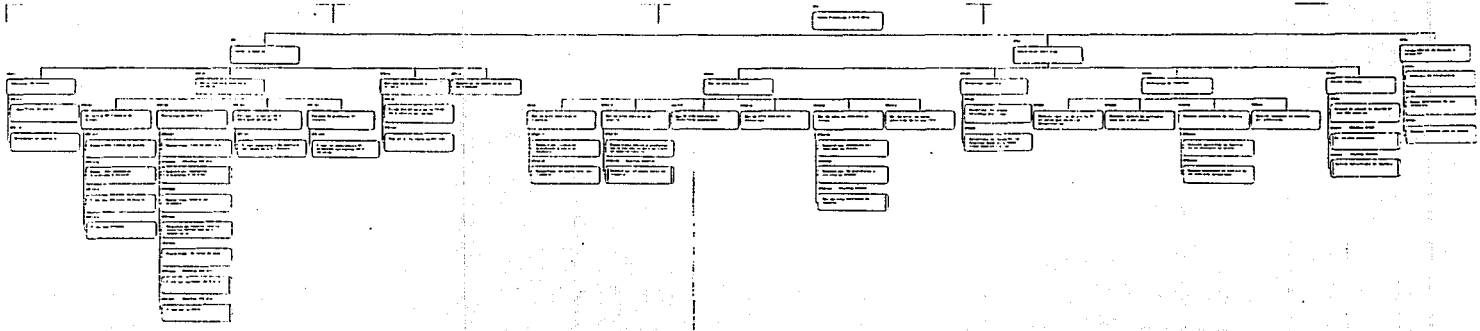


Figura 1.1.2 Diagrama Funcional de Ventas y Administración

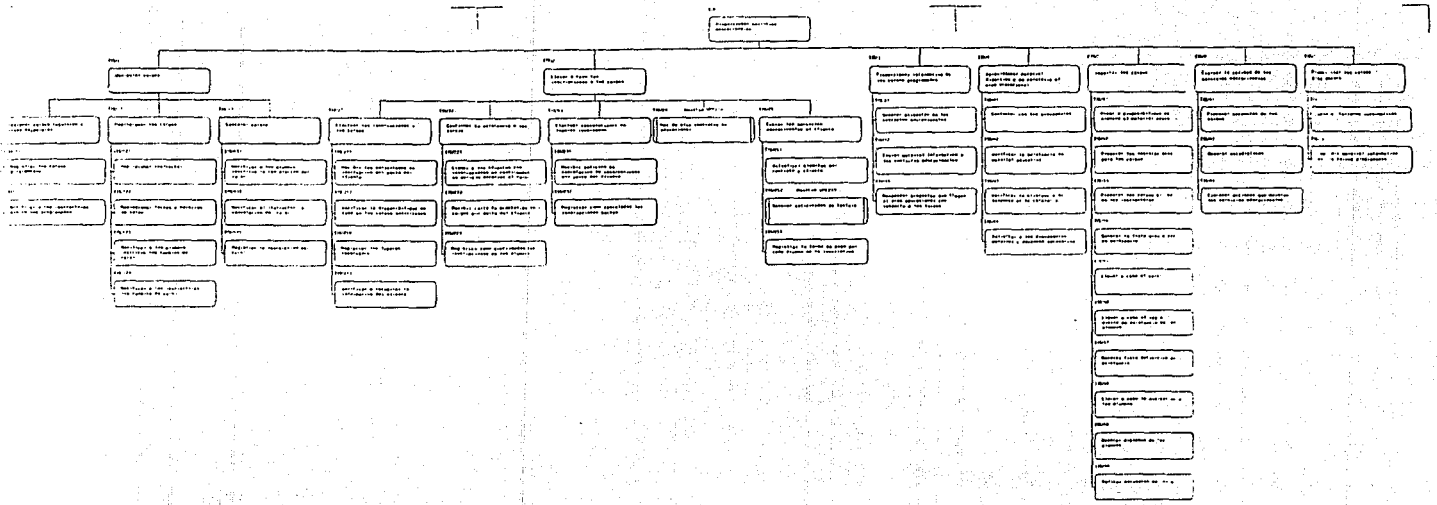


Figura 1.3.3 Diagrama Funcional del Area Educacional

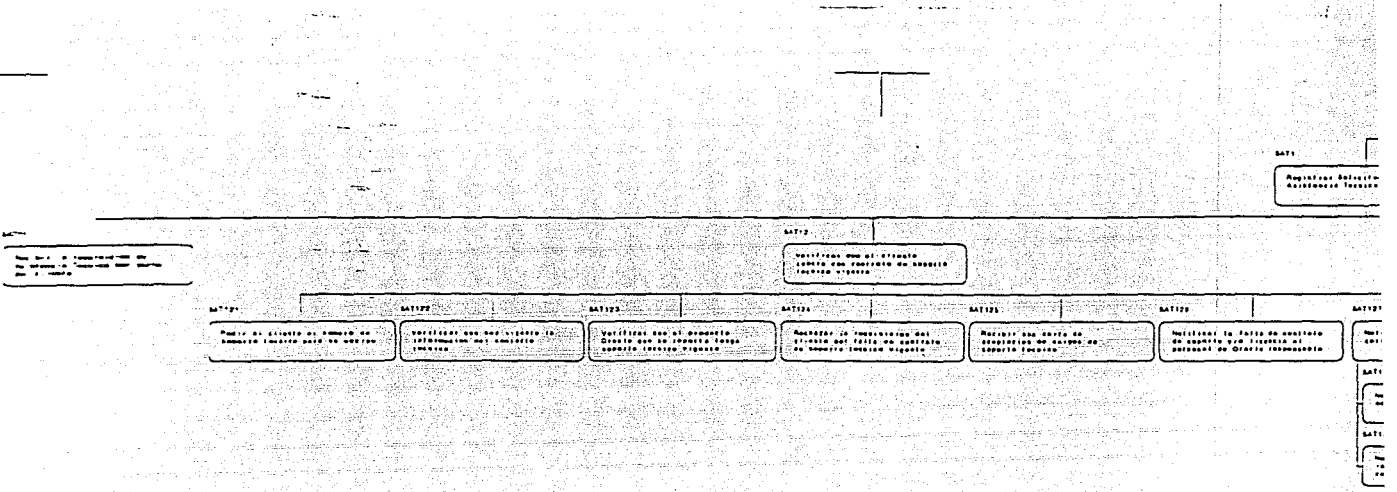
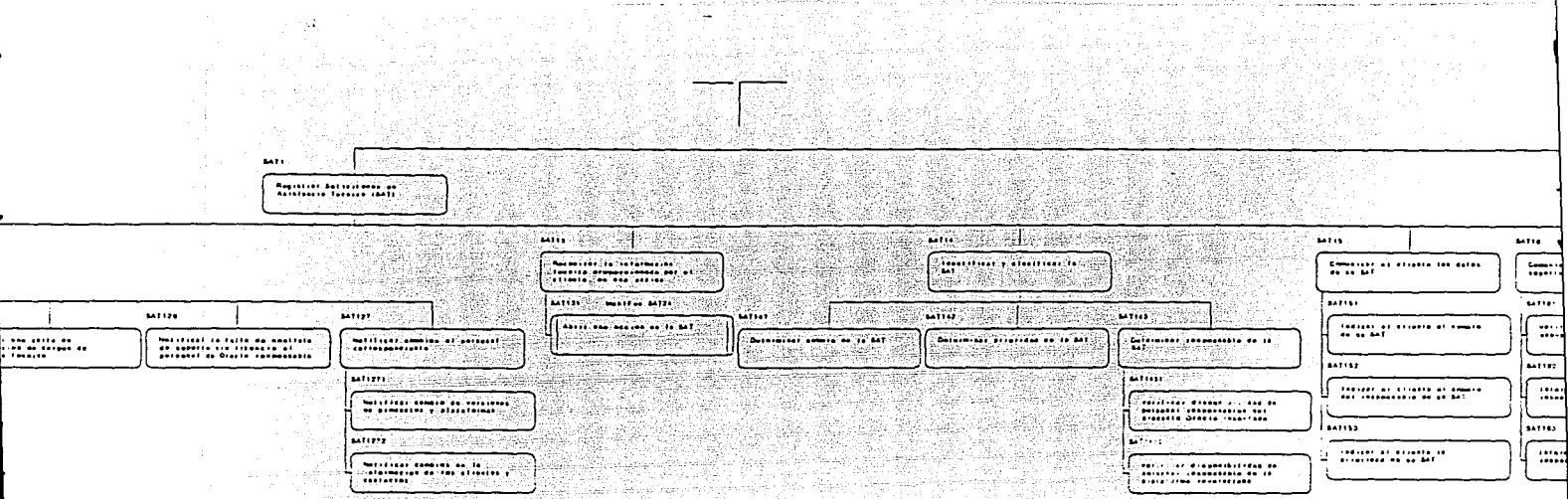
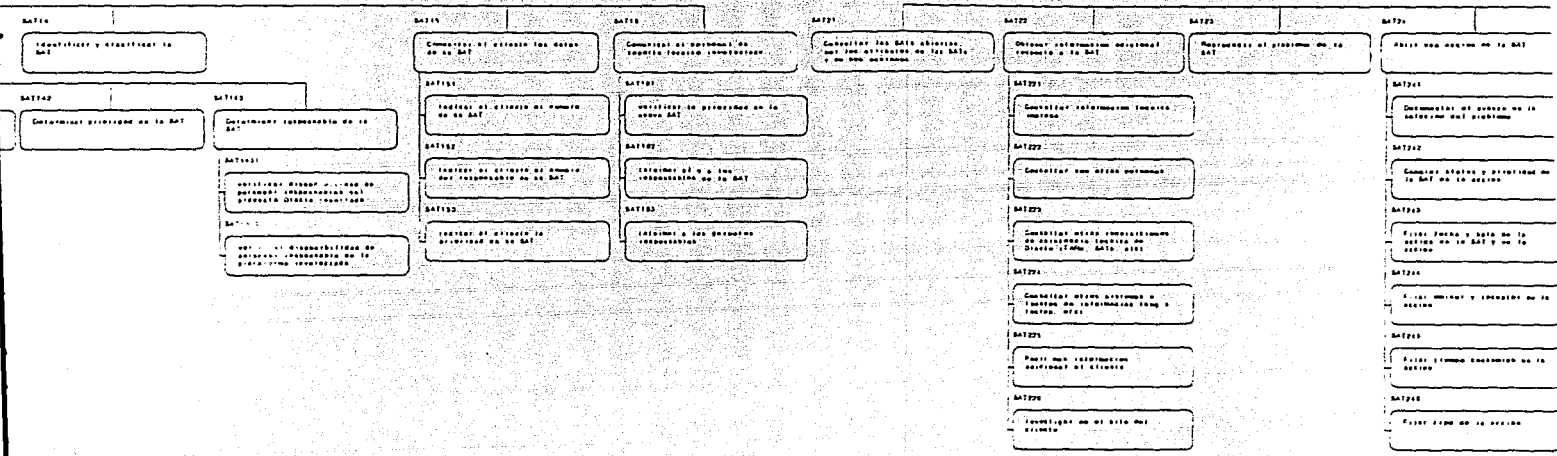
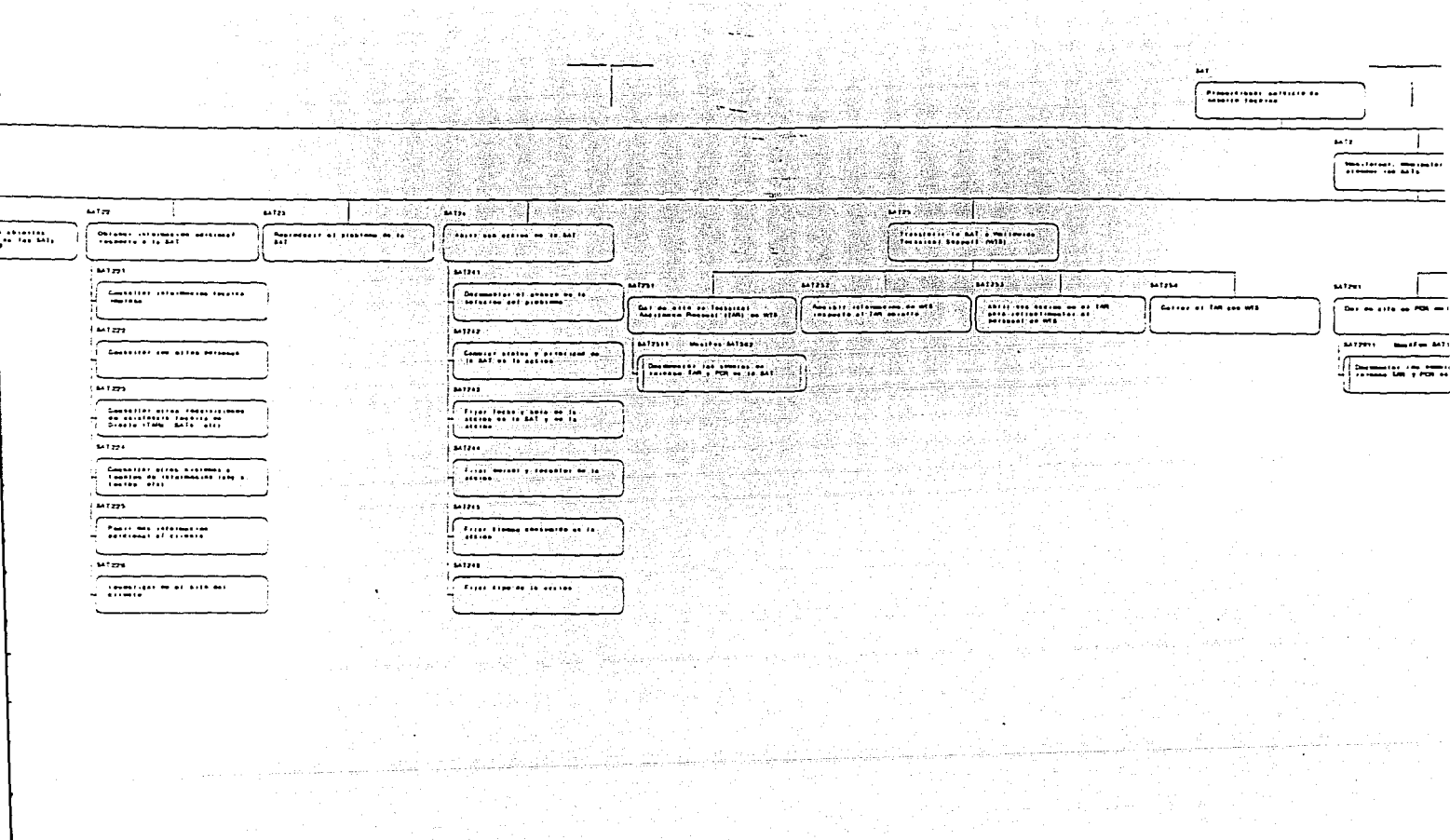
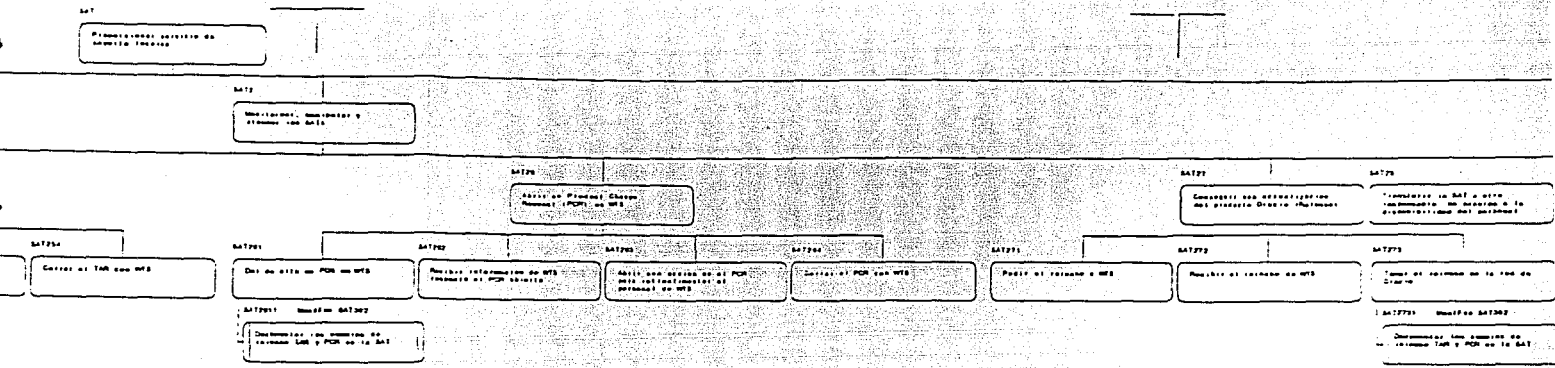


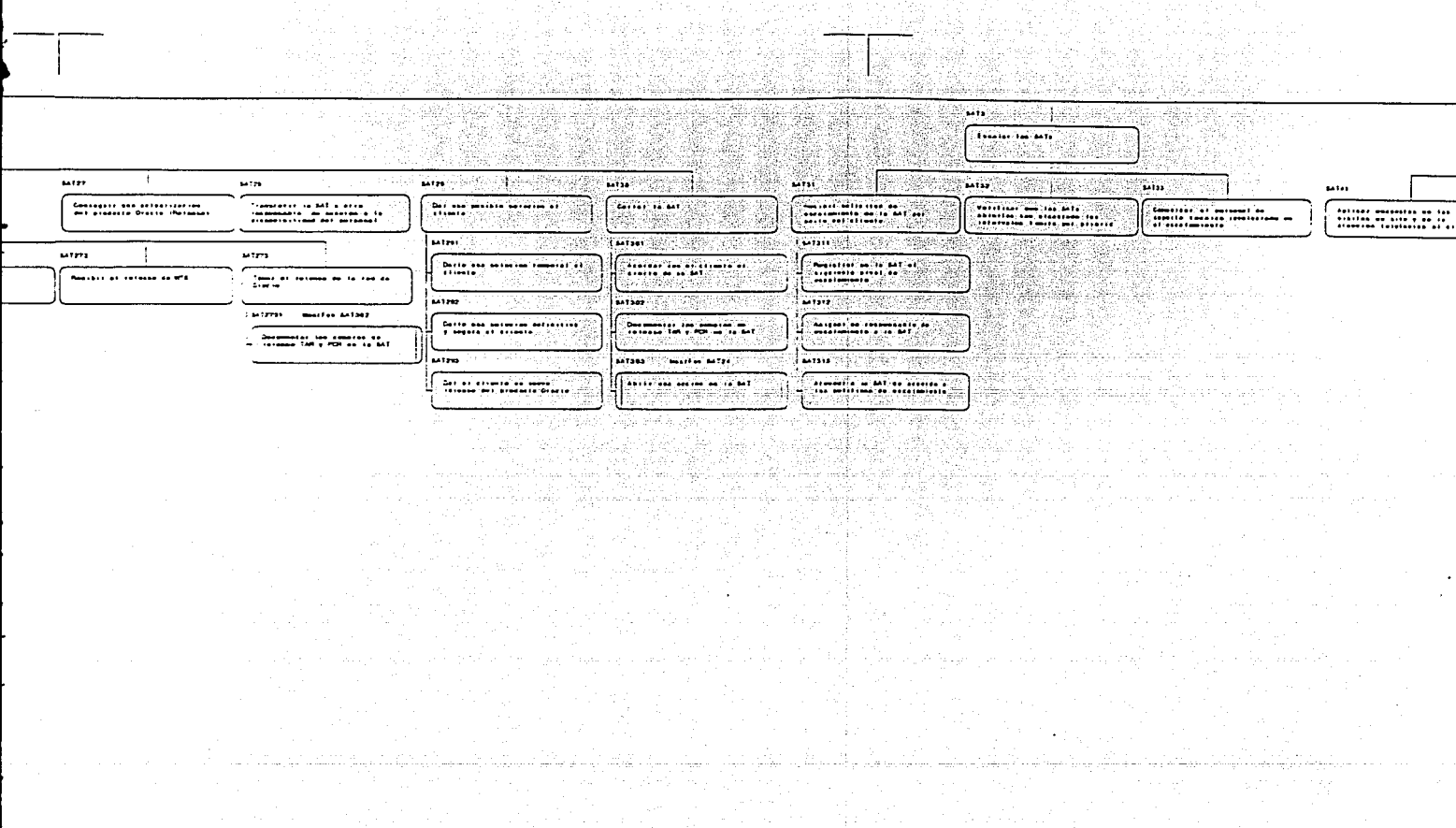
Figura 1.3.4 Diagrama Funcional del Area de Soporte Tecnico

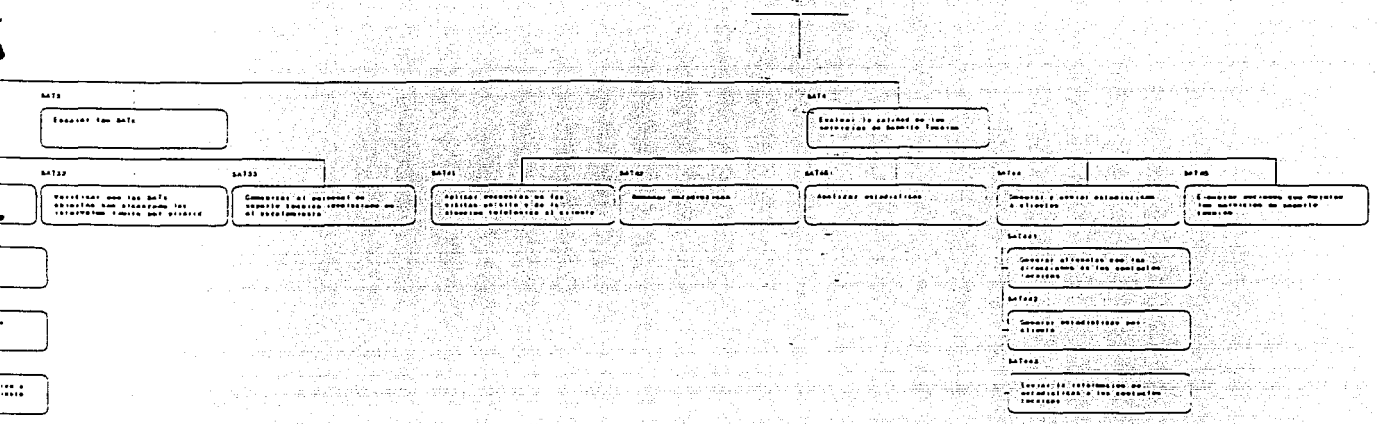









SAT27
SAT28
SAT29
SAT30
SAT31
SAT32
SAT33
SAT34
SAT35
SAT272
SAT273
SAT2701
SAT2702
SAT2703
SAT301
SAT302
SAT303
SAT3111
SAT312
SAT313



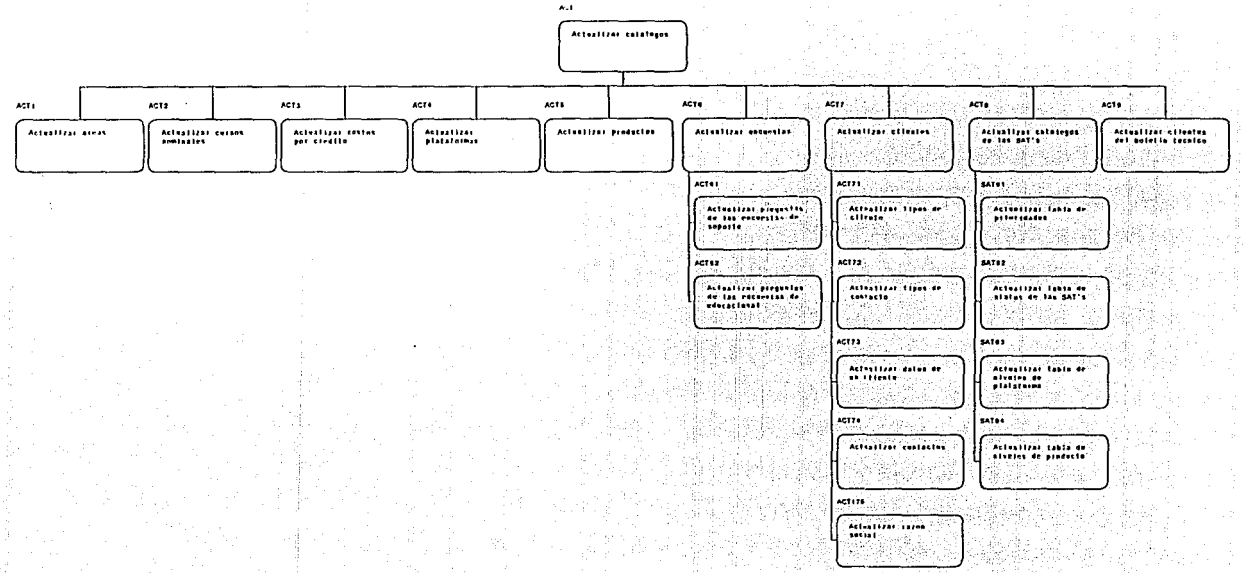


Figura 1.3.5 Diagrama Funcional del Modulo de Actualizaciones

DIAGRAMA ENTIDAD-RELACIÓN

El modelado entidad-relación es una técnica empleada en la definición de las necesidades de información de una organización. Estos diagramas proveen de una fuente fundamental para garantizar un alto nivel de calidad, asegurando que se cubren las necesidades de la empresa. El modelado entidad-relación involucra la identificación de los elementos de importancia en una organización (entidades), las propiedades de esos elementos (atributos) y como están relacionados uno con otro (relaciones).

Los objetivos del modelado entidad-relación son:

- Proporcionar un modelo preciso de las necesidades de información, el cual actúa como una base para el desarrollo de nuevas características al sistema.

- Proporcionar un modelo independiente de cualquier almacén de datos y método de acceso, para permitir que las decisiones objetivas sean hechas de acuerdo a las técnicas de implementación y puedan coexistir con el sistema existente.

El diagrama entidad-relación es una parte que modela el negocio o empresa y es producido en el estado de análisis, el cual es parte de todo el ciclo de vida de un sistema. Este diagrama nos muestra gráficamente las entidades y las relaciones vitales entre ellas.

El proceso de crear este diagrama es llamado "modelado de entidades". Los términos modelo de entidades, modelo entidad-relación o modelo entidad/relación son todos sinónimos del diagrama entidad-relación.

Durante los últimos veinte años los sistemas de cómputo se han vuelto más y más sofisticados. Han madurado a través de la evolución de:

- experimentación
- sistemas funcionales aislados
- sistemas departamentales
- sistemas operativos integrados
- automatización de oficinas
- manejo de información

Dado este progreso es necesario minimizar duplicidades y asegurar una verdadera integración de datos y la disponibilidad real de la información.

El modelado de entidad-relación moderno, en asociación con las sofisticadas herramientas *CASE*, proporcionan un medio oportuno y efectivo para definir y controlar las definiciones de las necesidades de información.

Este tipo de herramientas han sido usadas por casi veinte años en diversas circunstancias para proporcionar soluciones implementadas en sistemas de bases de datos relacionales, de red, jerárquicas o bien en sistemas de cómputo convencionales y para apoyar en el diseño de formas y en el control y refinamiento de los procedimientos.

Convenciones Básicas y Definiciones Empleadas

Entidad: Una entidad es un elemento u objeto con significado propio, ya sea real o imaginaria y cuya información se necesita conocer y guardar.

Una entidad es representada gráficamente por una caja rectangular con las orillas redondeadas y con el nombre de la entidad dentro, en singular y con letras mayúsculas, como se muestra en la *Figura 1.3.6*.

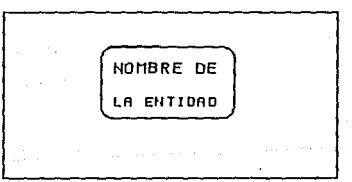


Figura 1.3.6
Representación Gráfica de una Entidad

Cada entidad debe tener un identificador único.

Cuando se modelan las entidades no hay ningún significado especial dado al tamaño de una entidad, las cajas pueden ser alargadas, ensanchadas o acortadas simplemente para ayudar a la lectura de los diagramas.

Existe otro tipo de elementos en el modelado, los cuales ofrecen más compromisos y oportunidades, ellos son los sub-tipos y super-tipos de entidades.

Un sub-tipo de entidad es también una entidad. Una entidad puede ser dividida en dos o más sub-tipos mutuamente exclusivos, cada uno de los cuales tienen atributos o relaciones en común. Estos atributos y/o relaciones son definidas explícitamente sólo una vez en el nivel más alto. Los sub-tipos pueden tener sus propios atributos.

Un sub-tipo de entidad (p.e. LICENCIA ver fig. 1.3.7) implícitamente cuenta con todos los atributos y relaciones de la entidad de más alto nivel conocida como super-tipo de entidad (p.e CONTRATO ver fig. 1.3.7).

Super-tipo de entidad es el nombre que se le da a una entidad que tiene sub-tipos, una entidad puede a su vez ser sub-tipo y super-tipo de entidades.

Para leer los super-tipos se puede agregar la frase "el cual puede ser un A,B,C o D" por ejemplo:

Un CONTRATO el cual puede ser LICENCIA, SOPORTE ...

Cuando leemos un sub-tipo agregamos la frase "el cual es un tipo de" por ejemplo:

Una LICENCIA el cual es un tipo de CONTRATO.

Un ejemplo de la representación de los sub-tipos y de los super-tipos de entidades se muestra en la *figura 1.3.7*

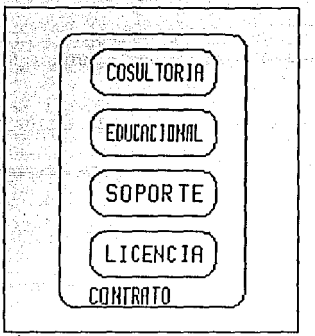


Figura 1.3.7
Representación de Sub-tipos
y Super-tipos de Entidades

Relación: Es una asociación con un nombre asignado y que representa alguna característica entre dos entidades. Una relación es binaria, en el sentido de que siempre es una asociación entre exactamente dos entidades o entre una entidad y ella misma.

Cada relación tiene dos extremos en cada uno de los cuales hay:

- Un nombre
- Un grado o cardinalidad (cuantos)
- Opcionalidad (Opcional u Obligatoria)

Estas propiedades son empleadas para describir la asociación en un extremo, ambos extremos deben estar definidos.

Una relación es representada por una línea que une dos entidades o bien recursivamente une una entidad consigo misma (*Figura 1.3.8*).

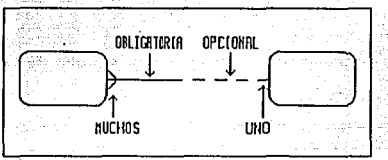


Figura 1.3.8
Representación de Relaciones entre entidades

Para representar el grado de muchos la línea de relación se une a la entidad en tres puntos conocido como "pata de gallo". Para representar el grado de uno la línea se une en un solo punto.

Los nombres de las relaciones deben ser colocados en cada extremo de la línea y se colocan cerca de la entidad apropiada y en letras minúsculas.

La obligatoriedad en una relación es representado por una línea continua y la opcionalidad se representa con una línea discontinua.

Cuando la relación es obligatoria se lee "debe ser o estar".
Cuando la relación es opcional se lee "puede ser o estar".
Si el grado es muchos se lee "uno o más"
Si el grado es uno se lee "uno y sólo uno"

El tipo de relación más común es la que tiene el grado de muchos a uno y es obligatoria del lado de muchos y opcional el lado de uno.

Para leer una relación se debe emplear la siguiente sintaxis para ambos lados de la relación:

Cada ENTIDAD-A [debe ser/estar | puede ser/estar] Nombre de relación [una y solo una | una o más] ENTIDAD-B (plural)

Por ejemplo y de acuerdo al diagrama entidad-relación de la *figura 1.3.11*:

Cada EMPLEADO debe ser parte de una y sola una AREA

y

Cada AREA puede estar formada por uno o más EMPLEADOS.

En la *figura 1.3.9* se muestra una relación recursiva con propiedades idénticas a las relaciones entre dos entidades.

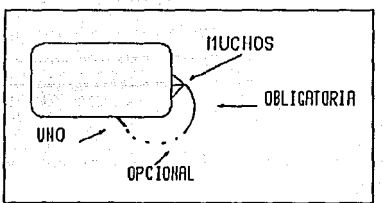


Figura 1.3.9
Relación Recursiva

En el diagrama de la *Figura 1.3.10* se muestran dos entidades y la relación entre ellas. La línea tiene una ramificación al final en el lado izquierdo y termina de forma sencilla en el lado opuesto derecho, indicando que puede haber muchos contratos para un solo cliente, es decir, es una relación muchos a uno. La línea solida indica que un contrato debe ser firmado por uno y sólo un cliente y la línea punteada indica un cliente puede tener uno o más contratos, es decir habrá clientes que no tengan necesariamente que firmar un contrato.

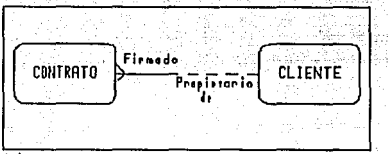


Figura 1.3.10
Relación entre Clientes y Contratos

La relación mostrada en la *figura 1.3.10* puede ser leída de izquierda a derecha como:

- Cada CONTRATO debe ser firmado por uno y solo un CLIENTE

La relación leída de derecha a izquierda dice:

- Cada CLIENTE puede ser propietario de uno o más CONTRATOS.

En la *figura 1.3.11* se muestra el diagrama entidad-relación que modela las necesidades de automatización de las 4 diferentes áreas de la compañía Oracle de México.

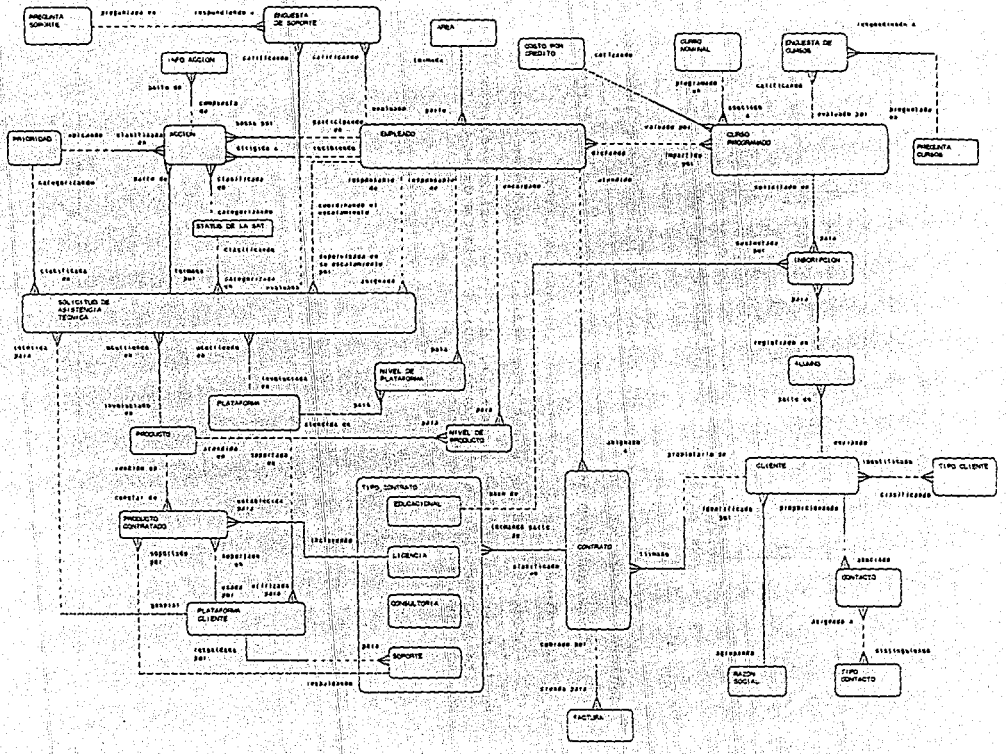

PREGUNTA SOPORTE
ENCUESTA DE SOPORTE
AREA
COSTO POR CREDITO
CURSO NORMAL
ENCUESTA DE CURSOS
INFO ACCION
ACCION
EMPLEADO
PRIORIDAD
CURSO PROGRAMADO
PREGUNTA CURSOS
STATUS DE LA SAT
INSCRIPCION
SOLICITUD DE ASISTENCIA TECNICA
NIVEL DE PLATAFORMA
ALUMNO
PLATAFORMA
PRODUCTO
NIVEL DE PRODUCTO
CLIENTE
TIPO CLIENTE
TIPO_CONTRATO
EDUCACIONAL
PRODUCTO CONTRATADO
CONTRATO
LICENCIA
CONTACTO
CONSULTORIA
PLATAFORMA CLIENTE
SOPORTE
RAZON SOCIAL
TIPO CONTACTO
FACTURA

Figura 1.3.11

ATRIBUTOS

Un atributo es cualquier detalle que nos sirve para calificar o expresar el estado de una entidad o bien cualquier descripción de una cosa que tenga significado dentro del sistema. Un atributo puede ser texto, números, fechas, una pintura, un sentimiento, etc. pero para propósitos de procesamiento de datos se emplean textos, números y fechas generalmente.

Dentro de la metodología Case*Method los atributos suelen representarse dentro de las entidades con sus nombres en minúsculas y en singular *(ver figura 1.3.12)*.

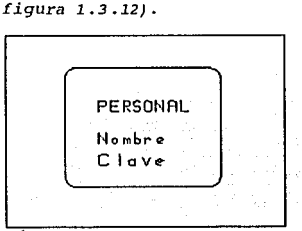


Figura 1.3.12
Representación de los Atributos en la Metodología *CASE*Method*

Sin embargo, en el diagrama entidad-relación no es necesario mostrar los atributos, pero esto podría ser útil para distinguir más fácilmente las entidades.

Los atributos deben cumplir con las siguientes reglas:

- Un atributo describe una entidad.
- Los atributos repetidos en una entidad deben removerse.
- Los nombres deben ser en singular.
- Pueden ser un identificador único, útil al sistema.

IDENTIFICADORES UNICOS

Cada entidad debe ser identificada de forma única, de tal forma que cada elemento de esa entidad es separado de todos los demás elementos de la misma y de otras entidades e indicado de forma única. El identificador único puede ser un atributo, una combinación de atributos, una combinación de relaciones o atributos.

DOMINIOS

Un dominio es un conjunto de reglas de validación de la empresa y está formado de restricciones y otras propiedades que aplican a un grupo de atributos, puede hablarse aquí de cantidades monetarias, las cuales deben ser enteros positivos o bien de las fechas que deben tener un formato fijo de definición.

Puede usarse por ejemplo:

- Una lista de valores.
- Un rango.
- Un calificador de una lista o rango.
- Cualquier combinación de estos.

Los siguientes reportes muestran los atributos que serán manejados para el desarrollo de este sistema y los identificadores únicos de cada una de las entidades generadas durante el análisis del sistema, así como los dominios de valores a los cuales pertenecen.

CASE * Dictionary

Report : ENTITIES AND THEIR ATTRIBUTES

Filename : cdattb14.lis

Run by : TESIS

Report Date : 04-NOV-92 04:56pm

Parameter values

Application System : ORACLE
Version : 1

Entities and their Attributes

Entity	Attribute Name	Optional	Format	Size	Dec Pl	Attribute Notes
ACCION	FECHA_ACCION	N	DATE			Fecha y hora en que fue creada la accion
	NO_ACCION	N	NUMBER	3		Numero de accion en la SAT
	STATUS_ACCION	N	NUMBER	1		Determina el estado de la accion
	TIEMPO	Y	CHAR	8		Tiempo empleado en esta accion
	VISIBLE	Y	CHAR	1		Permite o inhibe que la accion aparezca en un reporte
ALUMNO	NOMBRE	N	CHAR	32		Nombre del alumno
	NO_ALUMNO	N	NUMBER	6	0	Codigo del alumno
AREA	NOMBRE	N	CHAR	32		Nombre del area
	NO_AREA	N	NUMBER	2		Identifica en forma unica un area
CLIENTE	NOMBRE	N	CHAR	32		Nombre del area o direccion
	NO_CLIENTE	N	NUMBER	6	0	Codigo del cliente (area o direccion)
	COMENTARIO	Y	CHAR	80		Comentario general acerca de la situacion particular del cliente
	DIRECCION1	Y	CHAR	60		Direccion del area o direccion (1er renglon)
	DIRECCION2	Y	CHAR	60		Direccion del area o direccion (2o renglon)
	FAX	Y	CHAR	20		Numero telefonico general del area o empresa
	TELEFONO	Y	CHAR	20		Telefono general del area o empresa
CONSULTORIA : Sub-type of TIPO CONTRATO	NO_CONSULTORIA	N	NUMBER	6	0	Identificador de cada contrato

Entities and their Attributes

Entity	Attribute Name	Optional	Format	Size	Dec Pl	Attribute Notes
CONTACTO	NOMBRE	N	CHAR	32		Nombre del contacto
	NO_CONTACTO	N	NUMBER	6	0	Codigo del contacto
	TELEFONO	N	CHAR	20		Telefono de la oficina del contacto
	DIRECCION1	Y	CHAR	60		Direccion de la oficina del contacto (1er renglon)
	DIRECCION2	Y	CHAR	60		Direccion de la oficina del contacto (2o renglon)
	FAX	Y	CHAR	20		Numero de fax de la oficina del contacto
	MODEM	Y	CHAR	20		Numero de modem de la oficina del contacto
	PUESTO	Y	CHAR	32		Posicion del contacto dentro de su compañia
CONTRATO	NO_CONTRATO	N	NUMBER	6	0	Identifica de forma unica a cada contrato
	ESTADO	Y	CHAR	16		Indica el estado actual del contrato (cancelado, por pagar, etc)
	FECHA_FIRMA	Y	DATE			Fecha en que fue firmado este contrato
	FECHA_RECONOCIMIENTO	Y	DATE			Fecha en que se da por reconocida la venta
	TERMINOS_PAGO	Y	CHAR	200		Terminos en los que se pagara el contrato
	TOTAL_A_PAGAR	Y	NUMBER	12	2	Cantidad total a pagar por productos y/o servicios contratados
	TOTAL_DOLARES	Y	NUMBER	8		
COSTO POR CREDITO	COSTO	N	NUMBER	12	2	Costo en pesos por credito educacional
	FECHA_FINAL	N	DATE			Indica fecha final de la vigencia
	FECHA_INICIO	N	DATE			Indica fecha de inicio de vigencia

Entities and their Attributes

Entity	Attribute Name	Optional	Format	Size	Dec Pl	Attribute Notes
CURSO NOMINAL	CREDITOS	N	NUMBER	1		Valor del curso contabilizado en creditos (por alumno)
	CVE_CURSO	N	CHAR	10		Identifica en forma unica a un curso
	NOMBRE	N	CHAR	32		Nombre del curso
CURSO PROGRAMADO	CUPO	N	NUMBER	2		Numero maximo de personas que pueden inscribirse al curso
	FECHA_FINAL	N	DATE			Fecha programada para finalizar el curso
	FECHA_INICIO	N	DATE			Fecha programada para comienzo del curso
	NO_CURSO_PROGRAMADO	N	NUMBER	6	0	Codigo del curso calendarizado
	CANCELADO	Y	CHAR	1		Bandera que indica la cancelacion o vigencia del curso
	HORARIO_FINAL	Y	CHAR	5		Horario de termino programado
	HORARIO_INICIO	Y	CHAR	5		Horario de comienzo programado
	LUGAR	Y	CHAR	32		Lugar (ciudad, direccion, local) donde se imparte el curso
EDUCACIONAL : Sub-type of TIPO CONTRATO	NO_CREDITOS	N	NUMBER	4		Numero de creditos adquiridos en este contrato
	NO_EDUCACIONAL	N	NUMBER	6	0	Numero del contrato asociado
	CREDITOS_RESTANTES	Y	NUMBER	4		Numero de creditos aun no utilizados en este contrato
EMPLEADO	CVE_EMPLEADO	N	CHAR	8		Clave del empleado que labora en la compa~ia Oracle de Mexico
	NOMBRE	N	CHAR	32		Nombre completo del empleado
	FECHA_EGRESO	Y	DATE			Fecha en que el empleado dejo de prestar sus servicios
	FECHA_INGRESO	Y	DATE			Fecha en la que se considera en nomina el empleado
	PUESTO	Y	CHAR	20		

Entities and their Attributes

Entity	Attribute Name	Optional	Format	Size	Dec Pl	Attribute Notes
ENCUESTA DE CURSOS	NO_ENCUESTA	N	NUMBER	4	0	Numero de encuesta en ese curso
	COMENTARIO	Y	CHAR	100		Punto de vista del asistente al curso
	NO_RESPUESTA	Y	NUMBER	2	0	Numero de respuesta asignada a la pregunta anterior
ENCUESTA DE SOPORTE	COMENTARIO	Y	CHAR	100		Punto de vista del cliente para esa pregunta
	RESPUESTA	Y	NUMBER	1		Numero de respuesta hecha por el cliente
FACTURA	FECHA_FACTURACION	N	DATE			Fecha en que se efectua la factura
	NO_FACTURA	N	NUMBER	6	0	Numero secuencial de factura
	TOTAL_DOLARES	N	NUMBER	8		Total de la factura en US dollars
	TOTAL_PESOS	N	NUMBER	12	2	Total de la factura en pesos m.n.
	CONCEPTO1	Y	CHAR	60		Descripcion breve del concepto por el que se factura
	CONCEPTO2	Y	CHAR	60		Descripcion breve del concepto por el que se factura
	DESCUENTO	Y	NUMBER	2		Descuento efectuado por Oracle (porcentaje)
	ESTADO	Y	CHAR	16		Indica el estado de la factura
	FECHA_PAGO	Y	DATE			Fecha en que se reconoce el pago
	IVA	Y	NUMBER	2		Impuesto al Valor Agregado causado (porcentaje)
	PECE	Y	NUMBER	2		Descuento hecho por PECE (porcentaje)
	VIATICOS	Y	NUMBER	12	2	Gasto de viaticos que se facturan al cliente
INFO ACCION	NO_RENGLON	N	NUMBER	3		Numero de renglon en la documentacion de la accion
	TEXTO	Y	CHAR	80		Linea de documentacion

Entities and their Attributes

Entity	Attribute Name	Optional	Format	Size	Dec Pl	Attribute Notes
INSCRIPCION	CREDITOS	N	NUMBER	2		Numero de creditos utilizados en esta inscripcion
	FECHA_INSCRIPCION	N	DATE			Fecha en que se efectua la inscripcion al curso
	ASISTENCIA	Y	NUMBER	2		Numero de dias que un alumno asiste a un curso
	CALIFICACION	Y	NUMBER	2	2	Calificacion 0.00 a 10.00 que el alumno recibe en el curso
	FECHA_CONFIRMACION	Y	DATE			Fecha en que se confirma la asistencia al curso
	FORMA_PAGO	Y	CHAR	1		Especifica la forma de pago (C=creditos y F=factura)
	NUMERO_FACTURA	Y	NUMBER	6	0	Bandera indicativa de si el curso ya fue facturado o no
LICENCIA : Sub-type of TIPO CONTRATO	NO_LICENCIA	N	NUMBER	6	0	Numero de contrato asociado a esta licencia
NIVEL DE PLATAFORMA	NIVEL	N	NUMBER	1		Nivel de responsabilidad que el analista tiene con la plataforma
NIVEL DE PRODUCTO	NIVEL	N	NUMBER	1		Nivel de responsabilidad que el analista tiene con el producto
PLATAFORMA	DESCRIPCION	N	CHAR	40		Nombre de la plataforma
	NO_PLATAFORMA	N	NUMBER	3		Codigo de la plataforma
PLATAFORMA CLIENTE	CVE_SERIE	N	CHAR	32		Numero de serie del equipo
	NO_SOPT_CLIE	N	NUMBER	6	0	Codigo que permite ofrecer soporte tecnico
	VERSION_SO	N	CHAR	20		Numero de version del sistema operativo
	MEDIO_DISTRIBUCION	Y	CHAR	20		Tipo de unidad en la que se entrega el software al cliente
PREGUNTA CURSOS	NO_PREGUNTA	N	NUMBER	2		Codigo de la pregunta
	PREGUNTA	N	CHAR	80		Texto de la pregunta

Entities and their Attributes

Entity	Attribute Name	Optional	Format	Size	Dec Pl	Attribute Notes
PREGUNTA SOPORTE	NO_PREGUNTA	N	NUMBER	2		Codigo de la pregunta
	PREGUNTA	N	CHAR	80		Texto de la pregunta
PRIORIDAD	DESCRIPCION	N	CHAR	32		Descripcion de la prioridad y su aplicabilidad
	NO_PRIORIDAD	N	NUMBER	2		Codigo de la prioridad que da un valor de emergencia a una SAT
PRODUCTO	DESCRIPCION	N	CHAR	32		Descripcion del producto Oracle
	NO_PRODUCTO	N	NUMBER	3		Codigo del producto Oracle
PRODUCTO CONTRATADO	NO_PARTIDA	N	NUMBER	6	0	Identifica a cada partida en forma unica
	COSTO_DOLARES	Y	NUMBER	8		Costo por dicho producto en dolares
	COSTO_PESOS	Y	NUMBER	12	2	Costo por dicho producto en pesos
	VERSION	Y	CHAR	20		Numero de version del producto
RAZON SOCIAL	NOMBRE	N	CHAR	60		Nombre de la razon social
	NO_RAZON_SOCIAL	N	NUMBER	6	0	Identifica a un grupo de clientes

Entity	Attribute Name	Optional	Format	Size	Dec Pl	Attribute Notes
SOLICITUD DE ASISIENCIA TECNICA	ENCABEZADO	N	CHAR	100		Linea abstracta de descripcion de la SAT
	FECHA_ACTUALIZACION	N	DATE			Fecha y hora de la ultima vez que se actualizo la SAT
	FECHA_CREACION	N	DATE			Fecha y hora en que se dio de alta la SAT
	NIVEL	N	NUMBER	1		Nivel de la SAT (1=tiempo real, 2=respuesta diferida)
	NOMBRE_CLIENTE	N	CHAR	32		Nombre de la empresa que reporta la SAT
	NOMBRE_CONTACTO	N	CHAR	32		Nombre del contacto tecnico
	NO_SAT	N	NUMBER	6	0	Numero de reporte de cada requisicion tecnica
	TELEFONO	N	CHAR	20		Numero telefonico completo del contacto tecnico
	COMENTARIO	Y	CHAR	100		Linea multi-proposito de la SAT
	CVE_ERROR	Y	CHAR	3		Codigo del producto en el cual el error aparece
	CVE_PCR	Y	CHAR	7		Clave del PCR (Bug) en USA asociado a esta SAT
	CVE_SERIE	Y	CHAR	32		Numero de serie del equipo sobre el que se trabaja
	CVE_TAR	Y	CHAR	11		Clave del TAR en USA asociado a esta SAT
	DIRECCION1	Y	CHAR	60		Direccion en donde se encuentra el equipo (1er renglon)
	DIRECCION2	Y	CHAR	60		Direccion en donde se encuentra el equipo (2o renglon)
	ESCALAMIENTO	Y	NUMBER	1		Numero de escalamiento actual de la SAT
	FAX	Y	CHAR	20		Numero telefonico de fax completo del contacto tecnico
	FECHA_ESCALAMIENTO	Y	DATE			Contiene la fecha del ultimo escalamiento de la SAT
	MEDIO_DISTRIBUCION	Y	CHAR	20		Indica cual es el tipo de DATA-MEDIA (cartucho,dat,tk,etc)
	MODEM	Y	CHAR	20		Numero telefonico de modem completo del equipo
	NO_ERROR	Y	NUMBER	6		Numero del error obtenido
	NO_SOPORTE	Y	NUMBER	6	0	Numero de soporte otorgado por contrato y por cpu

Entities and their Attributes

Entity	Attribute Name	Optional	Format	Size	Dec Pl	Attribute Notes
SOLICITUD DE ASISTENCIA TECNICA	VERSION_PRODUCTO	Y	CHAR	20		Numero de version del producto de Oracle
	VERSION_RDBMS	Y	CHAR	20		Numero de version del manejador de Oracle
	VERSION_SO	Y	CHAR	20		Numero de version del sistema operativo
SOPORTE : Sub-type of TIPO CONTRATO	FECHA_TERMINO	N	DATE			Fecha en que se da por termindao el contrato de soporte tecnico
	NO_SOPORTE	N	NUMBER	6	0	Numero del contrato asociado
STATUS DE LA SAT	DESCRIPCION	N	CHAR	32		Descripcion del status
	NO_STATUS	N	NUMBER	2		Codigo del estado que define a la SAT
TIPO CLIENTE	CVE_TIPO_CLIENTE	N	CHAR	2		Codigo del tipo del cliente
	DESCRIPCION_TIPO_CLIENTE	N	CHAR	32		Descripcion del tipo del cliente
TIPO CONTACTO	CVE_TIPO_CONTACTO	N	CHAR	2		Codigo del tipo de contacto
	DESCRIPCION	N	CHAR	32		Descripcion de las funciones del contacto
TIPO CONTRATO	MONTO_DOLARES	Y	NUMBER	8		Cantidad a pagar en dolares por toda el contrato
	MONTO_PESOS	Y	NUMBER	12	2	Cantidad en total a pagar por el concepto

CASE * Dictionary

ENTITIES AND THEIR ATTRIBUTES

End of Report

Attributes in a Domain

Domain	Entity Name	Attribute Name	Format	Length	Precision
CVE_CURSO	CURSO NOMINAL	CVE_CURSO	CHAR	10	
CVE_TIPO	TIPO CLIENTE	CVE_TIPO_CLIENTE	CHAR	2	
	TIPO CONTACTO	CVE_TIPO_CONTACTO	CHAR	2	
DOLARES	CONTRATO	TOTAL_DOLARES	NUMBER	8	
	FACTURA	TOTAL_DOLARES	NUMBER	8	
	PRODUCTO CONTRATADO	COSTO_DOLARES	NUMBER	8	
	TIPO CONTRATO	MONTO_DOLARES	NUMBER	8	
IDNUMERO2	AREA	NO_AREA	NUMBER	2	
	PRIORIDAD	NO_PRIORIDAD	NUMBER	2	
IDNUMERO3	PLATAFORMA	NO_PLATAFORMA	NUMBER	3	
	PRODUCTO	NO_PRODUCTO	NUMBER	3	
IDNUMERO6	ALUMNO	NO_ALUMNO	NUMBER	6	0
	CLIENTE	NO_CLIENTE	NUMBER	6	0
	CONSULTORIA	NO_CONSULTORIA	NUMBER	6	0
	CONTACTO	NO_CONTACTO	NUMBER	6	0
	CONTRATO	NO_CONTRATO	NUMBER	6	0
	CURSO PROGRAMADO	NO_CURSO_PROGRAMADO	NUMBER	6	0
	EDUCACIONAL	NO_EDUCACIONAL	NUMBER	6	0
	FACTURA	NO_FACTURA	NUMBER	6	0
	INSCRIPCION	NUMERO_FACTURA	NUMBER	6	0
	LICENCIA	NO_LICENCIA	NUMBER	6	0
	PLATAFORMA CLIENTE	NO_SOPT_CLIE	NUMBER	6	0
	PRODUCTO CONTRATADO	NO_PARTIDA	NUMBER	6	0
	RAZON SOCIAL	NO_RAZON_SOCIAL	NUMBER	6	0
	SOLICITUD DE ASISTENCIA T	NO_SAT	NUMBER	6	0
	SOLICITUD DE ASISTENCIA T	NO_SOPORTE	NUMBER	6	0
	SOPORTE	NO_SOPORTE	NUMBER	6	0
IDPERSONAL	EMPLEADO	CVE_EMPLEADO	CHAR	8	

CASE * Dictionary

Report : ATTRIBUTES IN A DOMAIN

Filename : cddom2.lis

Run by : TESIS

Report Date : 04-NOV-92 05:41pm

Parameter values
Application system : ORACLE
Version : 1

Attributes in a Domain

Domain	Entity Name	Attribute Name	Format	Length	Precision
PESOS	CONTRATO	TOTAL_A_PAGAR	NUMBER	12	2
	COSTO POR CREDITO	COSTO	NUMBER	12	2
	FACTURA	TOTAL_PESOS	NUMBER	12	2
	FACTURA	VIATICOS	NUMBER	12	2
	PRODUCTO CONTRATADO	COSTO_PESOS	NUMBER	12	2
	TIPO CONTRATO	MONTO_PESOS	NUMBER	12	2

CASE * Dictionary

ATTRIBUTES IN A DOMAIN

End of Report

CASE * Dictionary

Report : UNIQUE IDENTIFIERS FOR AN ENTITY

Filename : cduie1.lis

Run by : TESIS

Report Date : 04-NOV-92 05:27pm

Parameter values

Application System : ORACLE
Version : 1

Unique Identifiers for an Entity

Entity	Unique Id	Attribute Name	Relationship	To Entity
*ACCION	ACC1	NO_ACCION	PARTE DE	SOLICITUD DE ASISTENCIA TECNICA
*ALUMNO	ALU1	NO_ALUMNO		
*AREA	ARE1	NO_AREA		
*CLIENTE	CLI1	NO_CLIENTE		
*CONSULTORIA	CON1	NO_CONSULTORIA		
*CONTACTO	CONT1	NO_CONTACTO		
*CONTRATO	CONTR1	NO_CONTRATO		
*COSTO POR CREDITO	CPC1	FECHA_FINAL FECHA_INICIO		
*CURSO NOMINAL	CNO1	CVE_CURSO		
*CURSO PROGRAMADO	CPR1	NO_CURSO_PROGRAMADO		
*EDUCACIONAL	EDU1	NO_EDUCACIONAL		
*EMPLEADO	EMP1	CVE_EMPLEADO		
*ENCUESTA DE CURSOS	EDC1	NO_ENCUESTA	CALIFICANDO	CURSO PROGRAMADO
			RESPONDIENDO	PREGUNTA CURSOS

(*) Denotes Application owns Entity

Unique Identifiers for an Entity

Entity	Unique Id	Attribute Name	Relationship	To Entity
*ENCUESTA DE SOPORTE	ESO1		CALIFICANDO	SOLICITUD DE ASISTENCIA TECNICA
			RESPONDIENDO	PREGUNTA SOPORTE
			CALIFICANDO	EMPLEADO
*FACTURA	FAC1	NO_FACTURA		
*INFO ACCION	IFA1		PARTE DE	ACCION
		NO_RENGLON		
*INSCRIPCION	INS1		PARA	ALUMNO
			PARA	CURSO PROGRAMADO
*LICENCIA	LIC1	NO_LICENCIA		
*NIVEL DE PLATAFORMA	NDP1		PARA	PLATAFORMA
			PARA	EMPLEADO
*NIVEL DE PRODUCTO	NDPR1		PARA	PRODUCTO
			PARA	EMPLEADO
*PLATAFORMA	PLA1	NO_PLATAFORMA		
*PLATAFORMA CLIENTE	PCL1	NO_SOPT_CLIE		
*PREGUNTA CURSOS	PCU1	NO_PREGUNTA		
*PREGUNTA SOPORTE	PSO1	NO_PREGUNTA		
*PRIORIDAD	PRI1	NO_PRIORIDAD		
*PRODUCTO	PRO1	NO_PRODUCTO		

(*) Denotes Application owns Entity

Unique Identifiers for an Entity

Entity	Unique Id	Attribute Name	Relationship	To Entity
*PRODUCTO CONTRATADO	PCO1	NO_PARTIDA	SOPORTADO EN	PLATAFORMA CLIENTE
*RAZON SOCIAL	RSO1	NO_RAZON_SOCIAL		
*SOLICITUD DE ASISTENCIA TECNICA	SAT1	NO_SAT		
*SOPORTE	SOP1	NO_SOPORTE		
*STATUS DE LA SAT	STA1	NO_STATUS		
*TIPO CLIENTE	TCL1	CVE_TIPO_CLIENTE		
*TIPO CONTACTO	TCO1	CVE_TIPO_CONTACTO		

(*) Denotes Application owns Entity

CASE * Dictionary

UNIQUE IDENTIFIERS FOR AN ENTITY

End of Report

2. HERRAMIENTAS CASE DE DESARROLLO Y ETAPA DE DISEÑO

HERRAMIENTAS CASE DE DESARROLLO Y ETAPA DE DISEÑO

El método de desarrollo de sistemas CASE*Method es independiente de las herramientas que se han de utilizar en la construcción del sistema tales como diagramadores, generadores de código, base de datos, etc.

En este capítulo se presentan algunas de las herramientas disponibles para el desarrollo de sistemas y se lleva a cabo el estudio comparativo de las mismas.

Se utilizará *CASE Oracle* en el procesamiento de la información obtenida en el capítulo anterior (diagramas funcionales, diagrama entidad relación, etc.) con el fin de obtener las estructuras lógica y física del sistema.

Así, en el tema 2.1 "HERRAMIENTAS CASE DE DESARROLLO" se describen algunas de las características que deben cumplir dichas herramientas, y se hace la comparación de las existentes en el mercado.

En el tema 2.2 "CASE ORACLE EN EL DISEÑO DEL SISTEMA" se detallan los pasos que se siguen dentro de la herramienta CASE ORACLE para cubrir la fase de diseño del sistema. Y por último en el tema 2.3 "RESULTADOS OBTENIDOS EN ESTE CAPITULO" se presenta un resumen de toda la información obtenida durante la etapa de diseño y que servirá como entrada pera la etapa de implementación.

2.1 HERRAMIENTAS *CASE* DE DESARROLLO

HERRAMIENTAS DE DESARROLLO

Herramientas de Análisis

Dentro del conjunto de herramientas que conforman un ambiente de análisis se encuentra un lenguaje de propósito específico, que sirve como interfaz entre usuario y un procesador para hacer un análisis automatizado de los requerimientos del sistema.

Existen tres objetivos básicos que debe cumplir una herramienta automática enfocada al análisis de sistemas:

- **Incremento en la productividad:** Por medio del soporte de una o más metodologías estructuradas, además de permitir dibujar y modificar los Diagramas de Flujo de Datos (DFD's: Data Flow Diagram) fácilmente, permite controlar las modificaciones.
- **Mejoramiento de la comunicación entre el analista y el usuario.**
- **Integración de las actividades del ciclo de vida.**

Herramientas de Diseño

Las especificaciones funcionales de las cuales se origina el diseño puede ser una gráfica, o una sucesión de gráficas. El diseño por computadora tiene la ventaja de que puede ajustarse a los cambios que se den por parte de los usuarios y los programas obtenidos a través de este diseño pueden volverse a generar sin mucha dificultad.

El diseño de sistemas nunca podrá ser totalmente automatizado. La inventiva y creatividad humana son el aspecto más importante de la fase de diseño.

Existen una serie de propiedades que se consideran deseables para una herramienta de especificación orientada al diseño, las cuales pueden resumirse en los siguientes puntos:

- Debe mejorar la claridad conceptual del sistema.
- Debe ser fácil de aprender y de usar.

- Debe ser muy precisa para que el código pueda generarse automáticamente a partir de ésta.

- Debe estar diseñada para automatizar al máximo el análisis, el diseño y la programación de sistemas.

- Debe ser rigurosa para obtener diseños correctos y consistentes.

- Debe ser muy versátil, de forma que los lenguajes de programación no sean utilizados en forma manual.

- Debe ser lo suficientemente completa para la creación de todo un sistema sin importar su complejidad.

- Debe emplear una jerarquía que descienda hacia niveles de mayor detalle.

- Debe propagar automáticamente los cambios hechos en módulos de niveles inferiores.

- Debe basarse en el uso de librerías que contengan subrutinas, programas, y todas las construcciones que emplee el lenguaje.

- Debe unirse en forma automática a herramientas para el manejo de base de datos, por ejemplo: diccionario de datos.

- Debe garantizar la consistencia en las interfaces cuando se utilizan llamadas a subrutinas o programas.

- El lenguaje utilizado debe ser independiente del equipo y de otros recursos que puedan cambiar.

Las ventajas que aporta el uso de técnicas automatizadas en el desarrollo de sistemas son:

- El desarrollo se realiza en forma más rápida.

- Se obtienen grandes reducciones en el costo de desarrollo.

- La programación se vuelve automática.

- La mayoría de los errores son detectados antes de la implantación.

- Cada nivel del diseño es una expansión del nivel previo.

- Se fuerza una especificación consistente y completa.
- Interfaces rigurosas y susceptibles de comprobación entre los subsistemas.
- Garantía de integridad después de la implantación.
- Existen equipos de desarrollo pequeños, o formados por una sola persona, por lo tanto hay menos problemas de comunicación.
- Los mecanismos de librería facilitan la construcción de módulos reutilizables.
- El mantenimiento se simplifica.
- El código de alta calidad se vuelto a generar después de cada cambio.
- Los diseños son portables a diferentes máquinas.

INGENIERIA DE *SOFTWARE* ASISTIDO POR COMPUTADORA (*CASE*)

Cuando se integran las herramientas de forma tal que la información creada por una herramienta pueda ser usada por otra, se establece un sistema para el soporte del desarrollo de *software* llamado Ingeniería de *software* Asistido por Computadora (*CASE*: Computer Aided Systems Engineering). *CASE* combina el *software*, *hardware* y bases de datos para crear una estructura de datos que contenga la información relevante sobre el análisis, diseño, codificación y prueba de un sistema.

CASE es una tecnología que provee ayuda computarizada para cualquier persona (usuarios y programadores) que trabaje con *software*.

***CASE*: Primera Generación**

CASE fue presentado en Abril de 1984, los primeros productos *CASE* automatizaban, por medio de *software*, el tiempo que requerían las tareas de análisis, diseño y programación, es decir, computarizaban las tareas de determinación de requerimientos, diseño de soluciones y construcción del sistema.

Las herramientas *CASE* facilitaron y agilizaron el uso de las técnicas de programación estructurada, que son esenciales en el

conocimiento de la ingeniería de *software*. Con *CASE* se puede dibujar, modificar, editar y almacenar varios diagramas utilizados en las fases iniciales del ciclo de vida del desarrollo de sistemas. La computadora se convirtió en una herramienta que verificaba la precisión de los diagramas. Estos productos se conocieron como herramientas *CASE* - fase inicial (*Upper-CASE*).

En contraste, las herramientas *CASE* - fase final (*Lower-CASE*), computarizan las tareas en las fases de: generación de código, pruebas, compilación, generación de prototipos e implementación.

Las herramientas *CASE* disponibles asistían las tareas de todas las fases del ciclo de vida del *software*; sin embargo, los datos e información no necesariamente fluían automáticamente de una fase a otra.

***CASE* : Segunda Generación**

La segunda generación de herramientas *CASE* surge entre 1985 y 1986, con la introducción de productos que soportaban tareas en todas las fases del ciclo de vida. Las herramientas proporcionaron una integración de requerimientos, especificaciones, diseño, código, pruebas, implementación y mantenimiento.

Se agregaron técnicas de programación en tiempo real y se probó que el código podía ser generado directamente de los diagramas. Además, se descubrió que los sistemas expertos, y las bases del conocimiento del campo de inteligencia artificial (IA), podían reducir el ciclo de desarrollo. Con IA, los diagramas pueden reemplazarse por modelos lógicos, desde los cuales pueden generarse automáticamente sistemas de tiempo real. La IA puede verificar diagramas extensos y complejos y darle significado al código existente.

Pre-CASE, se refiere a herramientas que interactuan con las cuestiones estratégicas y administrativas del desarrollo de *software*. Representan las herramientas de planeación para que los usuarios identifiquen y evalúen el alcance de los proyectos de *software*. Los productos *Pre-CASE* incluyen herramientas para la planeación estratégica, como las utilizadas para desarrollar el modelo de datos corporativo, agrupar datos en sistemas de información y catalogar estos sistemas de acuerdo a su impacto en las misiones de la corporación. Las funciones administrativas que soportan las herramientas *Pre-CASE* incluyen la planeación de proyectos, calendarización y estimación (*ver figura 2.1.1*).

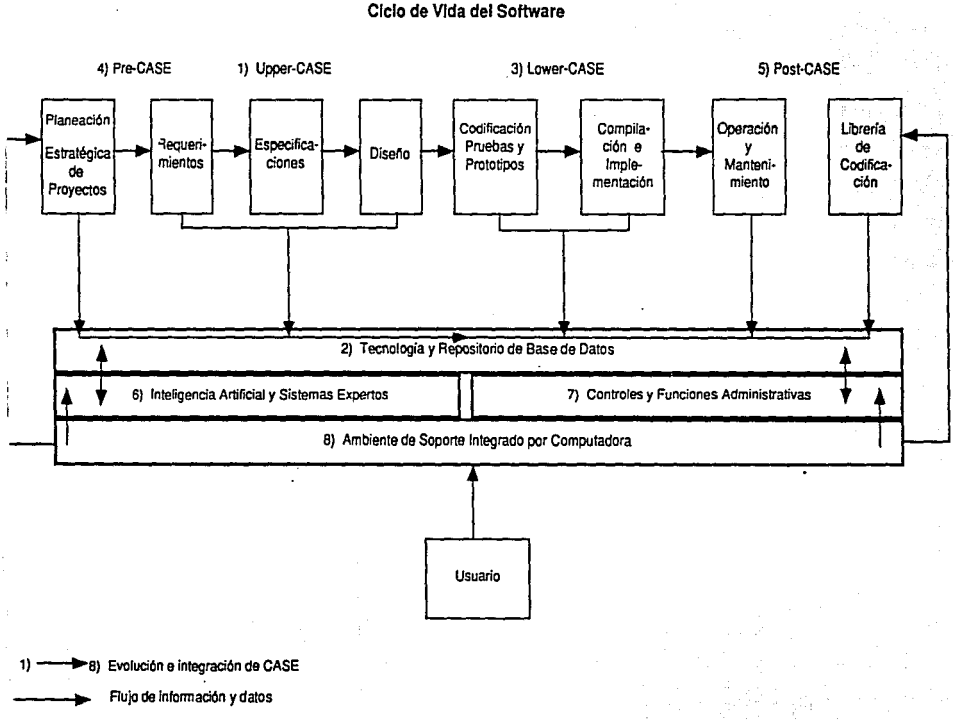


Figura 2.1.1 Ciclo de Vida del Software y fases de la metodología CASE

Con la tecnología de bases de datos establecida como parte de la estructura de conocimiento de *CASE*, los desarrolladores necesitan herramientas para transferir código existente al ambiente *CASE*. El código estructurado puede ser almacenado para reusarse como módulos de futuros programas, pero los programas mal estructurados no se ajustan bien en el ambiente *CASE*, (Ingeniería *botton-up*). Este es el proceso *'Post-CASE'* para convertir código no estructurado en código estructurado y colocarlo en el almacén *CASE*, donde está disponible para mantenimiento, edición, o reuso por medio de las herramientas *CASE*.

CASE EN COMPAÑIAS INDEPENDIENTES

En el capítulo anterior revisamos algunos de los métodos más usados para el desarrollo de sistemas, después del análisis comparativo hemos concluido que CASE*Method es la mejor opción para el desarrollo de sistemas de bases de datos relacionales. A continuación se mencionarán algunas firmas de *software* que proveen de productos para apoyar en el desarrollo de sistemas, los cuales están basadas en el método CASE.

El crecimiento del mercado de *CASE* ha ocasionado el surgimiento de muchas pequeñas firmas, dedicadas solamente al mercado de *CASE*. Los líderes de esta categoría son:

Compañía	Producto
Index Technology	Excelerator
Nastec	*CASE* 2000
KnowledgeWare	Information Engineering Workbench(IEW)
Oracle	*CASE**ORACLE

Por su naturaleza y dedicación a *CASE*, estos vendedores ofrecen actualmente los productos más completos del mercado. Muchas de estas compañías desarrollan interfaces y unen éstas para proporcionar un producto más completo.

Existen otras compañías que venden generadores de códigos para apoyar el desarrollo de sistemas, usualmente estas compañías son más pequeñas. En general, estos productos generan código en COBOL para "mainframes". Las compañías líderes en este campo son:

Compañía	Producto
Panasophic Systems	Telon
Sage Software	APS
CGI Systems	PacBase
KnowledgeWare	Gamma

Los productos *CASE* están ganando mucha popularidad en el área de desarrollo de productos de *software* y sistemas, debido al inmenso apoyo que ofrecen durante todo el desarrollo del sistema.

Los clientes de compañías de *hardware* están considerando cada vez más la disponibilidad de las herramientas *CASE* cuando ellos eligen un *hardware*. Para proporcionar esta facilidad algunos vendedores han hecho alianzas con vendedores de *CASE*. La desventaja de esta asociación es que se deja una mezcla de productos en un máquina, tanto externos como internos. Mientras estos productos generalmente tienen alguna forma de interfaz, ellos están raramente integrados completamente.

Las compañías que venden metodologías y consultoría han encontrado una ruta de migración natural dentro de las herramientas *CASE*. Estas compañías incluyen:

- Yourdon (Analyst/Designer toolkit)
- Ken Orr & Associates (DesignMachine)
- Peter Chen (ER-Designer, Normalizer)
- James Martin & Associates (quien está asociado con otros vendedores, entre otros Texas Instruments y KnowledgeWare).

Como es de esperarse , los productos *CASE* en esta categoría a menudo siguen rigurosamente la metodología de su compañía padre, ellos son difíciles de usar con otras metodologías. La fuerte liga entre *software* y metodología puede ser de gran ayuda para clientes actuales, pero también pude ser una desventaja potencial con clientes nuevos.

El segmento más grande del mercado es el de los sistemas para negocios. Los sistemas que necesitan ser desarrollados son a menudo caracterizados por:

- Usuarios múltiples, muy a menudo con conflictos de requerimientos y prioridades complejos.
- Definiciones ambiguas del problema y ambientes mal definidos.

CATEGORIAS DEL MERCADO *CASE*

Case IEW (Information Engineering Workbench)

El IEW soporta las siguientes técnicas:

- Diagrama entidad relación
- Diagrama de descomposición
- Diagrama de flujo de datos
- Diagrama de acción

El IEW está compuesto de cuatro herramientas basadas en estaciones de trabajo:

- IEW/Planeación
- IEW/Análisis
- IEW/Diseño
- IEW/Construcción

Cada estación de trabajo incorpora una enciclopedia al lugar donde la información del análisis, diseño y sus relaciones está almacenada. La enciclopedia no proporciona capacidad directa multi-usuario.

La integridad del contenido de la enciclopedia es mantenida por el coordinador de conocimiento, un sistema experto, que ofrece consistencia entre diagramas, y automáticamente transforma diagramas de un tipo en otro.

Las herramientas trabajan sólo sobre Sistemas Personales IBM Modelo 50 o posterior o en una Computador Personal AT IBM bajo el sistema operativo DOS 3.1, o posterior, y sobre una IBM 3270/AT.

IEW/Planeación

La estación de trabajo Planeación es usada para organizar, modelar y analizar los datos sobre una organización y la forma en que ésta utiliza la información.

La estación de trabajo Planeación proporciona las siguientes herramientas basadas en diagramas:

- Diagrama de Descomposición: Define las asociaciones jerárquicas de planeación de objetos en niveles sucesivos de detalle.

- Diagrama de entidades: Describe los requerimientos de datos de la organización y la relación entre estos.

- Matriz de Asociación: Documenta los requerimientos de información y las relaciones entre la fuente y el uso de la información.

- Matriz de propiedad: Asigna y revisa características de objetos.

IEW/Análisis

La estación de trabajo análisis permite a los analistas de sistemas, analizar gráficamente los requerimientos del usuario final y crea la especificación del sistema. Proporciona las siguientes herramientas basadas en diagramas:

- Diagrama de descomposición: Refina los datos y procesos dentro de sus partes componentes, en niveles sucesivos de detalle.

- Diagrama de flujo de datos: Describe como fluyen los datos dentro y fuera del sistema y como son transformados.

- Diagrama de entidades: Describe los requerimientos de datos y como se relacionan.

- Diagrama de acciones: Define procesos y procedimientos.

IEW/Diseño

La estación de trabajo diseño permite a los diseñadores de *software*, diseñar gráficamente sistemas de información y crear especificaciones físicas para datos y procesos. Esto ayuda a los diseñadores a cambiar de una representación lógica de lo que un sistema dado hace, a la especificación física de como el sistema será construido.

La estación de trabajo diseño proporciona las siguientes herramientas basadas en diagramas:

- Diagrama de carta estructurada: Proporciona un vista de la estructura modular de un sistema.

- Diagrama de acción: Crea y ve la lógica de un programa a algún nivel de detalle.

- Diagrama de presentación por pantallas: Proporciona una pantalla para edición de funciones.

- Diagrama para bases de datos jerárquicas: Define la estructura de segmentos para una base de datos jerárquica.

- Diagrama para bases de datos relacionales: Da una vista de todas las relaciones en una base de datos relacional.

- Diagrama para bases de datos formadas por archivos planos: Representa las relaciones entre registros en un archivo plano.

IEW/Construcción

La estación de trabajo construcción usa los requerimientos y las especificaciones de diseño creadas con la estación de trabajo diseño para generar código en COBOL para ambientes IBM MVS.

La estación de trabajo Construcción ejecuta las mismas funciones que IEW/GAMMA, de acuerdo a *KnowledgeWare*, el código generado por los dos productos es idéntico.

IEW/GAMMA

IEW/GAMMA es una herramienta basada en sistemas "*mainframe*" para el procesamiento de la información que ha sido coleccionada en la enciclopedia con la metodología apropiada (IEW) y las herramientas basadas en PCs. IEW/GAMMA genera código COBOL ANSI estándar o código fuente COBOL II. Este código puede ser compilado y ejecutado en los sistemas operativos IBM MVS, MVS/SP o MVS/XA.

Esta herramienta puede ser usada directamente por la estación de diseño o independientemente utilizando una interfaz 3270. IEW/GAMMA genera código fuente, rutinas de acceso a los archivos de la base de datos, esquemas de la base de datos y el lenguaje de definición de la base de datos. También crea las descripciones de los archivos para COBOL, los mapas de pantalla y documentación.

IEW/GAMMA también hace mucho del trabajo requerido para una aplicación en un ambiente específico. Estos incluyen DB2, IMS/DL/1 VSAM, bases de datos IDMS, monitores TP tales como CICS, IMS/DC y MS/DC/DB2.

CASE ORACLE

La familia de productos *CASE* ORACLE proporcionan un diccionario de datos sobre las aplicaciones y un taller gráfico integrado y avanzado, enfocado al desarrollo de sistemas estructurados, que ayuda y guía a través de todas las etapas del ciclo de vida de desarrollo del sistema.

La tecnología *CASE* combina el conocimiento de los negocios con el poder de los recursos computacionales para transformar los problemas del mundo real en soluciones prácticas. De la misma manera en que la demanda por aplicaciones ha producido históricamente soluciones aisladas y no integradas, muchos productos *CASE* han tenido en cuenta sólo componentes individuales de soluciones al problema generado por la demanda de aplicaciones.

El ambiente de *CASE* de ORACLE es de desarrollo completo, contiene una base de datos que almacena y organiza los requerimientos de los negocios y las especificaciones de las aplicaciones. Es un diccionario compartido que almacena una imagen completa de las necesidades de información, requerimientos funcionales, fuentes de datos y programas.

La familia *CASE* de ORACLE proporciona la habilidad de desarrollar rápidamente sistemas que cumplan con las necesidades de la organización. Ayudan a identificar y articular los requerimientos de los usuarios de manera limpia y consciente, y ayuda a desarrollar sistemas que cumplan con esas necesidades. El resultado son sistemas apropiados y oportunos que requieren menos mantenimiento, mejoran la comunicación, aumentan la calidad e incrementan la productividad en los usuarios.

CASE*Method es soportado por la familia de productos *CASE* de Oracle:

- CASE*Dictionary
- CASE*Designer
- CASE*Generator

Case*Dictionary

CASE*Dictionary es parte de la familia integrada *CASE* Oracle que provee de herramientas y métodos para guiar el desarrollo a través de todo el ciclo de vida del sistema. CASE*Dictionary es un almacén multiusuario, y portable, que ayuda en la organización y definición de la información generada en cada etapa del ciclo de vida del sistema. Con las herramientas de CASE*Dictionary para el análisis, diseño e implantación, se pueden especificar sistemas con precisión y sin ambigüedades, eliminando errores durante el proceso de desarrollo.

CASE*Dictionary es una base de datos multiusuario, distribuida y compartida que almacena toda la información y necesidades funcionales de una organización, junto con sus decisiones de diseño y detalles de implementación. Ayuda a recopilar y administrar grandes cantidades de información acerca de los sistemas con los que se está trabajando: lo que el sistema debe hacer y la manera en que debe hacerlo.

CASE*Dictionary soporta totalmente a CASE*Method y otros métodos de desarrollo comunes. Este soporte permite usar un método estructurado y después organizar, verificar y obtener conclusiones.

Permite técnicas de diseño múltiples, incluyendo Entidad-Relación, Funciones jerárquicas y de flujo de Datos, que ayudan a entender y definir las necesidades de las aplicaciones.

Una vez determinados los requerimientos de los usuarios, CASE*Dictionary ayuda a diseñar las aplicaciones adecuadas rápida y eficientemente. Propone y genera automáticamente un diseño de bases de datos y estima el tamaño que tendrá. Este diseño se puede afinar y ajustar dependiendo de las necesidades de la empresa.

Se puede agregar información a través de la interfaz que presenta CASE*Dictionary por medio de formas o utilizando las herramientas de diagramación incluidas en CASE*Designer.

La base de datos multiusuario de CASE*Dictionary permite el acceso o edición de información de manera concurrente a diferentes miembros del equipo de trabajo. El Sistema Manejador de la Base de datos (*DBMS* Data Base Management System) relacional de Oracle proporciona control de concurrencia, elimina la necesidad tediosa de utilizar procedimientos para la carga y descarga de información. Las facilidades de control de versión y seguridad permiten a los

usuarios accesar sólo la información que están autorizados a ver, mientras que simultáneamente se lleva un control de múltiples versiones del sistema.

CASE*Dictionary genera alrededor de 70 reportes, incluyendo referencias cruzadas, impacto del análisis, y reportes de excepción. De esta manera, se puede documentar precisamente lo que hacen los sistemas y como lo hacen.

CASE*Dictionary genera automáticamente un diseño de datos normalizado a partir de modelos de información de alto nivel. Se puede afinar este diseño y después CASE*Dictionary crea automáticamente la base de datos, la cual puede ser en ORACLE o en DB2.

CASE*Dictionary corre en casi cualquier plataforma de computadoras, incluyendo ambientes distribuidos. Se tiene la flexibilidad de construir y mantener aplicaciones exitosas a través de una amplia gama de sistemas operativos y de *hardware*.

Características Principales:

Modelo de Información

- Modelo Entidad Relación.
- Relaciones uno a uno, uno a muchos y muchos a muchos.
- Relaciones opcionales u obligatorias (no nulas).
- Relaciones exclusivas.
- Atributos.
- Dominios.
- Identificadores únicos.
- Estimados de volumen y rangos de crecimiento.
- Subgrupos y supergrupos en una entidad.
- Asegura la unicidad en el nombramiento de entidades.
- Sinónimos ilimitados.

Descomposición Funcional

- Jerarquía funcional.
- Modelo de evento.
- Dependencia entre funciones.
- Frecuencias.
- Funciones comunes entre aplicaciones.

Modelo de Flujo de Datos

- Flujos de datos de varios niveles.
- Balanceo automático.
- Integración del modelo entidad relación con los flujos de datos y almacenamiento de datos.

Modelo de Datos Físico

- Columnas y tablas de la base de datos.
- Definiciones de índices.
- Definiciones de espacios.
- Presentación de archivos y registros.

Referencias Cruzadas

- Crea y mantiene referencias cruzadas entre los objetos del diccionario.
- Lista parcial de los objetos con referencias cruzadas:
  Entidades y Atributos
  Funciones de Negocios
  Unidades de Negocio
  Tablas y Columnas de la Base de Datos
  Módulos de Programas

Utilerías

- Diseño de bases de datos por *default*:
  Estructura de base de datos normalizada, basada en la información del modelo.
- Diseño de índices por *default* para optimizar el rendimiento.
- Medición de la base de datos.
- Soporte de Ingeniería *bottom-up*.
- Analizador de léxico para generar matrices por *default*.

Generador de Reportes

- Genera alrededor de 70 reportes predefinidos.
- Referencias Cruzadas.
- Reportes sobre impacto del análisis.
- Posibilidad de diseñar reportes usando SQL*ReportWriter.

Diccionario De_llamado/Extracción

- Mover definiciones entre diccionarios.
- Respaldar/ Restaurar aplicaciones.

Creación y Mantenimiento de la Base de Datos

- Creación de base de datos Oracle o DB2.
- Definiciones de tablas, vistas, Indices, y espacios.
- Genera instrucciones para modificar la base de datos.

Seguridad y control

- Definiciones agrupadas por aplicación.
- Las aplicaciones pueden compartir definiciones.
- Se puede tener diferentes niveles de acceso a las aplicaciones para diferentes desarrolladores:

  Sin acceso, Solo lectura, Modificar,
  Insertar, Borrar, Compartir

Control de Versión

- Número ilimitado de versiones.
- Congelamiento opcional de una versión.
- Comparación de versiones.

Case*Designer

CASE*Designer es un medio ambiente de desarrollo gráfico multi-tareas y multi-ventanas que proporciona un juego de herramientas de desarrollo completo, las cuales ayudan a registrar en el CASE*Dictionary diferentes conceptos surgidos durante el análisis, diseño y desarrollo de un sistema, éstas son:

- Diagramador de jerarquía funcional. El análisis funcional consiste en identificar qué se ha hecho en el negocio, cuánto se ha hecho y qué necesidades se tienen para el futuro.

- Diagramador de entidades y relaciones. Permite la creación y modificación rápida de los modelos de información que más tarde van a ser usados para generar automáticamente su diseño de base de datos.

- Diagramador de diagrama de flujo de datos. El diagrama de flujo de datos es un modelo de cómo fluye la información dentro de un negocio y a través de su frontera con el mundo real. Los diagramas de flujo de datos de alto nivel son muy útiles para mostrar el alcance del sistema y las interfaces con otros sistemas. Los diagramas de flujo de datos de bajo

nivel son usados para validar el modelo de datos y la jerarquía de funciones.

- Diagramador de matrices de comprobación. Su objetivo es tener la posibilidad de resaltar cualquier inconsistencia y/o falta de información en el modelo. Se usan principalmente para verificar que las funciones sólo se refieran a entidades y atributos del modelo de entidades, que las entidades tengan una función que las cree, las modifique, las borre, las consulte, etc.

CASE*Designer permite a los equipos de desarrollo crear y mantener comprensivamente, modelos gráficos de los sistemas que ellos construyen. CASE*Designer ofrece un ambiente multi-ventana y soporta las siguientes técnicas basadas en diagramas:

- Diagrama Entidad Relación
- Diagrama de Jerarquía de funciones
- Diagrama de flujo de datos
- Diagrama de matriz

CASE*Designer permite a los usuarios agregar y editar información del diccionario (CASE*Dictionary), por medio de diagramas. Los diagramas son reflejados en CASE*Dictionary en tiempo real.

CASE*Generator para SQL*Forms

CASE*Generator para SQL*Forms permite generar aplicaciones a partir de las especificaciones de diseño en CASE*Dictionary. CASE*Generator para SQL*Forms ofrece las siguientes características:

- Genera formas basado en las definiciones sobre CASE*Dictionary.
- Cumple con las restricciones ANSI estándar para la integridad referencial.
- Incluye mensajes de ayuda y un rango de validaciones en las formas.
- Genera y valida llaves primarias.
- Crea soluciones portables sin importar el *hardware* de desarrollo.

En la *figura 2.1.2* se muestra la relación entre el ciclo de vida de un sistema y las herramientas *CASE* ORACLE de desarrollo.

CASE * Method
Enfoque estructurado para el desarrollo de sistemas

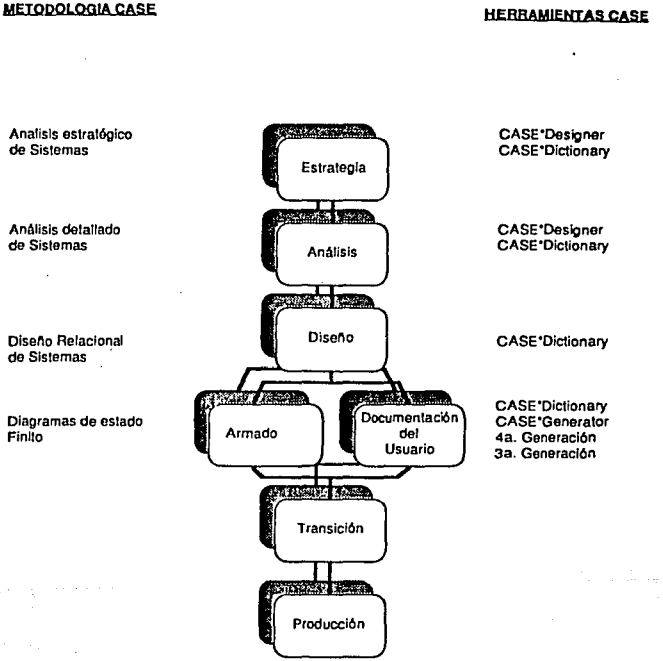


Figura 2.1.2 Relación entre Metodología CASE y herramientas CASE ORACLE

Evaluación de las Herramientas *CASE*

IEW

Ventajas

- Soporta todo el ciclo de vida de un sistema.
- Acercamiento estructurado al desarrollo de sistemas.
- Interfaz gráfica basada en GEM.
- Soporta color. El usuario puede jugar con el color de los diagramas y hacer tipos de objetos en la pantalla de algún color.
- Soporta diagramas de acción para documentar funciones.
- Habilidad para manejar múltiples construcciones de bases de datos: jerárquica, relacional.

Desventajas

- La enciclopedia no es multi-usuario.
- La enciclopedia no es muy completa. Los valores de volúmenes, frecuencias, formatos y dominios no están disponibles.
- No posee un control de versión. La versión es parcialmente soportada por medio de las facilidades "Copy/Rename".
- La capacidad de reportes no es muy extensa

Desventajas de la interfaz de usuario

- El IEW no permite objetos de tamaño variable en el mismo diagrama ni varia el tamaño de una entidad.
- Existe un límite (limitación de windows DOS) en el número de objetos que pueden ser desplegados en la pantalla.
- Existe un límite en la cantidad de texto desplegado en los objetos. Los usuarios deben seguir la regla de 9s, menos de nueve carácteres en una palabra para hacer diagramas entendibles.

Desventajas en las herramientas de modelado.

- No hay una implementación directa de subtipos.
- El formato del diagrama de descomposición de funciones es estrictamente horizontal.

CASE ORACLE

Ventajas

- Tienen las capacidades que son inherentes a los productos ORACLE: portabilidad, capacidad multi-usuario y conectividad.
- Corre en una gran variedad de máquinas.
- Cubre por completo el ciclo de vida de un sistema.
- Consta de una interfaz gráfica para desarrollo.
- Cuenta con una herramienta generadora de código para usarse con ORACLE o con DB2.
- Se pueden almacenar varias versiones de un diagrama.

Desventajas

- No soporta diagramas de control de flujo.
- No genera código para aplicaciones en tiempo real.
- Se basa en construcciones para bases de datos puramente relacionales.
- Existe un límite en la cantidad de objetos que se despliegan en pantalla.

Los productos de *CASE* que se emplearán para el desarrollo de esta tesis son los productos de *CASE* de Oracle, pues las necesidades de información que se tienen se adaptan completamente a un modelo de bases de datos relacional y los productos *CASE* de Oracle están basados en este tipo de modelos. Otra ventaja importante es que corren en una gran variedad de máquinas, es decir el sistema que se genere no estará sujeto a ningún tipo de máquina en particular.

Las herramientas IEW son también un excelente método para el desarrollo de sistemas; sin embargo, se muestran ya un poco obsoletas y tienen la gran desventaja de funcionar solamente en equipos IBM, si surgiera la necesidad de migrar de máquina, el sistema tendría que ser reconstruido totalmente, a diferencia de un sistema construido con herramientas *CASE* ORACLE, en donde únicamente se migran los datos y se recompilan las formas sobre las cuales esté basado el sistema.

Ambos tipos de herramienta cubren de principio a fin el ciclo de vida de un sistema, pero los productos ORACLE ofrecen interfaces al usuario mas fáciles de manejar. Además, constan de más procedimientos de verificación del modelo de datos, con lo cual se puede garantizar que el sistema cubrirá completamente las necesidades de información que se modelen.

2.2 ORACLE *CASE* EN EL DESARROLLO DEL SISTEMA

A continuación se detallan algunas de las tareas realizadas durante la etapa del diseño.

Diseño de la aplicación

Dentro de esta tarea las funciones son transformadas en módulos y se detallan las características de estos módulos.

Se revisan las funciones elementales, y se hace una matriz función/módulo para tener la seguridad de que todas las funciones necesarias han sido implementadas.

Se determina la forma en la cual cada uno de los módulos será implementado y la estructura de los mismos (incluyendo procedimientos manuales).

Se diseña la arquitectura del sistema:

- Estructura de los menúes.
- Pantallas de ayuda.
- Procedimientos *batch*.
- Procedimientos manuales.
- Interfaces de usuario y descripción de pantallas, reportes y formas.

Para cada módulo se define y se documenta lo siguiente:

- Detalle del procesamiento, definiendo como la herramienta que servirá para la implementación del módulo será utilizada para la realización de cada uno de los aspectos de la función definida.

- Uso de entidades y atributos.
- Procedimientos de actualización.
- Procedimientos de validación y control.
- Procedimientos especiales.

Se hace una revisión entre las especificaciones de los módulos y el diagrama entidad-relación y se produce una matriz inicial módulo(programa)/tabla para asegurar que todos los aspectos hayan sido cubiertos.

Diseño y construcción de la base de datos

En esta tarea las entidades son convertidas en tablas y archivos, y se gener a los índices y vistas necesarios.

Dado el diagrama entidad-relación que representa las necesidades de información de la empresa, el diseño de la base de datos consiste en la descripción del medio en el cual todos los datos serán almacenados. Posibles mecanismos de implementación incluyen papel, archivos manuales, archivos convencionales de computadora y diferentes tipos de sistemas manejadores de base de datos.

Las actividades a desarrollar son:

Determinar la técnica de implementación para las entidades: tablas de base de datos, archivos o medios especiales. Se valora si una entidad puede ser implementada como una sola tabla o archivo para el diseño inicial.

Realizar el diseño inicial de la base de datos o archivo.

Afinar el diseño de la base de datos o archivo para dar un mayor rendimiento a las necesidades de almacenamiento. Agregar índices u otros mecanismos de acceso. Predecir incrementos de datos.

Producir un plan detallado de tamaños y capacidades.

Revisar el diseño de la base de datos y predecir si el rendimiento podrá ser satisfactorio. Acordar los niveles de servicio y requerimientos de almacenamiento.

Crear la base de datos.

Las tareas descritas anteriormente son las más importantes y las que consumen más tiempo durante la etapa de diseño. De estas tareas se obtiene la mayor parte de los resultados de la etapa de diseño, los cuales se detallan adelante.

2.3 RESULTADOS OBTENIDOS EN ESTE CAPITULO

Definición de tablas, columnas, llaves primarias y llaves foráneas.
Definición de Indices por tabla.
Definición de Secuencias.
Definición de los módulos (Formas o Reportes)
Definición de los usuarios autorizados a usar la Aplicación.
Definición de la Base de Datos y de su Estructura (*Tablespaces*, Segmentos de *Rollback* y parámetros de almacenamiento - *Storage*-).

La información anterior es mostrada en los siguientes reportes.

CASE*Dictionary

Report : TABLES,COLUMNS AND FOREIGN KEY DERIVATIONS

Filename : cdkeys1.lis

Run by : TESIS

Report Date : 10-AUG-92 06:32pm

```
+-----------------------------------------------+
|                                               |
| Parameter values                              |
|                                               |
| Application System : ORACLE                   |
| Version            : 1                        |
| Table Name         : %                        |
|                                               |
+-----------------------------------------------+
```

Tables, Columns and Foreign Key Derivations

Table Name	Seq	Column Name	Column derived from
ACCIONES	2	NO_ACCION	
	4	NO_SAT	SATS.NO_SAT
	6	FECHA_ACCION	
	8	NO_STATUS	STATUS_SATS.NO_STATUS
	10	NO_PRIORIDAD	PRIORIDADES.NO_PRIORIDAD
	12	CVE_EMISOR	EMPLEADOS.CVE_EMPLEADO
	14	STATUS_ACCION	
	16	CVE_RECEPTOR	EMPLEADOS.CVE_EMPLEADO
	18	TIEMPO	
	20	VISIBLE	
ALUMNOS	2	NO_ALUMNO	
	4	NOMBRE	
	6	NO_CLIENTE	CLIENTES.NO_CLIENTE
AREAS	2	NO_AREA	
	4	NOMBRE	
CLIENTES	2	NO_CLIENTE	
	4	NOMBRE	
	6	NO_RAZON_SOCIAL	RAZON_SOCIAL.NO_RAZON_SOCIAL
	8	COMENTARIO	
	10	DIRECCION1	
	12	DIRECCION2	
	14	TELEFONO	
	16	FAX	
CLIENTE_TIPO	2	NO_CLIENTE	CLIENTES.NO_CLIENTE
	4	CVE_TIPO	TIPOS_D_CLIENTE.CLAVE
CONSULTORIAS	2	NO_CONSULTORIA	
	4	NO_CONTRATO	CONTRATOS.NO_CONTRATO
	6	MONTO	
	8	MONTO_DOLARES	

Tables, Columns and Foreign Key Derivations

Table Name	Seq	Column Name	Column derived from
CONTACTOS	2	NO_CONTACTO	
	4	NOMBRE	
	6	TELEFONO	
	8	NO_CLIENTE	CLIENTES.NO_CLIENTE
	10	DIRECCION1	
	12	DIRECCION2	
	14	FAX	
	16	PUESTO	
	18	MODEM	
CONTACTO_TIPO	2	CVE_TIPO	TIPOS_D_CONTACTO.CLAVE
	4	NO_CONTACTO	CONTACTOS.NO_CONTACTO
CONTRATOS	2	NO_CONTRATO	
	4	NO_CLIENTE	CLIENTES.NO_CLIENTE
	6	CVE_VENDEDOR	EMPLEADOS.CVE_EMPLEADO
	8	ESTADO	
	10	FECHA_RECONOCIMIENTO	
	12	TERMINOS_PAGO	
	12	TOTAL_DOLARES	
	14	TOTAL_A_PAGAR	
	16	FECHA_FIRMA	
COSTOS_POR_CREDITO	2	FECHA_FINAL	
	4	FECHA_INICIO	
	6	COSTO	
CURSOS_NOMINALES	2	CVE_CURSO	
	4	CREDITOS	
	6	NOMBRE	
CURSOS_PROGRAMADOS	2	NO_CURSO_PROGRAMADO	
	4	CUPO	
	6	CVE_CURSO	CURSOS_NOMINALES.CVE_CURSO
	12	FECHA_FINAL	
	14	FECHA_INICIO	
	16	CANCELADO	
	18	LUGAR	
	20	HORARIO_INICIO	
	22	HORARIO_FINAL	

Tables, Columns and Foreign Key Derivations

Table Name	Seq	Column Name	Column derived from
EDUCACIONAL	2	NO_EDUCACIONAL	
	4	NO_CREDITOS	
	6	NO_CONTRATO	CONTRATOS.NO_CONTRATO
	8	CREDITOS_RESTANTES	
	10	MONTO	
	10	MONTO_DOLARES	
EMPLEADOS	2	CVE_EMPLEADO	
	4	NOMBRE	
	6	NO_AREA	AREAS.NO_AREA
	8	FECHA_EGRESO	
	10	PUESTO	
	12	FECHA_INGRESO	
EMP_CURSO	2	CVE_EMPLEADO	EMPLEADOS.CVE_EMPLEADO
	4	NO_CURSO_PROGRAMADO	CURSOS_PROGRAMADOS.NO_CURSO_PROGRAMADO
ENCUESTAS_CURSOS	2	NO_ENCUESTA	
	4	NO_CURSO_PROGRAMADO	CURSOS_PROGRAMADOS.NO_CURSO_PROGRAMADO
	6	NO_PREGUNTA	PREGUNTAS_CURSOS.NO_PREGUNTA
	8	COMENTARIO	
	10	NO_RESPUESTA	
ENCUESTAS_SOPORTE	2	CVE_ING_SOPORTE	EMPLEADOS.CVE_EMPLEADO
	4	NO_PREGUNTA	PREGUNTAS_SOPORTE.NO_PREGUNTA
	6	NO_SAT	SATS.NO_SAT
	8	COMENTARIO	
	10	RESPUESTA	
FACTURAS	2	NO_FACTURA	
	4	FECHA_FACTURACION	
	6	NO_CONTRATO	CONTRATOS.NO_CONTRATO
	8	TOTAL_PESOS	
	10	TOTAL_DOLARES	
	12	CONCEPTO1	
	14	FECHA_PAGO	
	16	VIATICOS	
	18	PECE	
	20	IVA	
	22	ESTADO	
	24	DESCUENTO	
	26	CONCEPTO2	

Tables, Columns and Foreign Key Derivations

Table Name	Seq	Column Name	Column derived from
INFO_ACCIONES	2	NO_RENGLON	
	4	NO_ACCION	ACCIONES.NO_ACCION
	6	NO_SAT	SATS.NO_SAT
	8	TEXTO	
INSCRIPCIONES	2	NO_ALUMNO	ALUMNOS.NO_ALUMNO
	4	NO_CURSO_PROGRAMADO	CURSOS_PROGRAMADOS.NO_CURSO_PROGRAMADO
	6	CREDITOS	
	8	NO_EDUCACIONAL	EDUCACIONAL.NO_EDUCACIONAL
	10	FECHA_INSCRIPCION	
	12	ASISTENCIA	
	14	CALIFICACION	
	16	FORMA_PAGO	
	18	NUMERO_FACTURA	
	20	FECHA_CONFIRMACION	
LICENCIAS	2	NO_LICENCIA	
	4	NO_CONTRATO	CONTRATOS.NO_CONTRATO
	6	MONTO	
	8	MONTO_DOLARES	
NIVELES_PLATAFORMA	2	CVE_ING_SOPORTE	EMPLEADOS.CVE_EMPLEADO
	4	NO_PLATAFORMA	PLATAFORMAS.NO_PLATAFORMA
	6	NIVEL	
NIVELES_PRODUCTO	2	CVE_ING_SOPORTE	EMPLEADOS.CVE_EMPLEADO
	4	NO_PRODUCTO	PRODUCTOS.NO_PRODUCTO
	6	NIVEL	
PLATAFORMAS	2	NO_PLATAFORMA	
	4	DESCRIPCION	
PLATAFORMAS_CLIENTE	2	NO_SOFT_CLIE	
	4	CVE_SERIE	
	6	VERSION_SO	
	8	NO_PLATAFORMA	PLATAFORMAS.NO_PLATAFORMA
	10	MEDIO_DISTRIBUCION	
PREGUNTAS_CURSOS	2	NO_PREGUNTA	
	4	PREGUNTA	

Tables, Columns and Foreign Key Derivations

Table Name	Seq	Column Name	Column derived from
PREGUNTAS_SOPORTE	2	NO_PREGUNTA	
	4	PREGUNTA	
PRIORIDADES	2	NO_PRIORIDAD	
	4	DESCRIPCION	
PRODUCTOS	2	NO_PRODUCTO	
	4	DESCRIPCION	
PRODUCTOS_CONTRATADOS	2	NO_PARTIDA	
	4	NO_SOPT_CLIE	PLATAFORMAS_CLIENTE.NO_SOPT_CLIE
	6	NO_LICENCIA	LICENCIAS.NO_LICENCIA
	8	NO_PRODUCTO	PRODUCTOS.NO_PRODUCTO
	10	COSTO_DOLARES	
	10	NO_SOPORTE	SOPORTES.NO_SOPORTE
	12	VERSION	
	14	COSTO_PESOS	
RAZON_SOCIAL	2	NO_RAZON_SOCIAL	
	4	NOMBRE	

Tables, Columns and Foreign Key Derivations

Table Name	Seq	Column Name	Column derived from
SATS	2	NO_SAT	
	4	ENCABEZADO	
	6	FECHA_CREACION	
	8	NOMBRE_CONTACTO	
	10	NOMBRE_CLIENTE	
	12	NO_PRIORIDAD	PRIORIDADES.NO_PRIORIDAD
	14	NO_STATUS	STATUS_SATS.NO_STATUS
	16	NO_PRODUCTO	PRODUCTOS.NO_PRODUCTO
	18	FECHA_ACTUALIZACION	
	20	TELEFONO	
	22	CVE_ING_SOPORTE	EMPLEADOS.CVE_EMPLEADO
	24	NO_SOPORTE	PLATAFORMAS_CLIENTE.NO_SOPT_CLIE
	26	NO_PLATAFORMA	PLATAFORMAS.NO_PLATAFORMA
	28	CVE_RESPONSABLE_ESCALAMIENTO	EMPLEADOS.CVE_EMPLEADO
	30	COMENTARIO	
	32	CVE_ERROR	
	34	CVE_PCR	
	36	CVE_SERIE	
	38	MEDIO_DISTRIBUCION	
	40	FECHA_ESCALAMIENTO	
	42	VERSION_SO	
	44	VERSION_RDBMS	
	46	VERSION_PRODUCTO	
	50	NO_ERROR	
	52	MODEM	
	54	FAX	
	56	ESCALAMIENTO	
	58	CVE_TAR	
	60	DIRECCION1	
	62	DIRECCION2	
	64	NIVEL	
SOPORTES	2	NO_SOPORTE	
	4	FECHA_TERMINO	
	6	NO_CONTRATO	CONTRATOS.NO_CONTRATO
	8	MONTO	
	8	NO_SOPT_CLIE	PLATAFORMAS_CLIENTE.NO_SOPT_CLIE
	12	MONTO_DOLARES	
STATUS_SATS	2	NO_STATUS	
	4	DESCRIPCION	
TIPOS_O_CLIENTE	2	CLAVE	
	4	DESCRIPCION	

Tables, Columns and Foreign Key Derivations

Table Name	Seq	Column Name	Column derived from
TIPOS_D_CONTACTO	2	CLAVE	
	4	DESCRIPCION	

3. HERRAMIENTAS DE CUARTA GENERACION Y ETAPA DE IMPLEMENTACION

HERRAMIENTAS DE CUARTA GENERACION Y ETAPA DE IMPLEMENTACION

Durante la etapa de implementación los programas son codificados y probados utilizando las herramientas apropiadas.

El proceso de implementación incluye planificación, diseño de la estructura de los programas, codificación y las pruebas tanto a nivel módulo como a nivel sistema. Además se considera la selección del método apropiado con el cual se llevará el control de este trabajo y el control de las versiones de los programas, módulos de prueba, etc.

Actualmente existen varias herramientas que se pueden utilizar en la implementación de un sistema, éstas van desde los lenguajes de tercera generación hasta los ambientes de desarrollo, en los cuales se integran varias de las utilerías necesarias para la implementación del sistema, tales como lenguajes de cuarta generación, reporteadores, generadores de documentación, etc.

En el tema 3.1 "HERRAMIENTAS DE CUARTA GENERACION" de este capítulo se describe el proceso y herramientas que se utilizan en las llamadas TECNICAS DE CUARTA GENERACION para el desarrollo de sistemas, se presentan las características de los lenguajes de cuarta generación y se lleva a cabo la comparación entre las herramientas que ofrecen IBM y ORACLE para la implementación de sistemas, ya que los resultados que se obtienen del capítulo anterior, pueden ser utilizados por las herramientas de ambos proveedores.

En el tema 3.2 "ACTIVIDADES REALIZADAS EN LA FASE DE IMPLEMENTACION" se detallan los pasos que se siguen dentro de la herramienta CASE ORACLE para cubrir la fase de implementación del sistema y el uso de las herramientas ORACLE de desarrollo que han sido empleadas.

Por último en el tema 3.3 "RESULTADOS OBTENIDOS EN ESTE CAPITULO" se presenta un resumen de la información obtenida durante la etapa de implementación.

3.1 HERRAMIENTAS DE CUARTA GENERACION

TECNICAS DE CUARTA GENERACION

El término técnicas de cuarta generación (T4G) abarca un amplio rango de herramientas de *software* que tienen algo en común: todas ofrecen la facilidad durante el desarrollo de *software* de especificar las características del sistema a desarrollar a alto nivel. Luego, la herramienta genera automáticamente el código fuente basándose en las especificaciones dadas.

Actualmente, un entorno para el desarrollo del *software* que soporte la habilidad de especificar las necesidades a un nivel que sea más próximo al lenguaje natural o en una notación que proporcione funciones significativas, debe incluir algunas (o todas) de las siguientes herramientas: lenguajes no procedurales para consulta a bases de datos, generación de reportes, manipulación de datos, interacción y definición de pantallas y generación de código; capacidades gráficas de alto nivel; y capacidad de hoja de cálculo. Cada una de estas herramientas existen, pero sólo para dominios de aplicación muy específicos. No existe disponible actualmente un entorno T4G que pueda aplicarse con igual facilidad a todas las categorías de aplicaciones existentes.

Las técnicas de cuarta generación comienzan con la recolección de los requerimientos. Idealmente el cliente debe describir los requerimientos y éstos deben traducirse directamente en un prototipo operacional; pero esto no funciona en la realidad. El cliente puede no estar seguro de lo que necesita, puede ser ambiguo en la especificación de hechos que son conocidos y puede ser incapaz o no desear especificar la información en la forma en que una herramienta T4G pueda emplearla. Además, las herramientas actuales de las T4G no son lo suficientemente sofisticadas para acomodarse realmente al lenguaje "natural". En este momento el dialogo cliente-técnico permanece como una parte esencial del enfoque T4G (*Ver figura 3.1.1*).

Para aplicaciones pequeñas, puede ser posible ir directamente desde el paso de establecimiento de los requerimientos a la implementación, usando un lenguaje de cuarta generación no procedural (L4G). Sin embargo, es necesario un mayor esfuerzo para desarrollar una estrategia de diseño para el sistema, inclusive si se utiliza un L4G. El uso de T4G sin diseño (para grandes proyectos) causará las mismas dificultades (poca calidad, pobre

mantenimiento, mala aceptación por parte del cliente) que se encuentran cuando se desarrolla *software* usando los métodos convencionales.

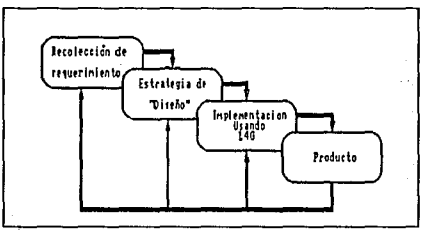


Figura 3.1.1
Técnicas de Cuarta Generación

La implementación usando L4G facilita al que desarrolla el *software*, la descripción de los resultados deseados, los cuales se traducen automáticamente en código fuente para producir dichos resultados. Obviamente, debe existir una estructura de datos con información relevante y debe se accesible por L4G con rapidez.

El último paso de la *figura 3.1.1* contiene la palabra "producto", para transformar una implementación T4G en un producto, el que lo desarrolla debe dirigir una prueba completa, desarrollar una documentación con sentido y ejecutar todas las "actividades de transición" requeridas. Además, el software desarrollado con T4G debe ser construido de forma que facilite el mantenimiento.

El estado actual de los métodos T4G es el siguiente:

- Con muy pocas excepciones, el dominio de aplicación actual de los T4G está limitado a las aplicaciones de sistemas de información comerciales, específicamente al análisis de información y la obtención de informes en grandes bases de datos. Hasta la fecha, T4G se ha usado muy poco en productos de ingeniería.

- La recolección de datos preliminares, que acompaña al uso de T4G, parece indicar que el tiempo requerido para producir *software* se reduce mucho.

- Sin embargo, el uso de T4G para grandes trabajos de desarrollo de *software*, exige el mismo o más tiempo de análisis, diseño y prueba (actividades de ingeniería de *software*), perdiéndose así un tiempo sustancial que se ahorra mediante la eliminación de la codificación.

Las técnicas de cuarta generación se convertirán probablemente en una parte importante del desarrollo de *software* en un tiempo no muy lejano.

La *figura 3.1.2* muestra cómo pueden combinarse los procesos necesarios para el desarrollo de *software*. En todos los casos el trabajo comienza con la recolección de requerimientos. El método puede seguir el ciclo de vida clásico (ingeniería de sistemas y análisis de requerimientos del software) o puede ser la definición menos formal del problema usada en la construcción de un prototipo. Independientemente, debe producirse la comunicación cliente-desarrollador del *software*.

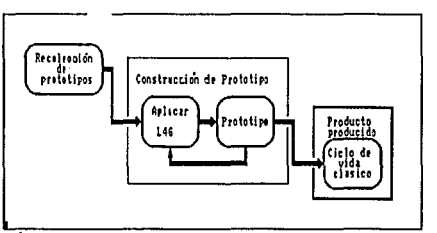


Figura 3.1.2
Procesos necesarios para el desarrollo de *software*

LENGUAJES DE CUARTA GENERACION

A lo largo de la historia del desarrollo del *software* siempre se han generado programas de computadora cada vez con mayores niveles de abstracción. Los lenguajes de la primera generación trabajaban a nivel de instrucciones de máquina, el menor nivel de abstracción posible. Los lenguajes de segunda y tercera generación han subido el nivel de representación de programas de computadora, pero aún hay que especificar distintos procedimientos algorítmicos perfectamente detallados. Durante la pasada década, los lenguajes de cuarta generación (L4G) han elevado aún más el nivel de abstracción.

Un lenguaje de cuarta generación es una herramienta para el desarrollo de sistemas integrados, la cual ofrece una mejora en la productividad de 10 a 1 con respecto a los lenguajes de tercera generación y que consta de un conjunto de instrucciones fáciles de usar y aplicar.

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 detalles algorítmicos. Por ejemplo, la sentencia:

```
COMPUTE NET-PRESENT-VALUE AND RETURN-ON-INVESTMENT
FOR EXPENDITURES #5 AND #9,
```

es típica de un L4G. El sistema "sabe" como calcular los datos financieros deseados y lo hace sin que el programador tenga que especificar los algoritmos apropiados. Claramente el "conocimiento" que se ha descrito es *específico del dominio financiero*. O sea, que ese L4G inevitablemente no entenderá lo siguiente:

```
COMPUTE THE ROOTS OF TRANSCENDENTAL EQUATION #3 AND APPLY THEM TO
PHYSICAL MODEL #4,
```

aunque otro L4G, diseñado específicamente para el dominio de este tipo de aplicación pudiera hacer correctamente el trabajo.

Los lenguajes de cuarta generación combinan características procedurales y no procedurales. O sea, el lenguaje permite al usuario especificar condiciones con sus correspondientes acciones (componente procedural). También, el usuario puede especificar

únicamente el resultado que desea obtener (componente no procedural), y el L4G es capaz de encontrar los detalles procedurales aplicando el conocimiento adquirido de algún dominio específico.

Hay diferentes tipos de L4G, algunos son solamente lenguajes de programación, mientras que otros son herramientas funcionales de desarrollo, las cuales pueden usar menús, diálogos u otros tipos de interfaces para el usuario.

Lenguajes de Petición

Hasta ahora, la gran mayoría de los L4G se han desarrollado para ser usados conjuntamente con aplicaciones de bases de datos. Estos a veces llamados lenguajes de petición permiten al usuario manipular de forma sofisticada la información contenida en la base de datos previamente creada. Algunos lenguajes de petición tienen una sintaxis compleja que no es más sencilla que la de los lenguajes de tercera generación. Sin embargo, otros lenguajes de petición actualmente disponibles ofrecen una interfaz de forma muy parecida al lenguaje natural.

Generadores de Programas

Los generadores de programas presentan 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 instrucciones. Estos lenguajes de programación de muy alto nivel hacen un fuerte uso de la abstracción de datos y de procedimientos.

Funciones

La orientación a funciones es más moderna que los lenguajes de programación, pero no puede ser usado para el mantenimiento de sistemas existentes e implica la instalación de una metodología nueva.

Sistemas de Soporte de Decisiones

Son una combinación de herramientas DBMS, análisis estadístico, modelo financiero y reporteador diseñado para soportar modelado interactivo y situaciones del estilo "Que pasa si". Estos sistemas son típicamente usados por jefes y analistas, no por programadores, y normalmente opera fuera del centro de información.

Los modelos son creados por o bajo la supervisión de quien toma las decisiones. El modelo final nunca es de propósito general, es creado para resolver un requerimiento de una decisión en particular.

Otros L4G

Aunque los lenguajes de petición y los generadores de programas son los L4G más comunes, existen otras categorías. Los lenguajes de soporte en la toma de decisiones permiten que los no programadores lleven acabo una gran variedad de análisis del tipo "que pasa si", que van desde los simples modelos de hojas de cálculo bidimensionales hasta los sofisticados sistemas de modelos estadísticos y de investigación operativa.

Los lenguajes de prototipos se han desarrollado para asistir en la creación de prototipos facilitando la creación de interfaces para el usuario y la creación de diálogos, además de proporcionar los medios para el modelado de datos.

Los lenguajes de especificación formal también se pueden considerar L4G cuando producen código máquina ejecutable.

Características de los ambientes de desarrollo que incluyen herramientas de cuarta generación.

Un ambiente de desarrollo que incluye herramientas de cuarta generación debe constar idealmente de los siguientes elementos:

- **DBMS:** El cual contenga al menos un lenguaje manipulador de datos de alto nivel con navegación automática a través de la base de datos y vistas lógicas para el usuario o procesamiento relacional, además interfaces para reportes hacia otros sistemas de DBMS.

- **Diccionario Integrado de datos:** Esto debe estar unido a DBMS y debe poder controlar funciones de integridad de datos.

- **Facilidades de Consultas:** Ya sea con un lenguaje parecido al natural o a través de menús que soporten funciones de selección booleana.

- **Lenguaje No Procedural:** El cual soporte instrucciones de alto nivel tales como SELECT FROM, SORT, DELETE

- **Lenguaje Procedural:** Con instrucciones propias como "IF...THEN, DO", comandos aritméticos, etc. o bien contar con interfaces hacia L3G.

- **Diseño de Pantallas:** Creación de sistemas en línea dibujando pantallas de entrada y salida, así como flujo de pantallas.

- **Procesamiento de Texto:** Mínimo un editor

- **Facilidades de Migración:** Para accesar utilerías existentes mientras se hace la migración a un nuevo medio ambiente (Transparencia).

- **Soporte completo del ciclo de vida de un sistema:** Incluyendo Codificar, diseñar, probar, documentar, mantenimiento y manejo de proyectos.

- **Comandos para el manejo de Integridad y Seguridad.**

- **Adicionales:** Gráficas, Paquete Estadístico, Respaldos y Recuperación, Seguridad, Portabilidad, Generación de Documentación, Librerías, Ayudas, Entrenamiento.

Los beneficios obtenidos de emplear L4G en la construcción de sistemas son varios, entre ellos podemos citar: los sistemas son construidos más rápidamente, los usuarios finales están conformes con sus sistemas bien afinados, el usuario final puede construir sus propias aplicaciones fácilmente, las aplicaciones construidas con L4G son más fáciles de mantener que aquellas escritas con lenguajes de tercera generación (L3G). Esto es principalmente por el hecho de que el código no tiene que ser alterado, compilado, depurado, etc. Además los sistemas construidos con L4G necesitan menos documentación.

HERRAMIENTAS L4G PARA EL DESARROLLO DEL SISTEMA

Como se vió en el capítulo anterior, la herramienta de CASE ORACLE, la cual se usó en el desarrollo de este sistema de tesis, puede generar la estructura de la base de datos en DB2 o en ORACLE y el código de *default* necesario para usarse con las herramientas de desarrollo de cada Compañía.

A continuación se detallan las características tanto de DB2 como de ORACLE, base de datos y herramientas, y se hace un análisis de éstas.

Herramientas de Desarrollo DB2

DB2 (*Database 2*) es la base de datos relacional de IBM, emplea el lenguaje estándar de consultas SQL. DB2 corre en equipos *mainframes* de IBM bajo los sistemas operativos MVS/XA y MVS/SP, DB2 requiere 12 Megas de memoria real. SQL/DS otra de las aplicaciones de bases de datos de IBM corre en equipos VM/SP y DOS/VSE.

Las principales características de la base de datos de DB2 se enumeran a continuación:

· **Portabilidad**

DB2 y SQL/DS son productos completamente diferentes. (DB2 y SQL/DS generan código no portables, ni de SQL/DS a DB2 ni viceversa). Ninguno de estos dos productos ofrece la portabilidad de las aplicaciones desarrolladas a un ambiente PC.

DB2 no es una base de datos cien por ciento portable, dado que las aplicaciones construidas con CSP y AS (herramientas de desarrollo descritas más adelante) son portables sólo a SQL/DS en sistemas operativos DOS/VSE y VM/SP, es decir sólo entre equipos IBM. Por otro lado, ni las aplicaciones construidas en DB2, ni la hechas en SQL/DS pueden correr en una Computadora Personal (PC Personal Computer) de IBM.

· **Conectabilidad**

DB2 no tiene procesamiento distribuido ni tampoco el soporte de bases de datos distribuidas.

IBM tiene varios productos que permiten extraer datos de DB2 en procesos *batch*, entre ellos están *Data Extract facility, DXT*. Todos los datos extraídos pueden ser leídos por varios productos de PC incluyendo Lotus 1-2-3.

Server-Requester Programming Interface (SRPI) son una serie de herramientas para desarrollar aplicaciones interactivas entre el *Host* y PC.

El propósito de SRPI es definir como son enviados los requerimientos de la PC al *host* y como el *host* responde. A través de SRPI, una aplicación en PC puede accesar la base de datos del host, sin ningún proceso intermedio tales como cargar los datos en la PC y convertir el archivo a ASCII.

El *Remote Relational Access System* (RRASS) permite que las aplicaciones en una máquina VM/SP se conecten a una base de datos SQL/DS en otra máquina. Esto es un procesamiento distribuido, no una base de datos distribuida y solamente trabaja en VM.

· **Respaldos y Recuperación**

DB2 soporta respaldos en línea, recuperación y respaldos parciales de la base de datos, *logical after-imaging* y *checkpoints*.

· **RDBMS**

El manejador de la base de datos de DB2 integra con su Customer Information Control System (CICS) transaction monitor y con el Information Management System (IMS) los productos de la base de datos tradicional jerárquica de IBM. Este incluye el protocolo "two-phase commit" para la coordinación de los datos referenciados en DB2 o en IMS. Además cuenta con interfaces a servicios del sistema operativo.

DB2 emplea candados en los recursos para obtener lecturas consistentes. Emplea candados a nivel página con escalación, lo cual puede causar *deadlocks* en muchas situaciones.

DB2 cuenta con el llamado Sistema-R, el cual es un prototipo relacional del manejador de la base de datos, este sistema nace con el producto SQL/DS y adiciona las capacidades de las bases de datos distribuidas.

Starburst es una extensión al manejador de la base de datos, el cual en teoría es una arquitectura abierta que proporciona al usuario la capacidad de incrementar la funcionalidad de la base de datos. Esta permite a los usuarios construir estructuras avanzadas de bases de datos, objetos complejos, tipos de datos definidos por el usuario y métodos de acceso, tales como una búsqueda geográfica. Una de las metas de esta función es permitir al usuario intercambiar objetos o componentes entre una base de datos y otra de acuerdo a las necesidades de la aplicación.

Entre las principales características de *performance* de DB2 se incluye:

- Un optimizador de consultas
- Módulos accesibles (Meta código SQL)
- Candados a nivel página

- Estabilidad de cursores (sacrifica lecturas consistentes para incrementar la concurrencia).
- *Fast-commit* (DB2 confirma los cambios escribiendo simplemente al archivo de *log*).
- *Rollback* a nivel instrucción.

· **Herramientas de desarrollo**

Las herramientas de aplicación más populares de DB2 son: *Cross System Product* (CSP) y *Aplication System* (AS). A continuación se detallan estas herramientas de aplicación de DB2 y otras.

CSP:

CSP es un lenguaje de cuarta generación (L4G) constructor de aplicaciones, su nombre *Cross System* significa que CSP soporta varios manejadores de bases de datos (DBMS) y sistemas operativos de los *mainframes* de IBM. En esencia CSP es un lenguaje procedural que proporciona un medio de construir aplicaciones basadas en formas.

Esta herramienta es la más significativa de las ofrecidas para desarrollo en DB2 y SQL/DS. Una de sus mayores ventajas está en poder mover aplicaciones entre productos de base de datos y sistemas operativos IBM.

CSP soporta transacciones pseudo-conversacionales bajo CICS, esto es comúnmente usado para el procesamiento de transacciones en línea a gran escala.

CSP proporciona una pantalla de diseño y una facilidad para definición de campos. CSP requiere que se escriban páginas de código para completar la aplicación. La lógica de la secuencia en la cual se despliegan las pantallas debe ser codificada cuando éstas consulten o actualicen la base de datos, en que momento desplegar un mensajes, etc.

Un conjunto de pantallas son usadas en CSP para construir los procedimientos requeridos por una aplicación, pero el usuario aún tiene que escribir código.

CSP tiene un soporte de modo bloque muy completo.

AS:

Application System (AS) es otro producto L4G de IBM, este producto tiene como finalidad el que el usuario o programador use datos existentes en reportes ya construidos, gráficas y modelos financieros simples. AS permite que los datos de DB2 sirvan como entradas y salidas de sus componentes. AS tiene su propio lenguaje procedural, el cual no es compatible con CSP, este lenguaje puede ser usado para controlar el flujo de las aplicaciones a límites extensos; sin embargo, el modelo de aplicaciones arcaicas de AS limita esta ventaja.

Partes de este producto cuentan con una interfaz basada en formas.

AS funciona también como procesador de documentos, es posible incorporar datos de la base de datos dentro de una carta, cuenta también con un módulo manejador de estadísticas. Ofrece la capacidad de generar reportes dentro de un rango respetable.

AS proporciona un paquete para planear proyectos de medio nivel, este paquete es más poderoso de lo que pudiera ser un similar a nivel PC.

Aún cuando AS cuenta con características de graficación comercial, el alto costo del procesamiento en *mainframes* ha ocasionado que muchos usuarios migren hacia ambientes de PC. AS no está disponible en computadoras personales.

Financial Modeling es otro módulo de AS que permite usar los *mainframes* para hacer cosas que son manejadas de mejor forma desde una PC.

QMF:

Query Management Facility provee de una interfaz hacia SQL basada en el llenado de formas y provee además del producto *Query-by-Example* (QBE) para reportes y gráficas. Este proporciona un editor de tablas para consultas simples y entrada de datos.

ICU:

Interactive Chart Utility, es un paquete de graficación comercial. Muchos productos IBM (como CSP) permiten llamadas a ICU para el generado de gráficas.

IC/1:

Information Center /1, es una colección de viejas herramientas, el cual incluye VS APL y ADRS II.

TIF:

The Information Facility es un simple producto para la ejecución de consultas.

Precompiladores:

DB2 ofrece características únicas en sus precompiladores, por lo cual no es posible migrar aplicaciones construidas con estos precompiladores entre Oracle y DB2. DB2 soporta estructuras, módulos de acceso, y ofrece un precompilador para el lenguaje ensamblador 370 llamado ALC.

DB2 soporta diversos lenguajes de programación, incluyendo aquellos con conjunto de caracteres de 16 bits.

- **Herramientas en PC IBM**

Las herramientas ofrecidas para las computadoras personales IBM incluyen: VM/BOND, ECF, RECC/PC, HDBV, *File Transfer* IBM 3270, PROFS/PC, PS/PC y más. Se puede ver que la gran cantidad de nombres y acrónimos empleados es a veces causa de confusión.

Herramientas de Desarrollo ORACLE

El sistema manejador de Bases de Datos relacionales de Oracle está compuesto de un *Kernel*, un diccionario integrado de datos y una gran variedad de utilerías (generador de aplicaciones, generador de reportes, consultas interactivas, editor de textos, procesador de palabras, hoja de cálculo, etc.) para bases de datos que permiten manejar todas las operaciones generalizadas a un DBMS (*Data Base Management System*). Aunque el lenguaje SQL es la base del RDBMS, existen muchas más utilerías opcionales que pueden ser usadas para satisfacer las necesidades particulares de cada usuario, algunas de estas opciones incluyen interfaces entre microcomputadoras y *mainframes*, interfaces de precompilación, opciones de comunicación entre redes, y una gran variedad de interfaces de consulta.

· **Portabilidad**

ORACLE corre en una gran variedad de sistemas operativos y en todos ellos presenta la misma funcionalidad.

En *Mainframes*: IBM System/370 y compatibles bajos los sistemas operativos MVS, VM/CMS y Amdahl UTS.

En microcomputadoras : Data General Elipse MV 4/8/10 mil, bajo AOS/VS; Equipos Digital VAX-11/725 a 11/785 y VAX 8000 bajo VAX/VMS. Además de una lista muy larga de sistemas basados en Unix. Muchos de estos sistemas están diseñados para correr en microprocesadores 80386- o basados en 68000.

En microcomputadoras también se tiene una lista muy larga de equipos soportados, entre las cuales podríamos mencionar: IBM PC XT y AT y 3270 PC bajo PC-DOS, IBM compatibles bajo MS-DOS o PC-DOS, Texas Instruments Professional bajo DOS, MicroVax y MicroVax II bajo MicroVMS, etc.

ORACLE tiene la habilidad de mover bases de datos y aplicaciones entre sistemas distintos. Esto permite al usuario desarrollar una aplicación en una microcomputadora y usarla en un *mainframe* o en una minicomputadora.

· **Conectividad**

Las capacidades de procesamiento distribuido de los datos son soportadas por SQL*Star con SQL*Net y el conjunto de protocolos asociados.

El usuario puede a través de SQL*Net usar una gran variedad de máquinas e incluso de DBMS en una red y emplear los nodos como si estuvieran consultando una sola base de datos colocada dentro del sistema ORACLE. Esta transparencia e independencia de máquinas puede eventualmente salvar a grandes corporaciones de gastar grandes sumas de dinero en reemplazos de *hardware* y tiempo de procesamiento. Es posible con estas redes accesar datos en cualquier parte del mundo sin tener que transferir los archivos.

· **Integridad y Seguridad**

La seguridad completa de la base de datos es soportada por una utilería de chequeo automático combinada con instrucciones que pueden ser usadas por usuarios privilegiados para establecer los privilegios de uso de cada nuevo usuario. Existen otras utilerías

con las cuales los usuarios pueden restringir el acceso a las aplicaciones (*Triggers*).

Para mantener la integridad de los datos ORACLE emplea candados que pueden ser exclusivos que previenen que cualquier dato sea alterado o bien compartidos que sólo previenen de que los datos que están siendo leídos sean alterados. Los candados pueden ser aplicados a tablas o renglones dentro de las tablas.

· **RDBMS**

El diccionario de datos está integrado dentro del RDBMS y proporciona una interfaz básica entre SQL y el Kernel. El diccionario define las columnas de las tablas, las relaciones entre ellas mismas y las vistas de las tablas. El diccionario mantiene la seguridad y el control de la información en datos y usuarios del sistema.

Está desarrollado en lenguaje C, lo cual deriva en muchos beneficios, ya que permite a ORACLE ser fácilmente adaptado a una gran variedad de medio ambientes *hardware* y lo hace accesible a un gran número de usuarios.

Las principales características de manejador de la base de datos son:

- Rutinas de ordenamiento internas (*Group by, Order by, Create Index y Select Disctinct*)
- Facilidades de *Insert-Fetch, Array Fetch e Insert*
- Optimización de Predicados
- Módulos accesibles compartidos
- *Fast-Commit*
- Candados a nivel registro

· **Herramientas de Desarrollo**

Las herramientas de desarrollo y manipulación de datos que ofrece Oracle son las siguientes:

SQL:

Es un lenguaje integrado para todo propósito que incluye funciones de Lenguaje de Definición de Datos (DDL Data Definition Language), Lenguaje de Manipulación de Datos (DML Data Manipulation Language), Lenguaje de Consultas, y Lenguaje de Control de Datos (DCL Data Control Language) para el manejo de seguridad. SQL es

usado para definir y manipular todas las tablas, índices y el diccionario de datos. SQL proporciona funciones de consulta extendidas, para hacer la selección de datos por diversos criterios, cuenta con la capacidad de usar operadores tales como MINUS e INTERSECT los cuales permiten combinaciones más flexibles para la manipulación de datos.

Uno de los atributos más importantes de las bases de datos relacionales es la de permitir que dos o más tablas sean unidas/mezcladas de acuerdo a un criterio específico de selección para derivar una nueva relación que contenga elementos de las relaciones originales. Oracle incluye además la capacidad de realizar *"outer joins"* lo cual permite relacionar elementos que no están explícitamente especificados en la condición de la unión. Las funciones de Conteo de registros (Count), Promedios (Average), Sumas (Sum), Valores mínimos (Minimum) y Valores máximos (Maximum) pueden ser incluidas en la instrucción de selección, para tener un criterio matemático de modulación de los datos.

ORACLE tiene la capacidad de reconocer campos nulos en las expresiones aritméticas y asignarles un *default* de cero o bien el valor especificado por el usuario al momento de realizar la consulta. ORACLE soporta el manejo de vistas, las vistas son una formá de visualizar proyecciones o la unión de una o más relaciones.

EASY*:

Esta interfaz está diseñada para usuarios ocasionales o novatos. No es necesario el conocimiento de la sintaxis de SQL dado que el usuario utiliza menúes de opción múltiple y técnicas de apuntar y seleccionar para hacer sus requerimientos. Además se tienen menúes *pull-down* y sensitivos al contexto. Se proveen adicionalmente ayudas en línea.

PRO*:

Estas interfaces de programación están diseñadas para usuarios profesionales (expertos), los cuales usan llamadas a programas o precompiladores de muchos de los lenguajes de alto-nivel que se usan hoy en día. Entre estos lenguajes se incluye COBOL, Fortran, PL/1, C, Basic y Ada.

SQL*:

Esta interfaz de comandos está diseñada para usuarios finales con el entrenamiento necesario para manejar la programación de sistemas. Las interfaces SQL* son más eficientes y poderosas que las interfaces EASY*. Estas soportan el uso de todas las funciones de SQL. Se emplean menúes de pantalla completa y técnicas interactivas. Cuenta además con opciones de ayuda en línea.

Entre estas opciones se incluye SQL*Plus para manipulación de datos y reportes sencillos, SQL*Forms la herramienta de desarrollo de aplicaciones, SQL*Net para opciones de comunicación y las orientadas a usuarios finales SQL*Graph, SQL*Calc y Easy*SQL.

SQL*Plus:

Este producto es un lenguaje de cuarta generación (L4G) completo diseñado para manejar todas interacciones con ORACLE. Incluye reportes y funciones inteligentes de transferencia. Permite a los usuarios crear, modificar y unir datos de tablas, control de acceso a la base de datos y transferencia de datos.

SQL*Forms:

ORACLE soporta una herramienta de desarrollo basada en formas, interactiva y multi-usuario, llamada SQL*Forms, la cual soporta desarrollo y definiciones de pantallas completas de captura y consulta de datos. La pantalla es definida a través de una forma la cual es definida con un generador interactivo de aplicaciones. La forma define los campos en la pantalla, los campos de aceptación de valores, valores iniciales, parámetros de validación de datos, valores calculados (incluyendo consultas), mensajes de ayuda y la función de valores accesible al operador y rangos de valores válidos.

Una pantalla de diseño interactiva proporciona la libertad de diseñar y colocar donde se desee los elementos de la forma. El rango completo de comandos de SQL pueden ser ejecutados condicionalmente como parte de la aplicación. SQL*Forms puede ser usado como una herramienta prototipo, incluyendo lógica condicional y soporte de funciones optimizadas. Provee al usuario de una gran variedad opciones de validación de datos y procedimientos de seguridad de datos, habilitando al usuario a implementar múltiples aplicaciones y macros procedurales.

SQL*Forms se utiliza generalmente para procesamiento de transacciones en línea, con una interfaz fácil de usar y amigable para el usuario, tanto en su diseño, como en su ejecución.

Este producto tiene la característica de tener una apariencia uniforme independiente del ambiente. No importa si se utiliza una interfaz ANSI, Motif, modo bloque, X-terminal, etc.

SQL*Menu:

SQL*Menu permite crear menús de aplicaciones para estructurar el acceso a la información de una forma más natural al usuario. El empleo más frecuente de menús se da en el uso de formas con la característica de tener una estructura jerárquica. Pero su uso es aún mayor, y puede extenderse tanto como aplicaciones haya, proporcionando siempre el nivel de seguridad que se necesita. El diseño y mantenimiento de los menús es sencillo.

Son soportadas una gran variedad de tipos de terminales y tiene integradas utilerías de depuración de menús.

SQL*Reportwriter:

Con esta herramienta es posible escribir una gran variedad de reportes: tabulares, etiquetas, cartas, matrices, etc. Es una herramienta interactiva a la cual se le suministran las especificaciones del reporte a través de una interfaz amigable y fácil de usar, se cuenta con ayuda en línea y sensible al contexto y listas de valores. Los reportes creados son vistos en pantalla al tiempo que se diseñan y la misma interfaz es empleada para modificarlos.

Soporta consultas múltiples a la base de datos de ORACLE, y soporta además consultas a bases de datos No-ORACLE como DB2 y SQL/DS vía SQL*Connect.

Cuenta con documentación automática vía "Reporte de reportes" y Soporta las características de Lenguaje Nacional (NLS National Language Support) para el manejo de mensajes y en la interfaz en general.

PL/SQL:

PL/SQL es un lenguaje de programación sofisticado para el desarrollo de aplicaciones que está integrado tanto al sistema manejador de la base de datos (RDBMS Relational Data Base Managment

System) de ORACLE como a sus herramientas clave (SQL*Forms y SQL*Menu). PL/SQL incrementa significativamente el rendimiento (performance) de las aplicaciones y la productividad de los desarrolladores.

PL/SQL es un poderoso lenguaje procedural diseñado para aplicaciones de bases de datos, ofrece la flexibilidad de soportar SQL integrado y soporta la funcionalidad de un lenguaje estándar tales como declaración de variables y cursores, control condicional, ciclos construidos y un mecanismo avanzado para el manejo de errores.

PL/SQL versión 2.0 cuenta con la facilidad de tener procedimientos almacenados en varias bases de datos remotas que pueden ser referenciados simultáneamente desde un solo bloque de PL/SQL. Es posible declarar arreglos de una dimensión cuya función es parecida a los arreglos en lenguajes de tercera generación y es posible especificar tablas como parámetros a procedimientos y funciones.

Los procedimientos de PL/SQL son almacenados en la base de datos, con lo cual es posible contar con procedimientos compartidos, los cuales pueden llamar a otros lenguajes y construir aplicaciones modulares y sofisticadas. El mantenimiento a los módulos es automático, si un módulo es actualizado, automáticamente los módulos que lo referencian son recompilados. La interfaz de paso de parámetros permite al desarrollador enviar y traer resultados de los procedimientos.

Los procedimientos podrán ser llamados simultáneamente desde las herramientas clave de ORACLE. Podrán declararse variables y cursores globales para todos los procedimientos y paquetes.

Se reducen los requerimientos de memoria con esta nueva funcionalidad de procedimientos compartidos. La herramientas desde las cuales se podrá hacer uso de esta nueva funcionalidad son: SQL*Forms, SQL*Menu, SQL*ReportWriter, SQL*Plus, SQL*DBA y la interfaz Programática de Oracle.

Oracle*Mail:

Es un sistema de correo electrónico portable y distribuido construido sobre el DBMS relacional ORACLE. Opera prácticamente en cualquiera de los computadores, sistemas operativos o redes conocido, ofreciendo una comunicación transparente para los usuarios en cualquier parte de su organización.

Facilita el flujo de información general en un ambiente de oficina. Facilita el enviar y recibir correo de cualquier parte del sistema y permite una fácil distribución de la información de cualquier tipo (texto, archivos binarios, hojas de cálculo etc.)

Ofrece *Gateways* de comunicación a otros productos importantes de correo electrónico.

· **Herramientas en PC ORACLE**

Las versiones para microcomputadoras o PC son una representación completa de las versiones en mainframes y minicomputadoras, no un subconjunto derivado para medioambientes pequeños.

Análisis Comparativo de las herramientas de Desarrollo

DB2

VENTAJAS

CSP tiene algunas ventajas de performance sobre ORACLE entre las cuales están:

- Elimina tiempo de corrida al no hacer el *Parsing* de las instrucciones de SQL pues lo hace con el plan de aplicaciones de DB2.
- Corre en modo pseudo-conversacional bajo CISC.
- DB2 provee candados a nivel página para mejorar la concurrencia

La ventaja de IBM está en soportar una amplia variedad de sistemas software de MVS. El soporte de CISC es importante para soportar aplicaciones que requieran de grandes transacciones procesadas en línea.

DESVENTAJAS

DB2 por sí mismo es sólo un manejador de la base de datos sin herramientas de acceso para usuarios finales. Los usuarios pueden

usar QMF el cual incluye tanto SLQ como QBE, ambos productos son significativamente menos que un L4G completo.

En una PC, no es posible accesar los datos de DB2 almacenados en un *mainframe* excepto cargando los archivos con productos no-SQL propios de una PC.

CPS es una herramienta que fuerza al usuario a escribir páginas de código aún para construir la aplicación más simple.

CPS no provee de una herramienta que facilite la construcción de reportes, es decir no provee de una solución completa para el desarrollo de aplicaciones, además de que no ofrece integración con los otros productos de IBM.

AS, por otro lado tiene muchas características pero está pobremente integrada con DB2. Ningún producto se comunica uno con otro.

ORACLE

VENTAJAS

Oracle tiene muchas ventajas, una de las más significativas es que las versiones para micros proporcionan la funcionalidad completa de los sistemas que corren en *mainframes* y minicomputadoras, permitiendo la instalación de un sistema en común para usarse a través de un medio ambiente corporativo.

Oracle tiene la habilidad de mover bases de datos y aplicaciones entre sistemas diferentes. Esto permite a los usuarios desarrollar aplicaciones en una microcomputadora y correrlas en un *mainframe* o minicomputadora. Esto permite que las compañías puedan descentralizarse y distribuir las aplicaciones de un mainframe a minicomputadoras departamentales o quizá en microcomputadoras de escritorio.

Otra ventaja importante es la introducción de SQL*Start. Con este ambiente de bases de datos distribuidas, el usuario puede emplear una red SQL*Net consistente de una variedad de máquinas diferentes y Sistemas Manejadores de Bases de Datos (DBMS) diferentes también, y utilizar estos nodos como si estuvieran en el mismo lugar en donde el sistema de ORACLE está siendo consultado.

Usa el lenguaje de consultas SQL de IBM, lo cual lo hace compatible con SQL/DS y DB2.

Consta de una gran variedad de herramientas de consulta de datos y desarrollo de aplicaciones, desde las más sencillas enfocadas a usuarios finales, quienes no necesitan tener ningún conocimiento previo de computación, hasta formas complejas de programar (proprecompiladores) para adaptar las aplicaciones a las necesidades más completas y complejas de los usuarios.

Con las herramientas ORACLE, es posible construir la gran mayoría de las aplicaciones sin tener que escribir código alguno. Sin embargo, si la aplicación así lo requiere, es posible incorporar código de algún lenguaje de cuarta generación (PL/SQL) y/o de alguno de tercera generación (gracias a los precompiladores Oracle)

Esta arquitectura es abierta y permite el ensamble de las diferentes partes de la aplicación de una forma simple y natural.

DESVENTAJAS

Las utilerías para generar reportes no son muy versátiles, se tiene limitado el tipo de reportes que pueden ser generados.

Las utilerías para exportar e importar datos son lentas.

Un uso excesivo del CPU y en general de recursos es algo que pudiera ser mejorado.

3.2 ACTIVIDADES REALIZADAS EN LA FASE DE IMPLEMENTACION

Una vez que se han definido las características de cada módulo, funcionalidad y herramienta en la cual será implementado (etapa de diseño), se procede a la generación del código de cada uno de estos módulos. Esta generación de código se lleva a cabo durante la etapa de implementación.

Las tareas con mayor importancia que se desarrollaron en esta etapa fueron:

- Generación de la base de datos
- La producción de programas
- Preparación, realización y revisión de las pruebas al sistema.
- Descripción y producción de los manuales de usuario y documentación del sistema.

Generación de la Base de Datos

Se crearon y ejecutaron los programas que contienen las definiciones obtenidas en el capítulo anterior (Diseño) para la base de datos. Dichas definiciones incluyen información tanto de la estructura de la base de datos como de tamaños de los archivos que integrarán la misma.

Producción de programas

Dentro de esta tarea se codificaron y probaron cada uno de los programas por separado. Para este sistema, los programas fueron de tres tipos: formas de captura, menús y reportes.

Las formas de captura son la parte principal del sistema ya que desde ellas se lleva el control de toda la información, es el medio por el cual los usuarios insertan, borran, actualizan y consultan la información que ellos deseen. Dentro de estas formas se lleva también el control de integridad y consistencia de la información, así como de muchas otras validaciones.

Los menús son utilizados para facilitar al usuario la interacción con las formas. Mediante el manejo de menús el usuario puede navegar y utilizar todas las facilidades que presenta el sistema sin importar el nombre o localización de éstas.

En los reportes, la información se presenta al usuario en un formato predefinido. Los reportes pueden ser impresos en papel o solamente consultados en pantalla.

Para la implementación de cada uno de estos tipos de programas se utilizaron las herramientas SQL*FORMS, SQL*MENU y SQL*REPORTWRITER.

Se puede decir que aproximadamente el 90% de la codificación de cada uno de estos programas es generada automáticamente por el CASE*Generator de ORACLE, tomando para ello toda la información que se especificó durante la etapa de diseño. Solamente características muy específicas del programa tienen que ser codificadas desde la herramienta correspondiente (SQL*FORMS por ejemplo).

Una vez que se ha terminado con la generación y pruebas de cada uno de los programas se procede a la unión de éstos para formar todo el sistema.

Para familiarizar a los usuarios con el uso del sistema, se le dio el nombre de AMIGO (Aplicación Maestra de la Información General de Oracle).

Preparación, realización y revisión de las pruebas al sistema

Una vez que se hace la unión de todos los programas, se lleva a cabo la prueba conjunta de éstos. Para ello se definen previamente los tipos de pruebas a las que el sistema va a ser sometido, se realizan estas pruebas y se revisan los resultados obtenidos. Si los resultados no son satisfactorios, se regresa hasta la etapa de diseño (si es necesario), para corregir el problema. Se generan nuevamente los programas que presentaron problemas y se prueba nuevamente el sistema. Este proceso se repite hasta que se tengan los resultados esperados.

Descripción y producción de los manuales de usuario y documentación del sistema.

En forma paralela a la implementación del sistema, se hace la generación de los manuales que serán utilizados por los usuarios del sistema, así como la documentación necesaria para la administración del sistema.

Dentro de la documentación generada se encuentra el manual de usuario, el manual de referencia y el manual del sistema.

EQUIPO Y HERRAMIENTAS EN LAS SE QUE DESARROLLO EL SISTEMA

Para el desarrollo de este trabajo de tesis se usaron las herramientas de desarrollo ORACLE, debido a la versatilidad y variedad de herramientas con las que cuenta, y a que el empleo de estas herramientas no representará ningún gasto adicional a la empresa. Además las herramientas CASE de ORACLE son capaces de generar código básico para las herramientas propias de Oracle: SQL*Forms, SQL*Menu, SQL*Reportwriter y SQL*Plus. Esto representa una gran ventaja para emplear dichas herramientas en el desarrollo del sistema.

Por otro lado actualmente se cuenta en las oficinas de ORACLE de México con los siguientes equipos:

Equipo:	Sequent Symetry
Sistema Operativo:	Dynix/ptx versión 1.3
Procesador:	80486
Memoria:	64 Megabytes
Multi-usuario:	Si

Equipo:	VAX/3600
Sistema Operativo:	VMS versión 5.1
Procesador:	Digital
Memoria:	32 Megabytes
Multi-Usuario:	Si

Equipo:	ALR 486
Sistema Operativo:	SCO Unix versión 3.2
Procesador:	80486
Memoria:	12 Megabytes
Multi-Usuario:	Si

Equipo:	PS/2 IBM
Sistema Operativo:	DOS versión 5.0
Procesador:	80386
Memoria:	3 Megabytes
Multi-Usuario:	No

Para la implementación del manejo de información de ORACLE de México hubiera sido posible emplear cualquiera de las plataformas antes mencionadas dado que los productos de ORACLE son los mismos en cualquier equipo; sin embargo, el sistema fue implementado en ORACLE corriendo bajo el equipo *Sequent* por razones de espacio en disco, capacidad de procesamiento y de memoria, es decir de

recursos y performance en general, este equipo ofrece más ventajas en cuanto a rendimiento corriendo las aplicaciones de ORACLE. En este equipo se tendrá toda la información referente a clientes e información de soporte y será accesado por todas las personas involucradas en el manejo de esta información, el acceso a la información puede ser *concurrente* o no.

Es importante destacar que no sería factible emplear las herramientas bajo un equipo PC dado que uno de los requerimientos del sistema es que sea multi-usuario y las computadoras personales no ofrecen esta característica.

3.3 RESULTADOS OBTENIDOS EN ESTE CAPITULO

Los resultados obtenidos de esta etapa fueron:

- Sistema completo
- Documentación del usuario
- Base de datos física

4. ETAPA DE TRANSICION

4.1 ETAPA DE TRANSICION

Dentro de la etapa de transición se realizan todas las tareas necesarias para que el sistema entre en funcionamiento, además, se provee el período inicial de soporte al sistema. La transición se tiene que llevar a cabo de forma tal que no cause problemas a la empresa, que sea lo más trasparente posible, y que los usuarios queden lo suficientemente preparados para la utilización del nuevo sistema.

Se provee la capacitación necesaria y se asiste al usuario en la realización de las pruebas de aceptación. Los datos (que utilizaba el sistema anterior) son convertidos y las instalaciones de *software/hardware* faltantes son terminadas. La puesta en marcha del sistema incluye la carga de los nuevos datos, conversión de archivos y una serie de pruebas que aseguren que el sistema está listo para entrar en producción. El período inicial de soporte al sistema incluye la asistencia a los usuarios, la observación del funcionamiento y rendimiento del sistema, y la corrección de fallas.

Algunos de los puntos importantes a tomar en cuenta en esta etapa de transición son:

- Asegurar que la capacitación a usuarios es apropiada y efectiva.

- Asegurar que el usuario efectúe tantas pruebas de aceptación como sean necesarias para probar que el sistema está operacionalmente correcto y que el usuario se familiarice con éste.

- Asegurar que las personas que van a proporcionar el soporte técnico tengan la capacitación y documentación necesaria, para que sean capaces tanto de entender el sistema como de diagnosticar los problemas y que sepan que acciones deben seguir en caso de que exista alguna falla.

- Asegurar que la integración o coexistencia con los sistemas existentes se lleve sin problemas.

Dado que la transición es frecuentemente una de las etapas más críticas, es muy posible que se plantee la situación de tener a los dos sistemas (nuevo y anterior) corriendo en paralelo con el fin de que no se presentes problemas en el desempeño de la empresa. Es muy difícil aceptar los cambios. No existe otra cosa más difícil que aprender a hacer las cosas utilizando un camino diferente.

Un punto importante es hacer que los usuarios del sistema proporcionen la capacitación a otros usuarios. Esto se puede lograr si se hace un programa de capacitación adecuado, con el manejo de datos de prueba reales y con la utilización de documentación entendible. Con este material, las personas involucradas en el desarrollo del sistema, proporcionarán la capacitación a los usuarios más calificados que tendrán la tarea de capacitar a otros usuarios.

El nuevo sistema será propiedad de los usuarios, por ello, éste tiene que ser aceptado por ellos. Una buena aceptación es obtenida cuando los usuarios entienden ampliamente lo que ellos están generando, incluyendo los problemas así como los beneficios. Es un hecho que la gente se acostumbra rápidamente a los beneficios, los cuales son aceptados de forma automática, olvidando los problemas que en un pasado existieron.

La aceptación es un proceso de dar y recibir. El grupo de desarrollo debe estar preparado para resolver inmediatamente todos los problemas que se le presenten al usuario. Los usuarios tienen que poner su mayor esfuerzo para que todos los problemas aparezcan durante esta etapa y no en la de producción. La pruebas de aceptación deben ser controladas por los usuarios y supervisadas por el equipo de desarrollo.

Las pruebas de aceptación deben ser utilizadas para probar los aspectos operacionales del sistema; por ejemplo, funcionamiento con datos reales, pruebas de volúmenes de información, solución de problemas vía telefónica, etc. Por lo tanto, estas pruebas deben ser corridas por los usuarios como si estuvieran en producción. La puesta en marcha del sistema es planeada y manejada por el equipo de desarrollo en conjunto con los usuarios para verificar las pruebas y dar a aprobación final para la utilización del sistema.

4.2 ACTIVIDADES REALIZADAS DURANTE LA ETAPA DE TRANSICION

En el desarrollo de nuestro sistema, la etapa de transición quedo conformada con las siguientes actividades.

CAPACITACION A USUARIOS

El lo particular, esta fue una tarea fácil, ya que los usuarios a los cuales capacitamos para la utilización del sistema, ya tenían conocimiento del manejo de las herramientas con las cuales se desarrolló dicho sistema (SQL*FORMS, SQL*MENU, etc.).

La capacitación se llevo a cabo con una plática de aproximadamente dos horas, en la cual se mostraron las características del sistema y las diferencias con el sistema anterior. En esta plática se proporcionó el manual de usuario y se explicó su contenido y utilización. Se dio una demostración detallada del funcionamiento del sistema.

Después se asesoró en forma individual a cada uno de los usuarios en su tarea específica de operación del sistema, hasta que se comprobó la correcta utilización de éste.

PRUEBAS DE ACEPTACION

En esta parte los usuarios corrieron el sistema con datos de prueba los cuales eran muy parecidos a los datos reales. Generaron una lista de todos los problemas a los que se enfrentaron, fue necesario resolver algunos de estos problemas en el momento en que se presentaron. Se llevó a cabo otra reunión con los usuarios para resolver las dudas específicas y poner a consideración todos los problemas presentados. En esta misma plática se pidieron los comentarios de los usuarios acerca del sistema los cuales fueron de mucha utilidad para mejoras al sistema. Se realizaron las modificaciones o adaptaciones del sistema de acuerdo a observaciones de funcionamiento o sugerencias propias de los usuarios.

CONVERSION DE DATOS

Para poder emplear los datos que se estaban utilizando en el sistema anterior y algunos datos que se usan actualmente en Oracle USA, se hizo la conversión de éstos utilizando las herramientas ORACLE. Las herramientas utilizadas fueron:

- SQL*LOADER

- IMPORT/EXPORT

- SQL*PLUS

SQL*LOADER es una herramienta que nos permite leer información de un archivo plano y cargarla a la base de datos. El archivo de entrada puede estar en cualquier formato ya que SQL*LOADER es muy flexible en este aspecto.

IMPORT/EXPORT son dos herramientas que nos permiten transferir información de una base de datos Oracle a otra. EXPORT lee la información de una base de datos y genera un archivo con un formato especial, el cual sólo puede ser leído mediante el IMPORT. Con el IMPORT se lee este archivo y su contenido es cargado en otra base de datos o en la misma de donde se obtuvo el EXPORT.

Con SQL*PLUS se hicieron algunos procedimientos que sirvieron para cargar información a la base de datos.

Con la utilización de estas herramientas se utilizó aproximadamente un 80% de la información que ya se manejaba.

INSTALACION DE HARDWARE/SOFTWARE PARA EL AMBIENTE DE PRODUCCION

En realidad no hizo falta instalar *hardware* ni *software* adicional para poner el sistema en producción ya que este se tenía instalado. Pero se quiere hacer mención en este punto de todas las diferentes plataformas que se utilizaron durante el desarrollo del sistema.

Máquina: Dec-station 2100.

Sistema Operativo: Ultrix 4.1.

Ubicación: Centro de Cálculo de la Facultad de Ingeniería UNAM.

Utilización: En esta máquina se llevaron a cabo las etapas de análisis y diseño del sistema, aquí se hicieron los diagramas funcionales y el diagrama entidad relación. Se utilizó este equipo ya que la herramienta CASE*DESIGNER de ORACLE necesita un equipo con capacidades gráficas y éste las tenía.

Máquina: MicroVax 3600.

Sistema Operativo: VMS 5.0.

Ubicación: Oficinas de ORACLE de México.

Utilización: Esta máquina se utilizó durante la etapa de implementación, esto fue debido a que en esta máquina se encontraban las versiones más recientes del *software* de ORACLE, además de que en esta máquina se encuentra corriendo el sistema anterior, lo que facilitó la utilización de algunos datos de prueba que se utilizaron durante la implementación.

Máquina: Sequent S27.

Sistema Operativo: Dynix/ptx.

Ubicación: Oficinas de ORACLE de México.

Utilización: Esta máquina se utilizó para la puesta en producción del sistema.

Como se observa, todos los equipos empleados son muy diferentes entre sí, sin embargo, la migración de información de uno a otro fue totalmente transparente.

PUESTA EN MARCHA DEL SISTEMA

La versión de producción del sistema se liberó en la máquina Sequent, para lo cual se hizo la migración desde la máquina VAX, nuevamente esta migración fue totalmente transparente.

4.3 RESULTADOS OBTENIDOS EN ESTE CAPITULO

Los resultados obtenidos de este capítulo fueron:

- Documentación completa del sistema
- Usuarios capacitados en la utilización del sistema
- Datos acondicionados al nuevo sistema
- Sistema instalado y puesto en operación

5. ETAPA DE PRODUCCION

5.1 ETAPA DE PRODUCCION

Esta es la fase final en la construcción de un sistema. En la etapa de producción se asegura que el sistema se encuentre funcionando de una manera correcta, sin que el equipo de soporte tenga que estar interviniendo continuamente. También, se realizan monitoreos del uso y rendimiento del sistema. Los cambios necesarios se deben hacer de forma tal que no produzcan problemas, asegurando la entera satisfacción de los usuarios.

5.2 ACTIVIDADES REALIZADAS DURANTE LA ETAPA DE PRODUCCION

MONITOREO Y REVISION DEL RENDIMIENTO

La actividad que principalmente realizamos en esta etapa fue la de monitorear el sistema. Esto fue con el fin de determinar el rendimiento que estaba teniendo el sistema y las posibles mejoras que se le pudieran hacer.

Con los resultados obtenidos del monitoreo, se determinaron los cambios que se deberían hacer a algunos parámetros de la base de datos, se tuvo el cálculo más exacto del crecimiento de las tablas, se aseguró que se habían creado los índices adecuados y se crearon otros para aumentar el rendimiento. Por otro lado, también se hizo la revisión de los parámetros del sistema operativo que podían influir en el rendimiento del sistema.

PROCEDIMIENTOS DE RESPALDO Y RECUPERACION

Se establecieron los mecanismos mediante los cuales se llevarían a cabo los procedimientos de respaldo y recuperación del sistema. Los cuales fueron documentados en el manual de usuario. Los métodos de respaldo utilizados fueron el respaldo en frío y el export.

ASISTENCIA A USUARIOS

Se proporcionó ayuda a los usuarios que tenían problemas en la utilización del sistema. Se creo un procedimiento mediante el cual los usuarios podían documentar todos los problemas a los cuales se enfrentaban, pero se dio un mayor énfasis a los nuevos requerimientos que pudieran surgir de la utilización del sistema. Después el equipo de soporte podía analizar cada uno de estos problemas o peticiones y podía determinar la acción a seguir.

Los problemas importantes fueron solucionados en el mismo momento en el cual se presentaron.

PRESENTACION DE RESULTADOS

Se hizo la presentación del sistema a todos los gerentes de la compañía.

Se obtuvieron algunos reportes con información real.

Se hizo la entrega de la documentación necesaria a las personas involucradas y encargadas directamente de algún módulo en particular dentro del sistema.

Se declaró formalmente la terminación del sistema.

6. CONCLUSIONES

CONCLUSIONES

Durante el desarrollo de esta tesis se efectúo una breve pero concisa revisión de las principales metodologías de desarrollo de sistemas. Como se señaló anteriormente, el método *CASE*Method* constituye una herramienta moderna que subsana los defectos de metodologías previas a ésta, y adaptándose a las necesidades actuales de información de una empresa, provee las características fundamentales de otras metodologías como son la modularidad y la cobertura del ciclo de vida completo de un sistema.

Partiendo de la metodología *CASE*Method* y después de llevar a cabo una serie de entrevistas con el personal clave de la empresa fue posible plasmar en diagramas las necesidades de información de Oracle de México, es importante señalar que los resultados principales de esta fase se concretan en dos tipos de diagrama: el diagrama entidad-relación y los diagramas funcionales, los cuales representan de forma óptima la estructura de la información que es necesario procesar, y las actividades fundamentales que se llevan a cabo con esta información.

Para el desarrollo de este trabajo empleamos la herramienta *Oracle*CASE* por las siguientes razones: Uno de los lineamientos del presente trabajo de tesis es desarrollar el sistema con herramientas de Oracle en forma integral, aun cuando alguna de las otras herramientas *CASE* descritas en el capítulo dos se hubiese adaptado al desarrollo de este sistema, *Oracle*CASE* fue utilizado por cumplir con la anterior restricción, y porque su empleo no representaría un gasto adicional para la empresa a la cual se le desarrolló el sistema, pero además queremos mencionar que es una herramienta completa para el desarrollo de sistemas y que presenta ventajas con respecto a productos similares. A esto podemos añadir la ventaja de contar con un soporte técnico en línea desde Oracle Corporation (San Francisco, Ca.) en caso de que algún problema se presentase durante el empleo de *Oracle*CASE*.

Como resultado de haber utilizado *Oracle*CASE*, se concretó la estructura física y lógica de la base de datos que contendría la información de la empresa, necesaria para nuestro sistema, generando *scripts* que servirían para la construcción automática de ésta. Esta misma herramienta generó formas de pantalla, reportes y menús que cumplían casi en su totalidad con las expectativas del proyecto. El siguiente paso consistió en utilizar las herramientas de cuarta generación que permitieron adecuar las interfaces y reportes generadores creados por *Oracle*CASE*.

Después de emplear *Oracle*CASE* dimos dos opciones, desarrollar el sistema empleando herramientas de DB2 o emplear las herramientas propias de Oracle. Por las razones expuestas anteriormente,se utilizó la segunda opción describiendo las herramientas con las que contamos durante la fase de la implementación para desarrollar formas, reportes, menús, etc.

Creemos importante resaltar que empleamos las herramientas Oracle por resultar de muy bajo o nulo costo para la empresa, no obstante, una de las desventajas mostrada de estos productos es su costo elevado con respecto a otro tipo de herramientas. Por ejemplo, las herramientas Oracle para ambientes de PC tienen un valor de 1500 US dólares, que comparado con el precio de paquetes manejadores de archivos o bases de datos de alrededor de 500 US dólares, puede ser muy costoso considerando el beneficio que el sistema de información proveerá.

Como parte final del trabajo aquí presentado se implementó el sistema de información adecuado a las necesidades reales de la empresa, el cual como se mencionó en el capítulo de implementación hemos llamado AMIGO. Este sistema se adaptó e integró en un todo la información que antes ya se manejaba por separado en cada una de las áreas de la compañía. Puesto que las diferentes áreas de la empresa no contaban con un sistema de información como éste, la etapa de transición fue relativamente simple y sin dificultades posteriores.

El sistema para la automatización de información de Oracle de México ha sido, hasta ahora, herramienta fundamental en el funcionamiento de la misma empresa, dado que cubre sus necesidades de información. Este resultado pudo haber sido no satisfactorio de no haber seguido correctamente una adecuada metodología de desarrollo, y su tiempo pudo haber sido muy prolongado de no haber utilizado herramientas de cuarta generación, que nos permitieron adecuar lo generado por *Oracle*CASE* en un lapso breve.

Al desarrollar este trabajo de tesis hemos aportado no solamente un sistema que es base fundamental para la compañía ORACLE, sino también, una investigación teórica de las metodologías de desarrollo, herramientas CASE y herramientas de cuarta generación, misma que puede ser una base para otros estudiantes que pueden encontrar aquí información básica acerca de CASE y Oracle*CASE.

Los conocimientos adquiridos a lo largo del estudio de nuestra carrera, fueron fundamentales en la toma de decisiones para el desarrollo de este tema de tesis y nos han servido como base durante todas las fases del proyecto.

Pensamos que con la culminación de este trabajo y el sistema se han cumplido los objetivos planteados al inicio; un estudio de algunas de las metodologías de desarrollo y la implementación de un sistema de información funcional para la empresa.

BIBLIOGRAFIA

BIBLIOGRAFIA

Methodology: The Experts Speak
Ken Orr, Chris Gane, Edward Yourdon,
Peter P Chen, Larry L. Constantine
Revista Byte Abril 1989

CASE*Method Market Analysis
Competitive Bulletin
Renee Taylor
Diciembre 31, 1989

CASE*Method vs. Yourdon Structured Design
Competitive Bulletin
Renee Taylor
Septiembre 19, 1989

Ingeniería del Software, un enfoque práctico.
Roger S. Pressman,
Ed. Mc Graw Hill

Software Engineering.
Marvin V. Zelkowitz,
Publicaciones Auerbach.

Tesis: Desarrollo de una herramienta automatizada
para el diseño de sistemas.
Eduardo Alejandro Bistrain, Juan Carlos Corral. UIA

CASE Method. Entity Relationship Modelling.
Richard Barker
A.W. Publishing Company

Revista: ORACLE, The SQL database Journal.
Vol. III, Número 2 1989

The 1986 National Database and
Fourth generation language symposium.
Digital consulting associates, Inc.
Spring 1986 Edition

CASE*DICTIONARY Reference Guide
Version 5.0 Beta II
ORACLE

APENDICE A

GLOSARIO DE TERMINOS

GLOSARIO DE TERMINOS

Análisis Funcional:
Identificar las funciones que se realizan en la empresa actualmente, como se hacen y que se necesita hacer para el futuro.

Análisis de Entidades:
Se definen y entienden las cosas significantes acerca de las cuales la empresa necesita conocer o mantener información y las relaciones entre ellas.

Análisis de Datos:
Basándose en las entidades se analizan los valores de los datos que serán estudiados.

Atributos:
Detalles que nos sirven para calificar, identificar, clasificar, cuantificar o expresar el estado de ocurrencia de una entidad.

Base de Datos:
Es una colección organizada de datos almacenados la cual es definida independientemente de los procesos que la usaran y soporta la definición y mantenimiento de las relaciones entre los datos.

CASE: *Computer Aided System Engineering* (Ingeniería de Software Asistido por Computadora).

DBMS: *Data-Base Management System* (Sistema manejador de Bases de Datos).

DFD: *Data-Flow Diagram* (Diagrama de Flujo de Datos).

Dominio: Un conjunto de valores con características en común estos valores están sujetos a un conjunto de reglas de validación en común.

DSSD: *Data-Structured System Development* (Desarrollo Estructurado de Sistemas de Datos).

Entidad: Algo que tenga existencia distinta y separada y acerca de lo cual la empresa necesita conocer o mantener información ya sea real o imaginada.

ERD: *Entity-Relationship Diagram* (Diagrama Entidad-Relación).

Evento: Cuando ocurre algo (borrar, agregar, conectar) bajo cierta circunstancia. Los eventos del sistema se definen al cumplirse una o más funciones.

IEM: *Information Engineering Method* (Método de Ingeniería de Información).

QBE: *Query By Example* (Consultas a través de ejemplos).

Relaciones:
La forma en la cual dos cosas del mismo o de diferente tipo se asocian, lo que una cosa hace con respecto a otra.

SAT: Solicitud de Asistencia Técnica.

SQL: *System Query Languages*.

STD: *State-Transition Diagram* (Diagrama de Transición de Estados).

APENDICE B

CODIGO PARA LA GENERACION DE LA BASE DE DATOS

```
REM
REM This ORACLE V6 RDBMS command file was generated by
REM CASE*Dictionary                    on 06-SEP-92
REM
REM For application system ORACLE version 1
REM
SET SCAN OFF

REM  Objects being generated in this file are:-
REM DATABASE
REM       ORAMEX
REM TABLESPACE
REM       DATOS
REM       INDICES
REM       ROLLBACKS
REM       TEMPORAL
REM USER GRANTS
REM       AAMADOR
REM       ACAMPOS
REM       AMONTERR
REM       ATOBIAS
REM       CGUZMAN
REM       DOLVERA
REM       FAGUIRRE
REM       JOROPEZA
REM       JRAMIREZ
REM       KOVSE
REM       MDIAZ
REM       MHERNANDEZ
REM       PUBLIC
REM       SVARGAS

PROMPT
PROMPT Creating Database ORAMEX

CREATE DATABASE ORAMEX
 CONTROLFILE REUSE
 MAXLOGFILES 10
 MAXDATAFILES 40
 MAXINSTANCES 1
 NOARCHIVELOG
 EXCLUSIVE
      DATAFILE '/d1s4/oramex/dbs/dbs1mex.dbf' SIZE 10 M
      LOGFILE '/d1s4/oramex/dbs/log1mex.dbf' SIZE 3 M
    ,       '/d1s4/oramex/dbs/log2mex.dbf' SIZE 3 M
;
```

```
REM Creando el diccionario de datos y vistas para IMP/EXP

PROMPT Creating data dictionary
SET TERMOUT OFF
@?/rdbms/admin/catalog ;
@?/rdbms/admin/expvew ;
SET TERMOUT ON

REM Tablespace para almacenamiento de Datos
REM
PROMPT
PROMPT Creating Tablespace DATOS
CREATE TABLESPACE datos
      DATAFILE '/d1s4/oramex/dbs/dat1mex.dbf' SIZE 50 M
      ONLINE
;

REM
REM Tablespace para el almacenamiento de Indices
REM
PROMPT
PROMPT Creating Tablespace INDICES
CREATE TABLESPACE indices
      DATAFILE '/d1s4/oramex/dbs/ind1mex.dbf' SIZE 30 M
      ONLINE
;

REM
REM Tablespace exclusivo de segmentos de rollback
REM
PROMPT
PROMPT Creating Tablespace ROLLBACKS
CREATE TABLESPACE rollbacks
      DATAFILE '/d1s4/oramex/dbs/roll1mex.dbf' SIZE 10 M
      ONLINE
;

REM
REM Tablespace dedicado a segmentos temporales
REM
PROMPT
PROMPT Creating Tablespace TEMPORAL
CREATE TABLESPACE temporal
      DATAFILE '/d1s4/oramex/dbs/tem1mex.dbf' SIZE 10 M
      ONLINE
;
```

```
REM
REM  Agustin Amador
REM
PROMPT Granting Access To AAMADOR
GRANT CONNECT ,RESOURCE TO aamador IDENTIFIED BY AAMADOR;

REM
REM  Alejandro CAmpos
REM
PROMPT Granting Access To ACAMPOS
GRANT CONNECT ,RESOURCE TO acampos IDENTIFIED BY ACAMPOS;

REM
REM  Angel Monterrubio
REM
PROMPT Granting Access To AMONTERR
GRANT CONNECT ,RESOURCE TO amonterr IDENTIFIED BY AMONTERR;

REM
REM  Gerente Soporte Tecnico
REM
PROMPT Granting Access To ATOBIAS
GRANT CONNECT ,RESOURCE TO atobias IDENTIFIED BY ATOBIAS;

REM
REM  Carlos Guzman
REM
PROMPT Granting Access To CGUZMAN
GRANT CONNECT ,RESOURCE TO cguzman IDENTIFIED BY CGUZMAN;

REM
REM  Domingo Olvera
REM
PROMPT Granting Access To DOLVERA
GRANT CONNECT ,RESOURCE TO dolvera IDENTIFIED BY DOLVERA;

REM
REM  Francisco Aguirre
REM
PROMPT Granting Access To FAGUIRRE
GRANT CONNECT ,RESOURCE TO faguirre IDENTIFIED BY FAGUIRRE;

REM  Gerente de Servicios Tecnics
REM
PROMPT Granting Access To JOROPEZA
GRANT CONNECT ,RESOURCE TO joropeza IDENTIFIED BY JOROPEZA;
```

```
REM
REM  Jorge Ramirez
REM
PROMPT Granting Access To JRAMIREZ
GRANT CONNECT ,RESOURCE TO jramirez IDENTIFIED BY JRAMIREZ;

REM
REM  Karen Ovseyevitz
REM
PROMPT Granting Access To KOVSE
GRANT CONNECT ,RESOURCE TO kovse IDENTIFIED BY KOVSE;

REM
REM  Miriam Diaz
REM
PROMPT Granting Access To MDIAZ
GRANT CONNECT ,RESOURCE TO mdiaz IDENTIFIED BY MDIAZ;

REM
REM  Margarita Hernandez
REM
PROMPT Granting Access To MHERNANDEZ
GRANT CONNECT ,RESOURCE TO mhernandez IDENTIFIED BY MHERNANDEZ;

REM
REM  Sandra Vargas
REM
PROMPT Granting Access To SVARGAS
GRANT CONNECT ,RESOURCE TO svargas IDENTIFIED BY SVARGAS;

REM
REM  End of command file
REM
EXIT
```

```
REM
REM This ORACLE V6 RDBMS command file was generated by
REM CASE*Dictionary                    on  06-SEP-92
REM
REM For application system ORACLE version 1
REM
SET SCAN OFF

REM  Objects being generated in this file are:-
REM TABLE
REM       ACCIONES
REM       ALUMNOS
REM       AREAS
REM       CLIENTES
REM       CLIENTE_TIPO
REM       CONSULTORIAS
REM       CONTACTOS
REM       CONTACTO_TIPO
REM       CONTRATOS
REM       COSTOS_POR_CREDITO
REM       CURSOS_NOMINALES
REM       CURSOS_PROGRAMADOS
REM       EDUCACIONAL
REM       EMPLEADOS
REM       EMP_CURSO
REM       ENCUESTAS_CURSOS
REM       ENCUESTAS_SOPORTE
REM       FACTURAS
REM       INFO_ACCIONES
REM       INSCRIPCIONES
REM       LICENCIAS
REM       NIVELES_PLATAFORMA
REM       NIVELES_PRODUCTO
REM       PLATAFORMAS
REM       PLATAFORMAS_CLIENTE
REM       PREGUNTAS_CURSOS
REM       PREGUNTAS_SOPORTE
REM       PRIORIDADES
REM       PRODUCTOS
REM       PRODUCTOS_CONTRATADOS
REM       RAZON_SOCIAL
REM       SATS
REM       SOPORTES
REM       STATUS_SATS
REM       TIPOS_D_CLIENTE
REM       TIPOS_D_CONTACTO
```

```
REM INDEX
REM       ACC_CLASIFICADA__EN_FRGN
REM       ACC_CLASIFICADA_____FRGN
REM       ACC_DIRIGIDA_A_FRGN
REM       ACC_HECHA_POR_FRGN
REM       ACC_PARTE_DE_FRGN
REM       ACC_PK_PRIM
REM       ALU_PARTE_DE_FRGN
REM       ALU_PK_PRIM
REM       AREA_PK_PRIM
REM       CLI_PK_PRIM
REM       CLI_TCL_FK2_FRGN
REM       CLI_TCL_FK_FRGN
REM       CLI_TIPO_PK_PRIM
REM       CNO_PK_PRIM
REM       CONSULTORIA_PK_PRIM
REM       CONS_CONT_FRGN
REM       CONTACTO_PK_PRIM
REM       CONTRATO_PK_PRIM
REM       CONTR_FIRMADO_FRGN
REM       CONTR_VENDIDO_POR_FRGN
REM       CONT_ASOCIADO_CON_FRGN
REM       CONT_EDU_FRGN
REM       CONT_LIC_FRGN
REM       CONT_TIPO_PK_PRIM
REM       CPC_PK_PRIM
REM       CPR_ASOCIADO_____A_FRGN
REM       CPR_PK_PRIM
REM       EDC_CALIFICANDO_FRGN
REM       EDC_RESPONDIENDO_FRGN
REM       EDU_PK_PRIM
REM       EMP_CPR_FK2_FRGN
REM       EMP_CPR_FK_FRGN
REM       EMP_CPR_PK_PRIM
REM       EMP_PARTE_DE_FRGN
REM       EMP_PK_PRIM
REM       ENCUESTAS_CURSOS_PK_PRIM
REM       ESO_CALIFICANDO2_FRGN
REM       ESO_CALIFICANDO_FRGN
REM       ESO_PK_PRIM
REM       ESO_RESPONDIENDO_FRGN
REM       FAC_CREADA_PARA_FRGN
REM       FAC_PK_PRIM
REM       IAC_PARTE_DE_FRGN
REM       IAC_PK_PRIM
REM       INS_PARA2_FRGN
```

```
REM       INS_PARA_FRGN
REM       INS_PK_PRIM
REM       INS_RESPALDADA_POR_FRGN
REM       LIC_PK_PRIM
REM       NDPR_PARA2_FRGN
REM       NDPR_PARA_FRGN
REM       NDPR_PK_PRIM
REM       NDP_PARA2_FRGN
REM       NDP_PARA_FRGN
REM       NDP_PK_PRIM
REM       NO_RAZON_SOCIAL_FRGN
REM       PCL_ASOCIADA_A_FRGN
REM       PCL_PK_PRIM
REM       PCO_CONSTAR_DE_FRGN
REM       PCO_ESTABLECIDO_PARA_FRGN
REM       PCO_PK_PRIM
REM       PCO_SOPORTADO_EN_FRGN
REM       PCU_PK_PRIM
REM       PLA_PK_PRIM
REM       PRI_PK_PRIM
REM       PRO_PK_PRIM
REM       PSO_PK_PRIM
REM       RSO_PK_PRIM
REM       SAT_ASIGNADA_FRGN
REM       SAT_ASISTIDA_EN_FRGN
REM       SAT_CATEGORIZADA_EN_FRGN
REM       SAT_CLASIFICADA_____FRGN
REM       SAT_OCURIENDO_____EN_FRGN
REM       SAT_OCURRIENDO_____E_FRGN
REM       SAT_PK_PRIM
REM       SAT_SUPERVISADA_EN_S_FRGN
REM       SDS_PK_PRIM
REM       SOP_CON_FRGN
REM       SOP_PK_PRIM
REM       TCL_PK_PRIM
REM       TCO_CONT_FK2_FRGN
REM       TCO_CONT_FK_FRGN
REM       TCO_PK_PRIM
```

```
REM     Created from Entity ACCION by TESIS on 29-APR-92
PROMPT
PROMPT Creating Table ACCIONES
CREATE TABLE acciones(
 no_accion                      NUMBER(3,0)       NOT NULL,
 no_sat                         NUMBER(5,0)       NOT NULL,
 fecha_accion                   DATE              NOT NULL,
 no_status                      NUMBER(2,0)       NULL,
 no_prioridad                   NUMBER(1,0)       NULL,
 cve_emisor                     CHAR(8)           NOT NULL,
 status_accion                  NUMBER(1,0)       NOT NULL,
 cve_receptor                   CHAR(8)           NOT NULL,
 tiempo                         CHAR(8)           NULL,
 visible                        CHAR(1)           NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE acciones
    IS 'Created from Entity ACCION by TESIS on 29-APR-92';

REM     Created from Entity ALUMNO by TESIS on 29-APR-92
PROMPT
PROMPT Creating Table ALUMNOS
CREATE TABLE alumnos(
 no_alumno                      NUMBER(6,0)       NOT NULL,
 nombre                         CHAR(32)          NOT NULL,
 no_cliente                     NUMBER(6,0)       NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE alumnos
    IS 'Created from Entity ALUMNO by TESIS on 29-APR-92';
REM
REM     Created from Entity AREA by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table AREAS
CREATE TABLE areas(
 no_area                        NUMBER(2)         NOT NULL,
 nombre                         CHAR(32)          NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE areas
    IS 'Created from Entity AREA by TESIS on 29-APR-92';
```

```
REM
REM      Created from Entity CLIENTE by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table CLIENTES
CREATE TABLE clientes(
 no_cliente                    NUMBER(6,0)       NOT NULL,
 nombre                        CHAR(32)          NOT NULL,
 no_razon_social               NUMBER(6,0)       NOT NULL,
 comentario                    CHAR(80)          NULL,
 direccion1                    CHAR(60)          NULL,
 direccion2                    CHAR(60)          NULL,
 telefono                      CHAR(20)          NULL,
 fax                           CHAR(20)          NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE clientes
    IS 'Created from Entity CLIENTE by TESIS on 29-APR-92';

REM      AUTOCREATED - Intersection table
PROMPT
PROMPT Creating Table CLIENTE_TIPO
CREATE TABLE cliente_tipo(
 no_cliente                    NUMBER(6,0)       NOT NULL,
 cve_tipo                      CHAR(2)           NOT NULL
) TABLESPACE DATOS
PCTFREE  10
;

COMMENT ON TABLE cliente_tipo
    IS 'AUTOCREATED - Intersection table';

REM      Tipo de Contrato Consultoria
PROMPT
PROMPT Creating Table CONSULTORIAS
CREATE TABLE consultorias(
 no_consultoria                NUMBER(6,0)       NOT NULL,
 no_contrato                   NUMBER(6,0)       NOT NULL,
 monto                         NUMBER(12,2)      NULL,
 monto_dolares                 NUMBER(8,0)       NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE consultorias
    IS 'Tipo de Contrato Consultoria';
```

```
REM
REM      Created from Entity CONTACTO by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table CONTACTOS
CREATE TABLE contactos(
 no_contacto                    NUMBER(6,0)        NOT NULL,
 nombre                         CHAR(32)           NOT NULL,
 telefono                       CHAR(20)           NOT NULL,
 no_cliente                     NUMBER(6,0)        NULL,
 direccion1                     CHAR(70)           NULL,
 direccion2                     CHAR(70)           NULL,
 fax                            CHAR(20)           NULL,
 puesto                         CHAR(32)           NULL,
 modem                          CHAR(20)           NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE contactos
    IS 'Created from Entity CONTACTO by TESIS on 29-APR-92';

REM
REM      AUTOCREATED - Intersection table
REM
PROMPT
PROMPT Creating Table CONTACTO_TIPO
CREATE TABLE contacto_tipo(
 cve_tipo                       CHAR(2)            NOT NULL,
 no_contacto                    NUMBER(6,0)        NOT NULL
) TABLESPACE DATOS
PCTFREE  10
;

COMMENT ON TABLE contacto_tipo
    IS 'AUTOCREATED - Intersection table';
```

```
REM
REM      Created from Entity CONTRATO by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table CONTRATOS
CREATE TABLE contratos(
 no_contrato                    NUMBER(6,0)       NOT NULL,
 no_cliente                     NUMBER(6,0)       NOT NULL,
 cve_vendedor                   CHAR(8)           NOT NULL,
 estado                         CHAR(16)          NULL,
 fecha_reconocimiento           DATE              NULL,
 terminos_pago                  CHAR(200)         NULL,
 total_dolares                  NUMBER(8,0)       NULL,
 total_a_pagar                  NUMBER(12,2)      NULL,
 fecha_firma                    DATE              NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE contratos
    IS 'Created from Entity CONTRATO by TESIS on 29-APR-92';

REM
REM      Created from Entity COSTO POR CREDITO by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table COSTOS_POR_CREDITO
CREATE TABLE costos_por_credito(
 fecha_final                    DATE              NOT NULL,
 fecha_inicio                   DATE              NOT NULL,
 costo                          NUMBER(12,2)      NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE costos_por_credito
    IS 'Created from Entity COSTO POR CREDITO by TESIS on
29-APR-92';
REM
REM      Created from Entity CURSO NOMINAL by TESIS on 29-APR-92
PROMPT
PROMPT Creating Table CURSOS_NOMINALES
CREATE TABLE cursos_nominales(
 cve_curso                      CHAR(10)          NOT NULL,
 creditos                       NUMBER(1,0)       NOT NULL,
 nombre                         CHAR(32)          NOT NULL
) TABLESPACE DATOS
;
```

```
COMMENT ON TABLE cursos_nominales
    IS 'Created from Entity CURSO NOMINAL by TESIS on 29-APR-92';

REM
REM     Created from Entity CURSO PROGRAMADO by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table CURSOS_PROGRAMADOS
CREATE TABLE cursos_programados(
 no_curso_programado                NUMBER(6,0)       NOT NULL,
 cupo                               NUMBER(2,0)       NOT NULL,
 cve_curso                          CHAR(10)          NOT NULL,
 fecha_final                        DATE              NOT NULL,
 fecha_inicio                       DATE              NOT NULL,
 cancelado                          CHAR(1)           NOT NULL,
 lugar                              CHAR(40)          NULL,
 horario_inicio                     CHAR(5)           NULL,
 horario_final                      CHAR(5)           NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE cursos_programados
    IS 'Created from Entity CURSO PROGRAMADO by TESIS on
29-APR-92';

REM
REM     Elementos de un contrato educacional
REM
PROMPT
PROMPT Creating Table EDUCACIONAL
CREATE TABLE educacional(
 no_educacional                     NUMBER(6,0)       NOT NULL,
 no_creditos                        NUMBER(4,0)       NOT NULL,
 no_contrato                        NUMBER(6,0)       NOT NULL,
 creditos_restantes                 NUMBER(4,0)       NULL,
 monto                              NUMBER(12,2)      NULL,
 monto_dolares                      NUMBER(8,0)       NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE educacional
    IS 'Elementos de un contrato educacional';

REM
REM     Created from Entity EMPLEADO by TESIS on 29-APR-92
REM
```

```
PROMPT
PROMPT Creating Table EMPLEADOS
CREATE TABLE empleados(
 cve_empleado                    CHAR(8)          NOT NULL,
 nombre                          CHAR(32)         NOT NULL,
 no_area                         NUMBER(2)        NOT NULL,
 fecha_egreso                    DATE             NULL,
 puesto                          CHAR(32)         NULL,
 fecha_ingreso                   DATE             NULL,
 no_empleado                     NUMBER(4)        NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE empleados
    IS 'Created from Entity EMPLEADO by TESIS on 29-APR-92';

REM
REM     AUTOCREATED - Intersection table
REM
PROMPT
PROMPT Creating Table EMP_CURSO
CREATE TABLE emp_curso(
 cve_empleado                    CHAR(8)          NOT NULL,
 no_curso_programado             NUMBER(6,0)      NOT NULL
) TABLESPACE DATOS
PCTFREE  10
;

COMMENT ON TABLE emp_curso
    IS 'AUTOCREATED - Intersection table';

REM
REM        Created from Entity ENCUESTA DE CURSOS by TESIS on
29-APR-92
REM
PROMPT
PROMPT Creating Table ENCUESTAS_CURSOS
CREATE TABLE encuestas_cursos(
 no_encuesta                     NUMBER(4,0)      NOT NULL,
 no_curso_programado             NUMBER(6,0)      NOT NULL,
 no_pregunta                     NUMBER(2,0)      NOT NULL,
 comentario                      CHAR(100)        NULL,
 no_respuesta                    NUMBER(2,0)      NULL
) TABLESPACE DATOS
;
```

```
COMMENT ON TABLE encuestas_cursos
    IS 'Created from Entity ENCUESTA DE CURSOS by TESIS on
29-APR-92';

REM
REM        Created from Entity ENCUESTA DE SOPORTE by TESIS on
29-APR-92
REM
PROMPT
PROMPT Creating Table ENCUESTAS_SOPORTE
CREATE TABLE encuestas_soporte(
 cve_ing_soporte                 CHAR(8)          NOT NULL,
 no_pregunta                     NUMBER(2,0)      NOT NULL,
 no_sat                          NUMBER(6,0)      NOT NULL,
 comentario                      CHAR(100)        NULL,
 respuesta                       NUMBER(1,0)      NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE encuestas_soporte
    IS 'Created from Entity ENCUESTA DE SOPORTE by TESIS on
29-APR-92';

REM
REM      Created from Entity FACTURA by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table FACTURAS
CREATE TABLE facturas(
 no_factura                      NUMBER(6,0)      NOT NULL,
 fecha_facturacion               DATE             NOT NULL,
 no_contrato                     NUMBER(6,0)      NOT NULL,
 total_pesos                     NUMBER(12,2)     NOT NULL,
 total_dolares                   NUMBER(8)        NOT NULL,
 concepto1                       CHAR(60)         NULL,
 fecha_pago                      DATE             NULL,
 viaticos                        NUMBER(12,2)     NULL,
 pece                            NUMBER(2,0)      NULL,
 iva                             NUMBER(2,0)      NULL,
 estado                          CHAR(16)         NULL,
 descuento                       NUMBER(2,0)      NULL,
 concepto2                       CHAR(60)         NULL
) TABLESPACE DATOS
;
```

```
COMMENT ON TABLE facturas
    IS 'Created from Entity FACTURA by TESIS on 29-APR-92';

REM
REM      Created from Entity INFO ACCION by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table INFO_ACCIONES
CREATE TABLE info_acciones(
 no_renglon                     NUMBER(3,0)       NOT NULL,
 no_accion                      NUMBER(3,0)       NOT NULL,
 no_sat                         NUMBER(6,0)       NOT NULL,
 texto                          CHAR(80)          NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE info_acciones
    IS 'Created from Entity INFO ACCION by TESIS on 29-APR-92';

REM
REM      Created from Entity INSCRIPCION by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table INSCRIPCIONES
CREATE TABLE inscripciones(
 no_alumno                      NUMBER(6,0)       NOT NULL,
 no_curso_programado            NUMBER(6,0)       NOT NULL,
 creditos                       NUMBER(2,0)       NOT NULL,
 no_educacional                 NUMBER(6,0)       NOT NULL,
 fecha_inscripcion              DATE              NOT NULL,
 asistencia                     NUMBER(2,0)       NULL,
 calificacion                   NUMBER(2,2)       NULL,
 forma_pago                     CHAR(1)           NULL,
 numero_factura                 NUMBER(6,0)       NULL,
 fecha_confirmacion             DATE              NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE inscripciones
    IS 'Created from Entity INSCRIPCION by TESIS on 29-APR-92';
```

```
REM
REM      Contrato de Tipo Licencia
REM
PROMPT
PROMPT Creating Table LICENCIAS
CREATE TABLE licencias(
 no_licencia                    NUMBER(6,0)       NOT NULL,
 no_contrato                    NUMBER(6,0)       NOT NULL,
 monto                          NUMBER(12,2)      NULL,
 monto_dolares                  NUMBER(8,0)       NULL,
 usuarios            NUMBER(3)          NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE licencias
    IS 'Contrato de Tipo Licencia';

REM
REM        Created from Entity NIVEL DE PLATAFORMA by TESIS on
29-APR-92
REM
PROMPT
PROMPT Creating Table NIVELES_PLATAFORMA
CREATE TABLE niveles_plataforma(
 cve_ing_soporte                CHAR(8)           NOT NULL,
 no_plataforma                  NUMBER(3)         NOT NULL,
 nivel                          NUMBER(1,0)       NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE niveles_plataforma
    IS 'Created from Entity NIVEL DE PLATAFORMA by TESIS on
29-APR-92';

REM
REM     Created from Entity NIVEL DE PRODUCTO by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table NIVELES_PRODUCTO
CREATE TABLE niveles_producto(
 cve_ing_soporte                CHAR(8)           NOT NULL,
 no_producto                    NUMBER(3)         NOT NULL,
 nivel                          NUMBER(1,0)       NOT NULL
) TABLESPACE DATOS
;
```

```
COMMENT ON TABLE niveles_producto
    IS 'Created from Entity NIVEL DE PRODUCTO by TESIS on
29-APR-92';

REM
REM      Created from Entity PLATAFORMA by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table PLATAFORMAS
CREATE TABLE plataformas(
 no_plataforma                  NUMBER(3)        NOT NULL,
 descripcion                    CHAR(50)         NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE plataformas
    IS 'Created from Entity PLATAFORMA by TESIS on 29-APR-92';

REM
REM        Created from Entity PLATAFORMA CLIENTE by TESIS on
29-APR-92
REM
PROMPT
PROMPT Creating Table PLATAFORMAS_CLIENTE
CREATE TABLE plataformas_cliente(
 no_sopt_clie                   NUMBER(6,0)      NOT NULL,
 cve_serie                      CHAR(32)         NOT NULL,
 version_so                     CHAR(20)         NOT NULL,
 no_plataforma                  NUMBER(3)        NOT NULL,
 medio_distribucion             CHAR(20)         NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE plataformas_cliente
    IS 'Created from Entity PLATAFORMA CLIENTE by TESIS on
29-APR-92';
REM
REM      Created from Entity PREGUNTA CURSOS by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table PREGUNTAS_CURSOS
CREATE TABLE preguntas_cursos(
 no_pregunta                    NUMBER(2,0)      NOT NULL,
 pregunta                       CHAR(100)        NOT NULL
) TABLESPACE DATOS
;
```

```
COMMENT ON TABLE preguntas_cursos
    IS 'Created from Entity PREGUNTA CURSOS by TESIS on 29-APR-92';

REM
REM     Created from Entity PREGUNTA SOPORTE by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table PREGUNTAS_SOPORTE
CREATE TABLE preguntas_soporte(
 no_pregunta                    NUMBER(2,0)       NOT NULL,
 pregunta                       CHAR(100)         NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE preguntas_soporte
    IS 'Created from Entity PREGUNTA SOPORTE by TESIS on
29-APR-92';

REM
REM     Created from Entity PRIORIDAD by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table PRIORIDADES
CREATE TABLE prioridades(
 no_prioridad                   NUMBER(2)         NOT NULL,
 descripcion                    CHAR(50)          NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE prioridades
    IS 'Created from Entity PRIORIDAD by TESIS on 29-APR-92';

REM
REM     Created from Entity PRODUCTO by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table PRODUCTOS
CREATE TABLE productos(
 no_producto                    NUMBER(3)         NOT NULL,
 descripcion                    CHAR(50)          NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE productos
    IS 'Created from Entity PRODUCTO by TESIS on 29-APR-92';
```

```
REM
REM        Created from Entity PRODUCTO CONTRATADO by TESIS on
29-APR-92
REM
PROMPT
PROMPT Creating Table PRODUCTOS_CONTRATADOS
CREATE TABLE productos_contratados(
 no_partida                     NUMBER(6,0)       NOT NULL,
 no_sopt_clie                   NUMBER(6,0)       NULL,
 no_licencia                    NUMBER(6,0)       NOT NULL,
 no_producto                    NUMBER(3)         NOT NULL,
 costo_dolares                  NUMBER(8,0)       NULL,
 no_soporte                     NUMBER(6,0)       NULL,
 version                        CHAR(20)          NULL,
 costo_pesos                    NUMBER(12,2)      NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE productos_contratados
    IS 'Created from Entity PRODUCTO CONTRATADO by TESIS on
29-APR-92';

REM
REM     Created from Entity RAZON SOCIAL by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table RAZON_SOCIAL
CREATE TABLE razon_social(
 no_razon_social                NUMBER(6,0)       NOT NULL,
 nombre                         CHAR(60)          NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE razon_social
    IS 'Created from Entity RAZON SOCIAL by TESIS on 29-APR-92';
```

```
REM
REM       Created from Entity SOLICITUD DE ASISTENCIA TECNICA by
TESIS on 29-APR-
REM - 92
REM
PROMPT
PROMPT Creating Table SATS
CREATE TABLE sats(
 no_sat                          NUMBER(6,0)       NOT NULL,
 encabezado                      CHAR(80)          NOT NULL,
 fecha_creacion                  DATE              NOT NULL,
 nombre_contacto                 CHAR(32)          NOT NULL,
 nombre_cliente                  CHAR(32)          NOT NULL,
 no_prioridad                    NUMBER(1,0)       NOT NULL,
 no_status                       NUMBER(2,0)       NOT NULL,
 no_producto                     NUMBER(3)         NOT NULL,
 fecha_actualizacion             DATE              NOT NULL,
 telefono                        CHAR(20)          NOT NULL,
 cve_ing_soporte                 CHAR(8)           NULL,
 no_soporte                      NUMBER(6,0)       NOT NULL,
 no_plataforma                   NUMBER(3)         NOT NULL,
 cve_responsable_escalamiento    CHAR(8)           NULL,
 comentario                      CHAR(80)          NULL,
 cve_error                       CHAR(3)           NULL,
 cve_pcr                         CHAR(7)           NULL,
 cve_serie                       CHAR(32)          NULL,
 medio_distribucion              CHAR(20)          NULL,
 fecha_escalamiento              DATE              NULL,
 version_so                      CHAR(20)          NULL,
 version_rdbms                   CHAR(20)          NOT NULL,
 version_producto                CHAR(20)          NOT NULL,
 no_error                        NUMBER(6,0)       NULL,
 modem                           CHAR(20)          NULL,
 fax                             CHAR(20)          NULL,
 escalamiento                    NUMBER(1,0)       NULL,
 cve_tar                         CHAR(11)          NULL,
 direccion1                      CHAR(60)          NULL,
 direccion2                      CHAR(60)          NULL,
 nivel                           NUMBER(1)         NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE sats
    IS 'Created from Entity SOLICITUD DE ASISTENCIA TECNICA by
TESIS on 29-APR-92';
```

```
REM
REM      Created from Entity SOPORTE by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table SOPORTES
CREATE TABLE soportes(
 no_soporte                    NUMBER(6,0)       NOT NULL,
 fecha_termino                 DATE              NOT NULL,
 no_contrato                   NUMBER(6,0)       NOT NULL,
 monto                         NUMBER(12,2)      NULL,
 no_sopt_clie                  NUMBER(6,0)       NULL,
 monto_dolares                 NUMBER(8,0)       NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE soportes
    IS 'Created from Entity SOPORTE by TESIS on 29-APR-92';

REM
REM      Created from Entity STATUS DE LA SAT by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table STATUS_SATS
CREATE TABLE status_sats(
 no_status                     NUMBER(2,0)       NOT NULL,
 descripcion                   CHAR(50)          NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE status_sats
    IS 'Created from Entity STATUS DE LA SAT by TESIS on
29-APR-92';

REM
REM      Created from Entity TIPO CLIENTE by TESIS on 29-APR-92
REM
PROMPT
PROMPT Creating Table TIPOS_D_CLIENTE
CREATE TABLE tipos_d_cliente(
 clave                         CHAR(2)           NOT NULL,
 descripcion                   CHAR(32)          NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE tipos_d_cliente
    IS 'Created from Entity TIPO CLIENTE by TESIS on 29-APR-92';
```

```
REM
REM      Created from Entity TIPO CONTACTO by TESIS on 29-APR-92
REM .
PROMPT
PROMPT Creating Table TIPOS_D_CONTACTO
CREATE TABLE tipos_d_contacto(
 clave                          CHAR(2)          NOT NULL,
 descripcion                    CHAR(32)         NOT NULL
) TABLESPACE DATOS
;

COMMENT ON TABLE tipos_d_contacto
    IS 'Created from Entity TIPO CONTACTO by TESIS on 29-APR-92';
```

```
REM
REM This ORACLE V6 RDBMS command file was generated by
REM CASE*Dictionary                          on  12-SEP-92
REM
REM For application system ORACLE version 1
REM
SET SCAN OFF

REM  Objects being generated in this file are:-
REM SEQUENCE
REM       NO_ALUMNO
REM       NO_CLIENTE
REM       NO_CONS
REM       NO_CONTACTO
REM       NO_CONTRATO
REM       NO_CURSO_PROG
REM       NO_EDU
REM       NO_LIC
REM       NO_PLAT_CLIE
REM       NO_RAZON_SOCIAL
REM       NO_SAT
REM       NO_SOP


REM
REM Secuencia asociada a la entidad ALUMNO
REM
PROMPT
PROMPT Creating Sequence NO_ALUMNO
CREATE SEQUENCE no_alumno
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;
```

```
REM
REM Secuencia asociada a la llave de la entidad cliente
REM
PROMPT
PROMPT Creating Sequence NO_CLIENTE
CREATE SEQUENCE no_cliente
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 ORDER
;


REM
REM Secuencia asociada a los numeros unicos de la entidad
Consultoria
REM
PROMPT
PROMPT Creating Sequence NO_CONS
CREATE SEQUENCE no_cons
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;

REM
REM Secuencia asociada a los identificadores de la entidad contacto
REM
PROMPT
PROMPT Creating Sequence NO_CONTACTO
CREATE SEQUENCE no_contacto
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;
```

```
REM
REM Secuencia asociada al identificador de la entidad Contratos
REM
PROMPT
PROMPT Creating Sequence NO_CONTRATO
CREATE SEQUENCE no_contrato
 INCREMENT BY 1
 START WITH 1000
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;


REM
REM Secuencia asociada al identificador de Cursos Programados
REM
PROMPT
PROMPT Creating Sequence NO_CURSO_PROG
CREATE SEQUENCE no_curso_prog
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 CACHE 50
 NOORDER
;

REM
REM  Secuencia  asociada  al  identificador  unico  de  contratos
educacionales
REM
PROMPT
PROMPT Creating Sequence NO_EDU
CREATE SEQUENCE no_edu
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;
```

```
REM
REM Secuencia asociado a los contratos de tipo licencia
REM
PROMPT
PROMPT Creating Sequence NO_LIC
CREATE SEQUENCE no_lic
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;

REM
REM  Secuencia  asociada  al  numero  unico  de  soporte  para  la
plataforma del client
REM - e
REM
PROMPT
PROMPT Creating Sequence NO_PLAT_CLIE
CREATE SEQUENCE no_plat_clie
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;

REM
REM Secuencia asociada a la entidad Razon Social
REM
PROMPT
PROMPT Creating Sequence NO_RAZON_SOCIAL
CREATE SEQUENCE no_razon_social
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;
```

```
REM
REM Secuencia asociada al identificador unico de una SAT
REM
PROMPT
PROMPT Creating Sequence NO_SAT
CREATE SEQUENCE no_sat
 INCREMENT BY 1
 START WITH 2500
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 CACHE 50
 NOORDER
;


REM
REM Secuencia Asociada al identificador de un contrato de tipo
soporte tecnico
REM
PROMPT
PROMPT Creating Sequence NO_SOP
CREATE SEQUENCE no_sop
 INCREMENT BY 1
 START WITH 1
 MINVALUE 1
 NOMAXVALUE
 NOCYCLE
 NOCACHE
 NOORDER
;

REM
REM  End of command file
REM
EXIT
```

```
REM
REM   Dar acceso a todos los usuarios del sistema a
REM   las secuencias generadas
REM
grant select on NO_ALUMNO to public;

grant select on NO_CLIENTE to public;

grant select on NO_CONS to public;

grant select on NO_CONTACTO to public;

grant select on NO_CONTRATO to public;

grant select on NO_CURSO_PROG to public;

grant select on NO_EDU to public;

grant select on NO_LIC to public;

grant select on NO_PLAT_CLIE to public;

grant select on NO_RAZON_SOCIAL to public;

grant select on NO_SAT to public;

grant select on NO_SOP to public;
```